8-2015

Recovery of Crash Site Web Application

Shilpi Jain
University of Redlands

University of Redlands

Recovery of Crash Site Web Application

A Major Individual Project submitted in partial satisfaction of the requirements for the degree of Master of Science in Geographic Information Systems

by
Shilpi Jain

Fang Ren, Ph.D., Committee Chair

Mark Kumler, Ph.D.

August 2015

Recovery of Crash Site Web Application

The report of Shilpi Jain is approved.

[illegible]

Mark Kumler, Ph.D.

[illegible]

Fang Ren, Ph.D., Committee Chair

August 2015

Acknowledgements

I would like to express gratitude to my advisor, Dr. Fang Ren, who supported me throughout my thesis with her expertise and knowledge. I appreciate her assistance in designing the model for this project and writing reports. I would like to thank Dr. Mark Kumler for taking time out from his busy schedule to assist me with designing the poster and serving as my second reader.

Any project begins with an idea. I am grateful to Mr. Fon Allan Duke, the Manager at Mojave Desert Ecosystem Program (MDEP), for bringing this project to the table, providing all the required data, and promoting the web application.

This project included various programming languages and making them coordinate together was a huge task. For their patience and insights while in the development phase, a special thanks to Mr. Jason Boggs from MDEP and Mr. Nathan Strout from University of Redlands.

The following sloka is best to acknowledge my parents, Sarita Jain and Sudhir Jain.

श्रीमदभागवत 10455

सर्वार्थसंभवो देहो जनित: पोषितो यत: |

न तयोर्याति निर्वेशं पित्रोर्मर्त्य: शतायुषा ||

It means - A mortal (a man) with the life of one hundred years even, cannot be free from the debts of his parents, from whom the body, which is the root of the four principal objects of human life (Dharma, Artha, Kaama and Moksha), has originated and by whom it has been nourished.

I love my little brother Saransh, whose unconditional support and love made me believe in myself. Completing the thesis and defending it in time was a lot of work for 10 months and a little intense for me. I have been blessed with wonderful friends (Neel, Karina, Nick, John, and JP) who helped me to go through this journey of ups and downs.

Finally, I would like to thank my Guru, Mrs. Ramya M. B., who introduced me to this amazing field of GIS and Mrs. Ritu Malik.

Abstract

Recovery of Crash Site Web Application

by
Shilpi Jain

Any airplane crash is followed by investigations, search and rescue operations, and clean ups. Determining the debris concentration and affected areas is a challenge for the first responders and clean-up crews as they work based on their past experiences. Recovery of Crash Site is a web application which projects the estimated debris area using the various crash characteristics. It is developed to help the first responders, clean-up crews, and investigators to initiate the search for the debris and work more efficiently by reducing the guesswork. Testing the application on various devices gives the results which can be very helpful for transportation safety and aviation industry.

Table of Contents

Table of Figures

List of Tables

List of Acronyms and Definitions

ACTA	Asteroid and Comet Tracking Agency
AFB	Air Force Base
AGL	Above Ground Level
C_d	Drag Coefficient
CRTF	Common Real-Time Footprint
CSDRMA	Crash Site Debris Recovery Mobile Application
CSS	Cascade Style Sheet
ER model	Entity-relationship model
FAA	Federal Aviation Administration
GIS	Geographic Information System
HTML	HyperText Markup Language
JPG	Joint Photographic Experts Group
js	JavaScript File
kts	Knots
ldf	Log Data File
MDEP	Mojave Desert Ecosystem Program
mdf	Master Data File
MS	Microsoft
MSL	Mean Sea Level
NASA	National Aeronautics and Space Administration
NGA	National Geospatial-Intelligence Agency
NTSB	National Transportation Safety Board
PHP	Hypertext Preprocessor
py	Python File
RoCS	Recovery of Crash Site
SAR	Search and Rescue
SQL	Structured Query Language
TAP	Trajectory Analysis Program
tbx	ArcMap Toolbox File
URL	Uniform Resource Locator
WGS	World Geodetic System

Chapter 1 – Introduction

Various factors, such as human error like air traffic control error, design flaw, explosive device on board, fuel starvation, hijacking, and unsuitable weather conditions like lightning, heavy rains or bird strikes can cause an airplane crash. Investigations following any airplane crash range from searching debris area for crashes to analyzing and mapping them. An important part of these investigations is locating areas with high concentrations of debris. Collecting debris from the incident area is a very high priority task for two reasons. First, the authorities must rebuild the plane from recovered debris in order to determine the cause of the crash. Second, the debris may contain some toxic materials that may endanger the surrounding areas. The area may also contain human remains or survivors, and identifying the debris area can help the first responders rescue as many people as possible.

Immediately after a crash, authorized personnel give an approximate location of the crash to first responders who search the area for debris, pilot, and passengers. Without a model to inform their search they use their personal experience to approximate the location of the debris, which may result in a delay in providing medical help to injured passengers. This project aimed to develop a web application that generates the debris area, which can considerably reduce the time taken by the first responders and help them work more efficiently.

1.1 Client

The Mojave Desert Ecosystem Program (MDEP) was the client for this project and Mr. Fon Allan Duke, Project Manager, was the point of contact. This organization provides solutions and services to government agencies to ensure their smooth functioning (Mojave Desert Ecosystem Program, n.d.). In 2012, Steve Mesa, formally with the National Geospatial-Intelligence Agency (NGA) out of Nellis Air Force Base (AFB), expressed interest to MDEP for an application which could help first responders recover aircraft debris more efficiently (F. A. Duke, personal communication, July 7, 2015). Fon Allan Duke identified the need for a mobile application that generated probable debris area of an airplane crash. The original application was developed by Nicholas Janzen in 2012 on the Windows mobile platform. Since the application was a native mobile application, the target audience was very limited. In 2015, to overcome this issue, Fon Allan Duke recognized the need for a web application which is platform independent.

1.2 Problem Statement

The problem addressed in this project is the difficulty faced by first responders and investigators of an airplane crash to make an estimate of the debris area. In addition, the operational environment of the existing Windows mobile application hinders the use of the application by a broader audience. Therefore, a program independent of operating system was in demand which would have a widespread use so that the users can have access to the application from any device or browser.

1.3 Proposed Solution

The proposed solution was to develop a JavaScript web application with Cascade Style Sheet (CSS), HyperText Markup Language (HTML), Hypertext Preprocessor (PHP), and ArcGIS geoprocessing service. Any device with a HTML compliant web browser and GPS and internet connectivity can use this application. The point selection map works by geo-locating from the on-board GPS. It will not function without that feature enabled. An additional feature of the web application would consider the impact of various terrain factors on the debris area.

1.3.1 Goals and Objectives

The main goal of this project was to develop a solution for the first responders and investigators of an airplane crash, to estimate the debris area. To fulfill this goal, there were three objectives. The first was to develop a MS SQL 2008 R2 database that stores information about airplanes. The second was to publish a geoprocessing tool that generated the debris area using a python script. The last was to develop a web application that is platform independent, takes various inputs, and displays the debris area.

1.3.2 Scope

The scope of the project outlines the deliverables of the project as well as the constraints considered in the project. Deliverables of this project include a geoprocessing service, a web application, and a non-spatial SQL database. The geoprocessing service calculates the debris area according to the input of the user. The geographic information system (GIS) web application was the main deliverable, which executes the published geoprocessing service and allows the users to visualize the calculated debris area. Any device with a working internet connection, a browser, and a GPS can access the application. The application relies on a database of airplanes which was developed in Microsoft SQL 2008 R2.

There are a few constraints applied to the web application. The airplane models are confined to the model information provided by the client. If a required airplane is not present in the database, the user can add it through the provided link in the web application. The debris area calculated by the geoprocessing service considers the terrain angle which is uphill or downhill, but it does not consider the aspect of terrain because of the limitations of the adopted debris model. The area calculated depends solely on the values entered by the user.

1.3.3 Methods

The project had clear and fixed requirements from the beginning, so a waterfall model was utilized. The project required limited collaboration with the client and the technology was well understood. Figure 1-1 shows the phases of this methodology. Since none of the phases overlap, this model was best suited for his project. Each phase is processed at a time and has a specific deliverable.

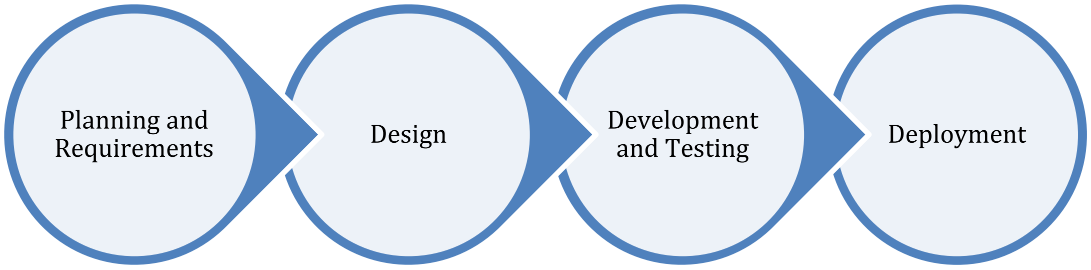


Figure 1-1: Waterfall Model.

The first phase, planning and requirements, dealt with requirement gathering and planning milestones which helped in on-time delivery of the application. During the following design phase, the layout of the web application was finalized in consultation with the client. For the third phase, a web application was developed in JavaScript, Hypertext Preprocessor (PHP), HyperText Markup Language (HTML), and Cascade Style Sheet (CSS). Various parameters are taken as an input from the user, which are used to call the published geoprocessing service, and the resultant debris area is displayed on a map. Users can save this map as a jpg image file or email it. For the final phase deployment, the geoprocessing service and the web application were published on the client's server.

1.4 Audience

The users of this web application will include non-technical GIS individuals and GIS professionals. Users can use the application to obtain the debris area or as a supplement to analyze various aspects, like affected population and terrain of the area, so that the first responders can reach the destination in time.

Users can be employees of the US Air Force or the National Geospatial-Intelligence Agency, and others who knows how to operate the web site.

1.5 Overview of the Rest of this Report

Chapter 2 gives a background of airplane crash modeling. Chapter 3 discusses the various steps of the waterfall model implemented to deliver the project. Chapter 4 provides details about the data model used. Chapter 5 presents the implementation process used for this project. Chapter 6 defines the results and analysis. Chapter 7 concludes the report and lists the future work.

Chapter 2 – Background and Literature Review

GIS techniques are very useful in search and rescue (SAR) operations because they can produce results very quickly by specifying the incident area and finding the optimal route to it. This, in turn, helps to decrease search time and reduce search efforts (Söylemez & Usul, 2006). This chapter gives contextual details about the models and technologies used in this project to create the Recovery of Crash Site (RoCS) web application. The Chapter is organized as follows. Section 2.1 introduces various debris models considered to create this application. Section 2.2 gives details about the terrain model incorporated in the application. The following section explains relationship between GIS and web technology and application of web GIS. The chapter ends with a summary.

2.1 Debris Modeling

The research and empirical work done for debris modeling of airplane crashes is very limited. The major development in this field was after the 2003 NASA Space Shuttle Columbia disaster. This section gives details about the debris models considered for this project.

2.1.1 Trajectory Analysis Program

The Federal Aviation Administration (FAA) and National Transportation Safety Board (NTSB) developed a formula that provides adequate spectator separation distances for an Air Race type of event to make sure that the airshow spectators are safe from any possible accident during the event. Spectator separation distance can be defined as distance between the spectator area of an event and the area where the event is taking place. The FAA formula is as follows:

$$Scatter\ Distance = \frac{Aircraft\ Speed\ \sqrt{2\ (Aircraft\ Altitude)}}{32.2}$$

Calculated scatter distance is in feet, when aircraft speed is in miles per hour and aircraft altitude is in feet. However, it does not consider all the variables involved in the airshow environment, such as weight, frontal area and drag characteristics of the projectile (Oldham, 1990).

The FAA formula was customized into a new program to yield a more accurate scatter distance. This new program named the Trajectory Analysis Program (TAP) was designed using the Basic programming language for airshow environment and in-flight disintegration by Oldham in 1990. Variables considered in the TAP model include initial altitude of disintegration (in feet AGL), initial density altitude (in feet AGL), altitude of impact at ground level (in MSL feet), wind velocity (in knots), wind direction (in degrees), airspeed at disintegration (in knots), rate of climb or sink at disintegration (in degrees), projectile weight (in pounds), projectile drag coefficient, and projectile frontal area (in square feet). The outputs given by the model are the horizontal distance from disintegration at impact (in feet), the total velocity (in knots), the terminal velocity (in knots), the time of fall (in seconds), the flight path angle at impact (in degrees), and the ground speed of projectile at impact (in miles per hour) (Oldham, 1990).

2.1.2 Common Real-Time Footprint

The Asteroid and Comet Tracking Agency (ACTA) developed the Common Real-Time Footprint (CRTF) program for risk analysis at the Air Force Eastern and Western Ranges. This program generates dispersion footprints and impact probability contours to define the hazard areas after a vehicle breakup. CRTF models various uncertainties like real-time vehicle state vector, course change at the time of malfunction, uncertainty of fragments, ballistic coefficient, lift effect of fragments, and wind (Lin, Larson, & Collins, 2003).

NASA uses CRTF to calculate safe flying distances during the launch and reentry of Reusable Launch Vehicles. However, CRTF cannot be used for the development of the RoCS web application because being a proprietary software the algorithms of CRTF are not released for public (Sarconi, 2013).

2.1.3 Crash Site Debris Recovery Mobile Application

Janzen (2012) developed the Crash Site Debris Recovery Mobile Application (CSDRMA) as a thesis project. This was a Windows mobile application developed for the Mojave Desert Ecosystem Program (MDEP). The application projects an estimated area of high debris concentration using the GPS location of the mobile device, a geoprocessing service, and user-defined variables of affected aircraft and its crash characteristics. The debris trajectory formula employed in the calculation was adopted from the Trajectory Analysis Program mentioned earlier, with an expansion to include the impact of terrain as described in the next section. In addition, four accuracy test cases were conducted using ArcMap. The first test was on the 1991 Boeing 737-200 crash which occurred at Colorado Springs Municipal Airport, Colorado. The actual debris area of this airplane crash was within the results of the area generated in CSDRMA. The next test involved a Hawker Beechcraft 125-800A that crashed at Owatonna Degner Regional Airport, Minnesota in 2008. The actual debris area of this crash was much larger than the area generated in CSDRMA. The third test was carried out on a Lockheed Martin F-22A Raptor crash that occurred in Alaska in 2010. The actual debris area did not match the generated test area. The last test was on a Piper PA-31 Navajo that crashed near Mauna Loa volcano, Hawaii in 1999. The debris area of this crash partially coincided with the estimated area. Therefore, two out of these four test cases were considered as reasonably successful.

Unfortunately, the majority of targeted users were not able to use this application because the application was developed for only Windows Phone 7. Given this limitation, there was a need to develop similar applications that are platform independent. Since the problem addressed in Janzen (2012) was similar to that of RoCS, the geoprocessing service of CSDRMA was incorporated in RoCS application with some modifications.

2.2 Impact of Terrain on Crash Site

Terrain plays an important role while calculating the impact of an accident, whether it is a car crash, an airplane crash, or any other. Different angles and directions of terrain can result in very different debris area even if all the other parameters are same. To include the effect of terrain on debris area of an airplane crash, Janzen (2012) used terrain angle and terrain characteristic as uphill or downhill.

There are various investigation handbooks written for the investigation of airplane crashes by National Transportation Safety Board, US Army, and other organizations. These handbooks describe how terrain can be incorporated while investigating the impact of an airplane crash. For a clear understanding, the following definitions are included (Coltman, Ingen, Johnson, & Zimmermann, 1989).

- Flight path angle is defined as the angle between the aircraft flight path and the horizontal at the moment of impact.
- Terrain angle is defined as the angle between the impact surface and the horizontal, measured in a vertical plane.
- Impact angle is defined as the angle between the flight path and the terrain, measured in a vertical plane. It is the algebraic sum of the flight path angle and the terrain angle.

Figure 2-1 gives a visual interpretation of the definitions. If all the other parameters remain unchanged, uphill terrain at any angle will reverse the direction of debris area, whereas downhill terrain increases the debris spread in the same direction. Since uphill increases the impact angle, the resultant debris area is smaller than on flat terrain. On the contrary, downhill terrain yields a larger debris area because it decreases the impact angle. Terrain characteristics were considered in this project as well.

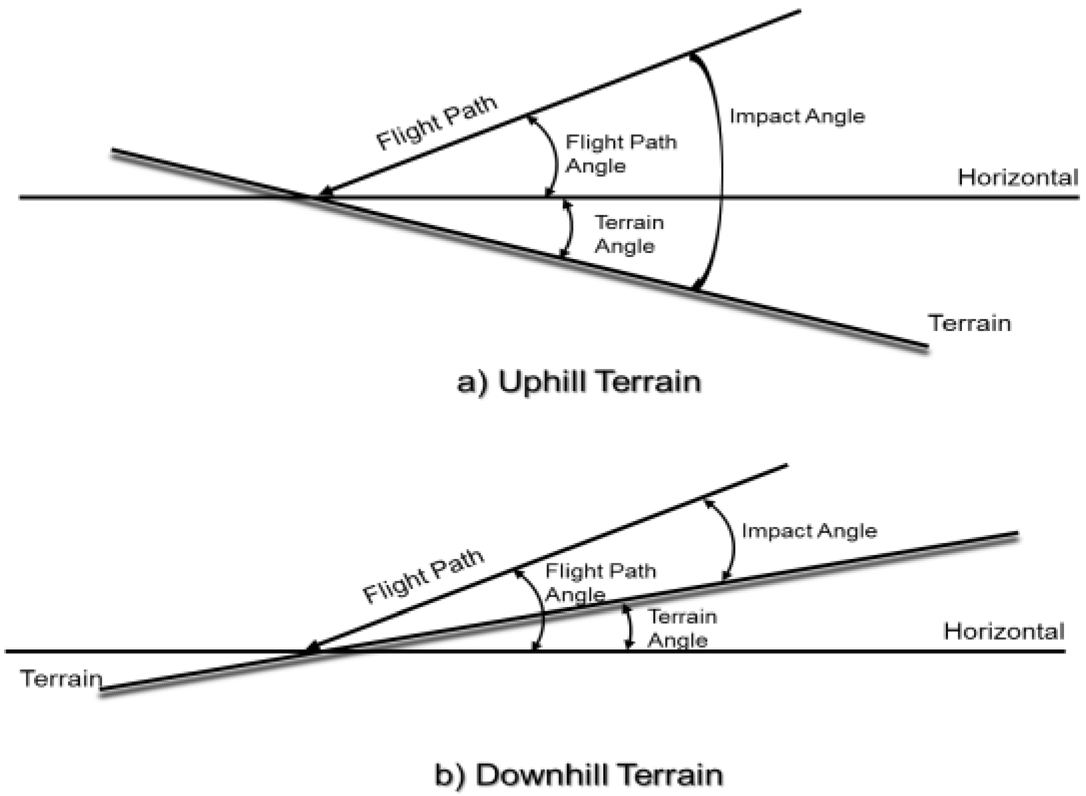


Figure 2-1: Relation between Terrain and Flight Path

2.3 Geographic Information System and Web

Any GIS that uses web technology to communicate between components is termed as Web GIS (Fu & Sun, 2011). Integrating GIS and the web technology opens the gate to many opportunities as people can visualize, analyze, and even build data online. This helps to increase the target audience for GIS applications. Moreover, it aids in building data by people rather than by formal data producers so that data are accessible freely. The basic structure of web GIS is shown in Figure 2-2 (adapted from Fu & Sun, 2011). It includes a database server, a GIS server, a web server, the Internet, and client(s). The architecture shown in the figure is three-tier, which means data, logic, and presentation are on separate computers. According to the requirement of an application and availability of resources, web GIS can be implemented as one-tier or two-tier as well.

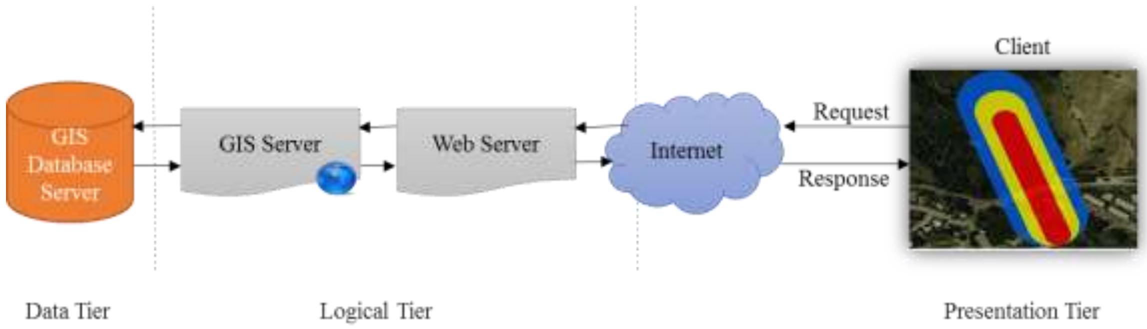


Figure 2-2: Architecture of Web GIS (adapted from Fu & Sun, 2011)

There are many web-based GIS applications used throughout the world in various fields. The government uses them for public information services, communication with people in a more efficient way, and as a tool to make decisions (Esri, 2015a). For example, the City of Rancho Cucamonga, California, USA uses interactive GIS web applications such as *My Community* to give people information about zoning, flood zones, general plan, trash pickup zones, and street sweep zones (City of Rancho Cucamonga, 2015). Business organizations use GIS applications to open new stores, increase the target customers, generate routes for distribution trucks, analyze trade areas, and other business operations (Esri, 2015b). An example is banks providing locations of their ATMs and offices as online maps to their customers, such as US Bank (U.S. Bank, 2015). Researchers use these applications to collect and organize data, analyze them, and discover new facts or procedures (Esri, 2015c). For example, Whale mAPP is used by California's Marine Sanctuaries, Point Blue Conservation Science to protect endangered whales (Point Blue Conservation Science, 2015).

Since the requirement of the client to build the RoCS application was to increase the range of target audience, web GIS seemed appropriate approach. Web GIS can be a native mobile application or a web application. A native mobile application is platform dependent. It can work on only the supported mobile operating system (Apple, Android, and Windows etc). A web application is supported on all platforms and devices that have an internet connection and a web browser. Therefore, it was decided to develop the RoCS as a web application because a native mobile application already existed for Windows mobile phones (Janzen, 2012).

2.4 Summary

This chapter reviews various debris models, terrain model, and web GIS. Based on the past work, the web GIS technology and the TAP debris model were chosen to implement the RoCS application to broaden the future users. Due to limitations of the implemented model, effect of obstacles in the debris area is not considered in the application. This web application would include search and rescue operations to produce a geo-referenced area of high concentration of debris so that the first responders and clean-up crews can work more effectively in a given period.

Chapter 3 – Systems Analysis and Design

This chapter defines in detail the problem addressed in this project and the design phase of the project. Section 3.1 revisits the problem statement and Section 3.2 explains the requirements analysis conducted for this project. System design is illustrated in Section 3.3 and Section 3.4 focuses on the project plan. The chapter ends with a summary of the system analysis and design.

3.1 Problem Statement

The problem addressed in this project is the difficulty faced by first responders and investigators of an airplane crash to make an estimate of the debris area. In addition, the operational environment of the existing Windows mobile application hinders the use of the application by a broader audience. Therefore, a program independent of operating system was in demand which would have a widespread use so that the users can have access to the application from any device or browser.

3.2 Requirements Analysis

Requirements analysis is a very crucial phase for a project because misunderstanding of a client's requirements can lead to rework, cost overruns, quality issues, or schedule delays. This section lists various requirements of this project.

3.2.1 Functional Requirements

The functional requirements describe what functions or capabilities the system must provide. Table 3-1 outlines the functional requirements associated with this project. These include zoom into the current location, zooming-in and out on the map, visualizing debris area, emailing debris area, saving the debris area as a JPG file, and switching base maps.

Table 3-1: Functional Requirements

	Functional Requirement	**Description**
1.	Current Location	Application shall zoom in to the current location on *Location* page.
2.	Zoom In/Out	Application shall zoom in/out, whenever zoom in/out button is clicked or by using finger gestures on *Location* or *Results* page.
3.	Pan/Move	Application shall move the map, using mouse or finger gestures on *Location* or *Results* page.
4.	Display Debris Area	Application shall display the calculated area when *Finish* button is clicked.
5.	Email Map	Application shall Email the map, if *Email* button is clicked.
6.	Save Map as JPG	Application shall save the map as a JPG file, if *Print* option is selected.
7.	Switch Basemap	Application shall change the basemap to the one selected from *Switch Basemap* menu.

3.2.2 Non-Functional Requirements

The non-functional requirements focus on how well the system must perform. These requirements relate to interfaces, usability, accessibility, integration, operational environment, performance, security requirements, maintenance, system administration, and documentation. Table 3-2 shows the non-functional requirements associated with this project, which were related to user interface, performance, and responsiveness.

Table 3-2: Non-Functional Requirements

	Non-Functional Requirement	**Description**
1.	User Interface	Application shall be usable by people with no GIS experience.
2.	Performance	Application shall display the calculated debris area within one minute after *Finish* button is clicked.
3.	Responsive	Application shall work on phones, tablets, and laptops.

3.3 System Design

The web application for debris estimation, Recovery of Crash Site (RoCS), is accessible from mobile phones, tablets, computers, or any modern devices that have internet access and a browser. The system design of this web application is displayed in Figure 3-1.

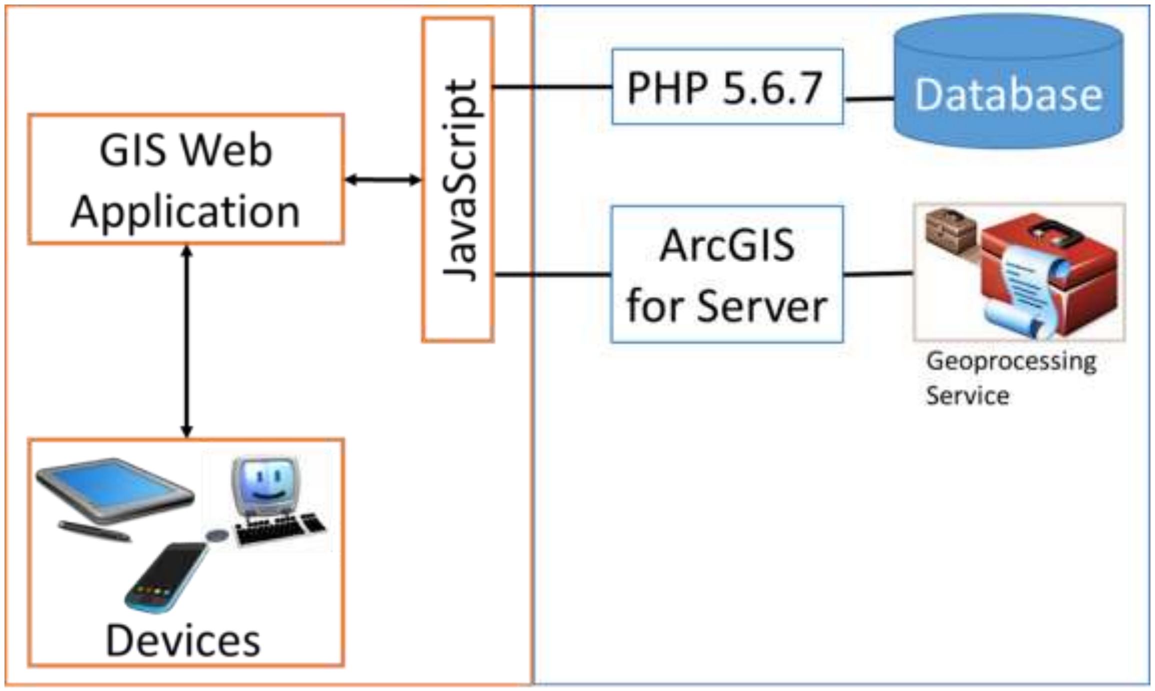


Figure 3-1: System Design.

When the user interacts with RoCS, he/she needs to enter the required parameters for calculating the debris area, such as manufacturer, model, speed, direction heading and angle of descent of the aircraft, speed of wind at ground level, direction of wind at ground level, angle of terrain which is uphill/downhill. These values are at the time of initial impact. Other parameters, including frontal area, drag coefficient, weight, and wingspan of the aircraft are loaded from the MS SQL 2008 R2 database, through PHP (Hypertext Preprocessor) code by selecting the manufacturer and model of the airplane that crashed. Selecting the aircraft manufacturer limited the choices of available models based on the selection. These values are passed to the geoprocessing (GP) service when the *Finish* button is clicked. Published on ArcGIS for Server, the geoprocessing service calculates the debris area based on a formula created by Oldham (1990) and refined by Janzen (2012). The final estimated debris area is displayed on a map. The user can then save this map or email it as a JPG file, as per the requirement.

3.4 Project Plan

A project plan defines all the phases of a project and the respective tasks allocated to them. This section gives details about the project phases, tasks, deliverables, and the assumptions made during development.

3.4.1 Project Phases

Various phases of this project are discussed in this section. Figure 3-2 gives a hierarchical structure of these phases.

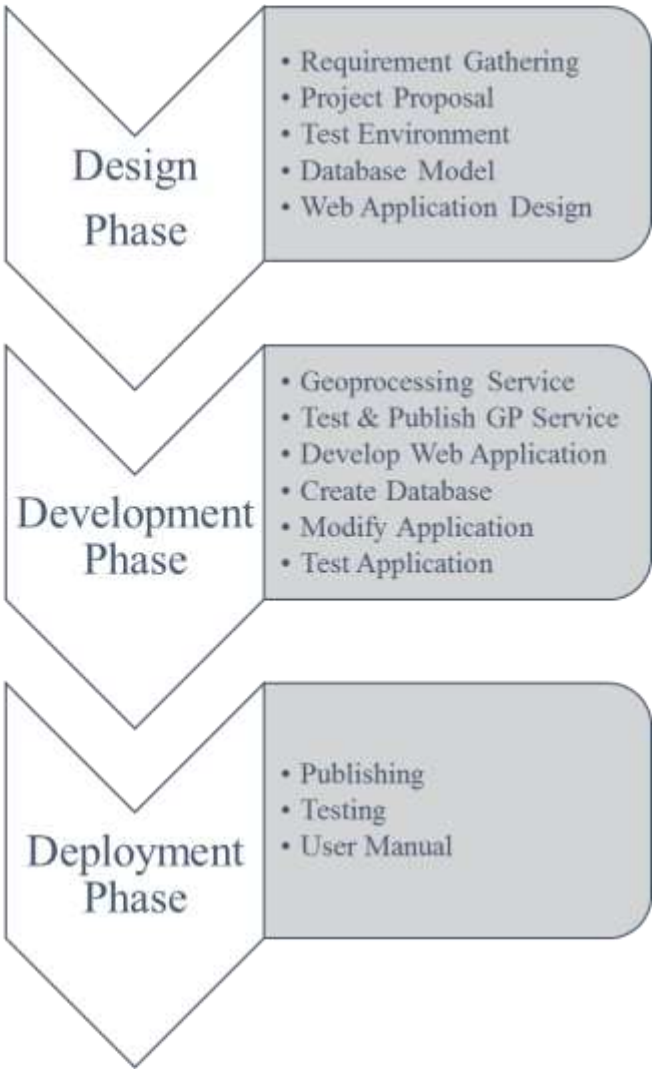


Figure 3-2: Project Phases.

Design phase included gathering requirement and creating a logical model of the project. The first task in this phase was to gather information required to proceed with the development of the application from the client. Since this project was implemented using the waterfall approach, client requirements were finalized by the end of the first month and there were no further changes. For the second task, a document was provided to the client to define the project requirements and the proposed solution and methodology that were adopted. The next task created a testing environment. The project development was done on a local machine, but the client provided a virtual machine, which was a mirror image of the working environment. It was used in further phases to test the application. The fourth task was to define a database model so that the logical structure of the database could be finalized. The model explained how to traverse or modify the database. The look and feel of the web application was one of the focuses for the client, because this application will be used during emergencies. This was the final task of the design phase. The visual structure of the application was developed in HTML (Hypertext Markup Language) 5.0 and CSS (Cascading Style Sheet).

The phase of development included building the physical data model, developing the application and tools, and performing quality checks. The first task of this phase was to develop a geoprocessing service using Python 2.7 to calculate the debris area. The next phase was to test this service. After fixing the identified bugs, the service was published on ArcGIS for Server. The third task was to develop the web application using JavaScript, HTML 5.0, PHP, and CSS for calculating and displaying the debris area. The Trajectory Analysis Program (TAP) developed by Oldham (1990) and the Crash Site Debris Recovery Mobile Application (CSDRMA) developed by Janzen (2012) were used for calculation of the area. The fourth task was to create a non-spatial MS SQL 2008 R2 database according to the designed database model. Next, the application was modified to connect with the database. In addition, a new page was created, through which new airplanes can be entered in the database. The final task was to test the web application.

Deployment was the final phase of this project. Final testing, review, and final implementation were done in this phase. Publishing the application, geoprocessing service, and database was the primary task. It was followed by testing the web application's features and functionalities. The final task was to create a help page within the application to facilitate the use of the application.

3.4.2 Deliverables

The deliverables associated with this project included both tangible and intangible objects. The first deliverable was the web application as a package, consisting of .html, .js, .php, and .css files. The second was the geoprocessing service in .tbx and .py format and published on ArcGIS for Server. The third deliverable was the database of airplanes as a .mdf file, which stores data, and .ldf, which stores the log.

3.4.3 Assumptions

The chief assumption made in this project was regarding the data. It was assumed that the client would provide assistance to collect the data for the database and testing the web application. Another assumption was that the test environment provided by the client as a virtual machine was a mirror image of the live environment.

3.5 Summary

The clearly defined requirements provided by the client reduced the time needed to complete the initial phase of the project. However, using multiple technologies to develop the web application caused some delay as compared to the original project schedule. Apart from this, while developing the web application, the layout was changed multiple times to get the most suitable and functional view. In spite of some challenges, the Recovery of Crash Site (RoCS) web application was developed successfully.

Chapter 4 – Database Design

The database was one of the most important parts of this project. The adopted debris model needs many inputs from the user. Some of these parameters are based on the airplane manufacturer and model. Therefore, to reduce the parameters to be entered by the user, a database was developed. Once the user enters the airplane manufacturer and model, the respective values will be extracted automatically from the database, which simplifies the use of the web application for the users. Section 4.1 gives details about the conceptual model of the project. Section 4.2 defines the logical model of the database. Section 4.3 discusses the data sources and data collection methods. A summary of the chapter is provided in Section 4.4.

4.1 Conceptual Data Model

A conceptual model defines the various entities and their relationships required to demonstrate a problem. In this project, the problem addressed was the difficulty faced by first responders and investigators after an airplane crash to search for debris. Figure 4-1 displays a conceptual model for this project.

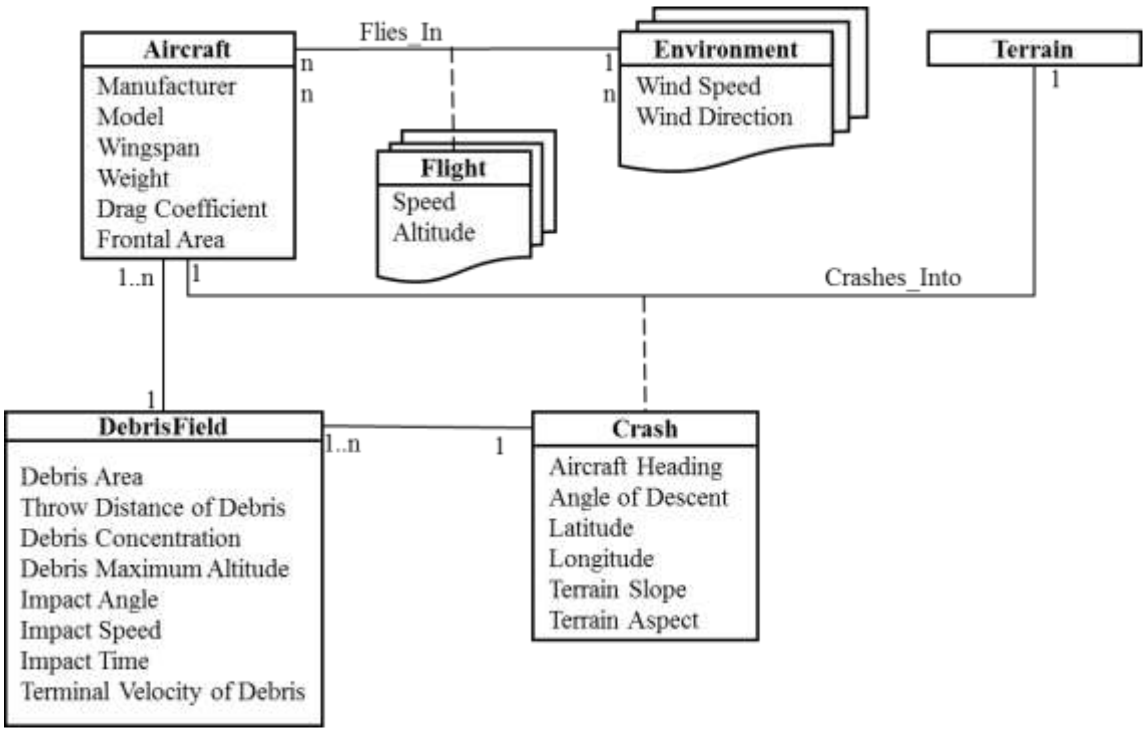


Figure 4-1: Conceptual Data Model.

Aircraft, environment, terrain, flight, crash, and debris field were the entities involved in this project. When an aircraft crashes into terrain due to natural factors or human error a debris field is generated. The debris field is affected by airplane characteristics, flight entities, terrain, and environment. A change in any of these parameters can result in different debris throw distance and debris velocity. The model also describes how these entities are related. The relationship can be one-to-one, one-to-many, many-to-one, or many-to-many. In Figure 4-1, *n* means many and *1..n* means one to any number. For example, one crash can have a single debris field or many. Therefore, the relationship is mentioned as *1* to *1..n*.

4.2 Logical Data Model

A logical data model gives details about the solution provided to resolve the problem. It is a subset of the conceptual model, wherein entities used are explained. Figure 4-2 explains the logical model of this project. To develop the Recovery of Crash Site (RoCS) web application, all the entities mentioned in Section 4.1 were used. However, instead of storing all instances of Environment and Flight, these two entities were considered as single values because only one instant of crash were will be calculated in the web application. In an ideal situation, terrain slope and aspect should be used in crash characteristics, but for this application, only terrain slope and direction (uphill or downhill) were considered and they were entered as single values as well. The Aircrafts entity was designed into a Microsoft SQL 2008 R2 database, which sends the required aircraft related attributes to the web application to generate the desired information about the debris field.

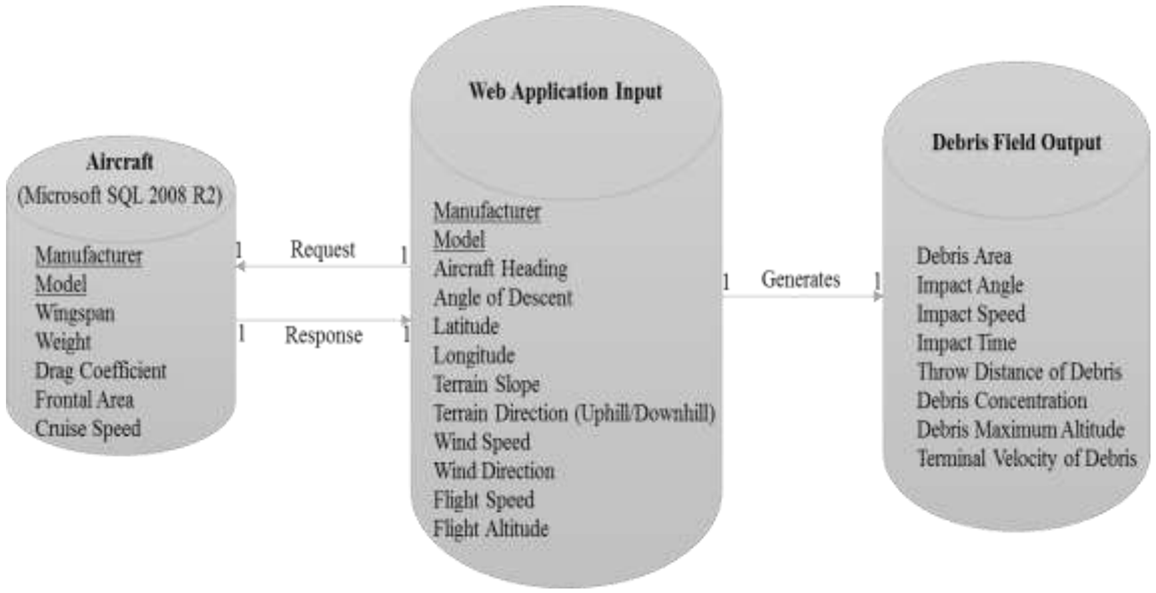


Figure 4-2: Logical Data Model.

In this MS SQL database, a table called Aircraft was developed. The table has eight fields (id, aircraftname, aircraftmodel, aircraftdrag, aircrafftweight, aircrafttwingspan, aircraftfrontal, and cruisespeed) to store aircraft ID, manufacturer, model, drag coefficient, weight, wingspan, frontal area, and speed. ID is the dynamically generated primary key of the table.

4.3 Data Sources and Loading

The client provided the data used for the database and testing. The test data included feature classes of crash locations and actual debris fields for four crashes. The first dataset was of a 1991 Boeing 737-200 crash which occurred at Colorado Springs Municipal Airport, Colorado. The second was of UPS flight 1352, an Airbus A300-600, N155UP, which crashed while landing in 2013 at Birmingham-Shuttlesworth International Airport, Alabama. The third was of a Boeing 777-236ER, which also crashed while landing, in 2008, at London Heathrow Airport. The last was of a crash during an experimental test flight of a Gulfstream Aerospace Corporation GVI (G650) in 2011 at Roswell International Air Center, New Mexico.

There were two feature classes associated with each test case, one for the crash location and other for the actual debris field. Therefore, eight feature classes were used for testing. The results from RoCS were compared to these images to determine accuracy and efficiency.

The test dataset for the Colorado crash was the same as that of Crash Site Debris Recovery Mobile Application (CSDRMA) (Janzen, 2012). The actual debris fields for the Alabama and New Mexico crashes were digitized from the images in the National Transportation Safety Board (NTSB) reports (National Transportation Safety Board, 2001, 2012, 2014). Very few of the NTSB reports include the imagery, some of which were not usable due to various factors such as not having a full picture of the debris field, or having no other reference points to estimate the size of the field. The debris field of the London crash was digitized from the Department for Transport, London report (Department for Transport (Air Accidents Investigation Branch), 2010).

Since the data for the Colorado crash consisted of ellipses to depict the actual debris field, the same format was adopted for the Alabama, New Mexico, and London crashes. Approximate radii of ellipses were used to digitize, based on the debris in images. Refer to Appendix G for the images.

Data for the MS SQL database were collected from various online sources (Federal Aviation Administration, 2009; aerospaceweb.org, 2011; Palt, 2015). In addition, some data are the same as that was used in the Crash Site Debris Recovery Mobile Application (CSDRMA) (Janzen, 2012).

The data provided by the client for the database and testing were loaded directly to the database for testing.

4.4 Summary

There were some differences between the conceptual model and the logical model of the project because of the approach required to address the client’s problem statement. The data provided by the client were in an appropriate format. After testing, the generated debris field was added to the geodatabase as a feature class for further analysis.

Chapter 5 – Implementation

The solution provided to the client was the development of a web application which can be accessed from any device which has an Internet connection, a browser, and GPS capability. Figure 5-1 shows the basic components of this application, including a database, a geoprocessing service, a web application, and users. This chapter will focus on the implementation of the geoprocessing service and the web application. Section 5.1 discusses development of the Recovery of Crash Site (RoCS) Python script tool and publishing it as a geoprocessing service. The next section gives details about the development of the revised RoCS tool that can automatically extract terrain parameters. Section 5.3 explains development of the RoCS web application, using the MS SQL database and the geoprocessing service. The chapter ends with a summary.

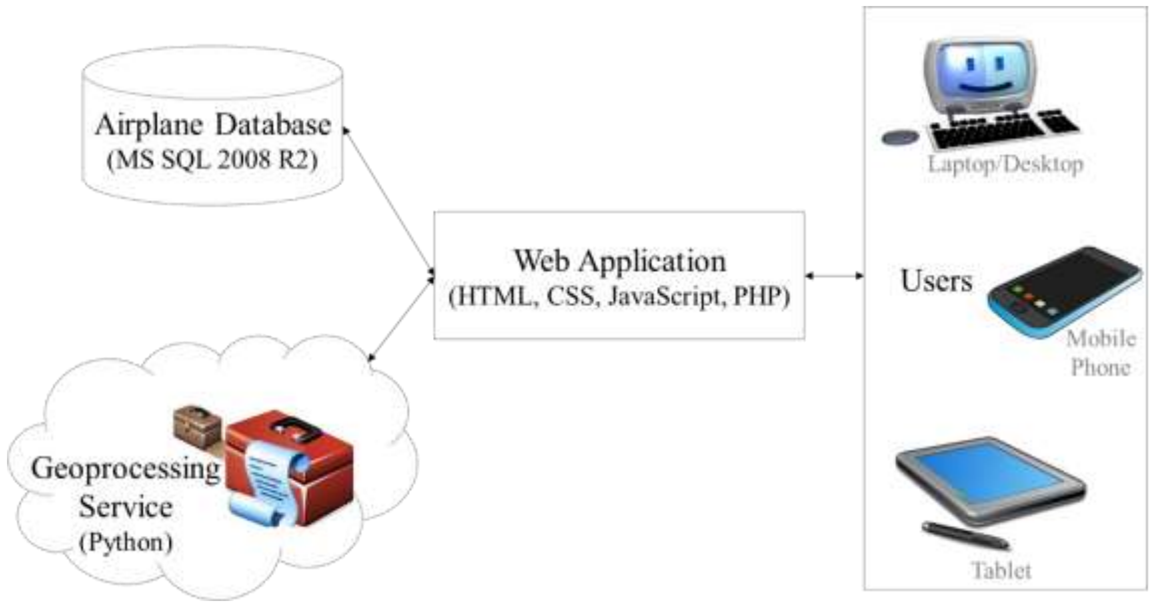


Figure 5-1: Components of RoCS Web Application.

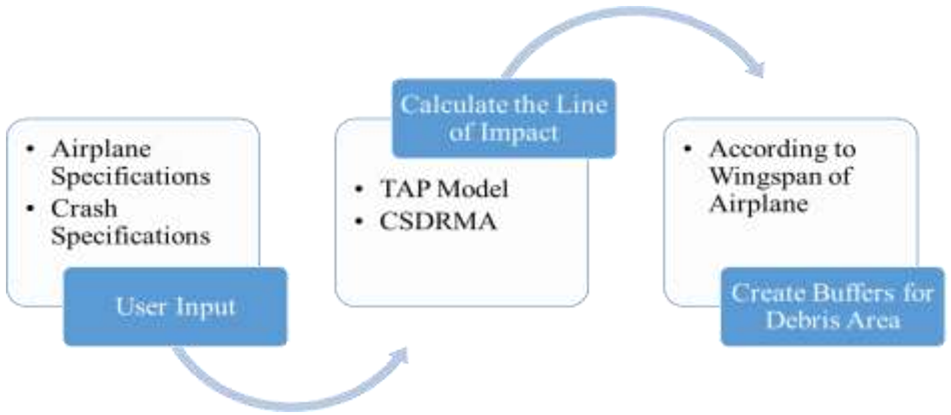
5.1 Developing the RoCS Geoprocessing Service

The RoCS web application sends a request to the geoprocessing service, that calculates the debris area and sends it as a response to the web application. The RoCS geoprocessing service was developed using a Python script and ArcMap 10.2 Toolbox, and then it was published on ArcGIS for Server. The RoCS geoprocessing service was based on the Trajectory Analysis Program (TAP) model and Crash Site Debris Recovery Mobile Application (CSDRMA) mentioned in Section 2.1 and 2.2. Table 5-1 shows the parameters of the geoprocessing service. These comprise the latitude and longitude of crash location, level of ground, wind speed, wind direction, terrain angle, terrain characteristic, and aircraft parameters including speed, altitude, heading, descent, frontal area, drag coefficient, weight, and wingspan. The parameters used must be the ones recorded at the time of impact.

Table 5-1: Parameters for RoCS Geoprocessing Service

Data Source	**Input Parameter (unit)**	**Description**
User Input	Crash Location (a point)	The input feature class comprising of one point location. This location is the first point of contact of airplane with ground during the crash.
	Speed of Aircraft (knots)	The speed of aircraft at the time of impact.
	Altitude of Aircraft (feet AGL)	Altitude of the flight path.
	Aircraft Heading (degrees)	Direction that the aircraft's nose was pointing to. It must lie between 0^o and 360^o.
	Descent of Aircraft (degrees)	Angle of descent. It must lie between 0^o and 90^o.
	Ground Level (feet MSL)	Elevation above mean sea level. It must be less than the altitude of aircraft.
	Ground Level Wind Speed (knots)	Speed of wind at ground level.
	Ground Level Wind Direction (degrees)	Direction of wind at ground level.
	Angle of Terrain (degrees)	Slope at the first point of ground contact during the crash.
	Terrain Characteristic (None/ Uphill/ Downhill)	Direction of terrain from perspective of aircraft heading.
Database	Frontal Area of Aircraft (square feet)	Measurement of the area of aircraft, presented to the airflow.
	Drag Coefficient of Aircraft (C_d)	Aerodynamic forces experienced by aircraft in horizontal and vertical directions.
	Weight of Aircraft (pounds)	Maximum weight of the aircraft.
	Wingspan (feet)	Distance from one wingtip to the other of the aircraft.

Figure 5-2 shows the workflow involved in developing the Python script for the RoCS geoprocessing service, including specifying user and database input, calculating the distance impact line using the CSDRMA script and TAP model, and generating buffers of different debris concentrations. This Python script can also serve as a standalone tool in ArcGIS for Desktop.



Airplane Specifications
Crash Specifications
User Input
Calculate the Line of Impact
TAP Model
CSDRMA
According to Wingspan of Airplane
Create Buffers for Debris Area

Figure 5-2: Workflow of RoCS Geoprocessing Service.

Using the user and database input as specified, the CSDRMA Python script generated an impact distance line along which debris was spread and a single buffer around the line to indicate the possible search area for debris (Janzen, 2012). For more information of the CSDRMA Python script, please refer to Appendix A. However, the client requested an output including multiple rings for varying debris concentrations as shown in Figure 5-3. To address this request, three multiple buffers sharing the same start point were created based on the wingspan of the aircraft. The procedure is as follows.

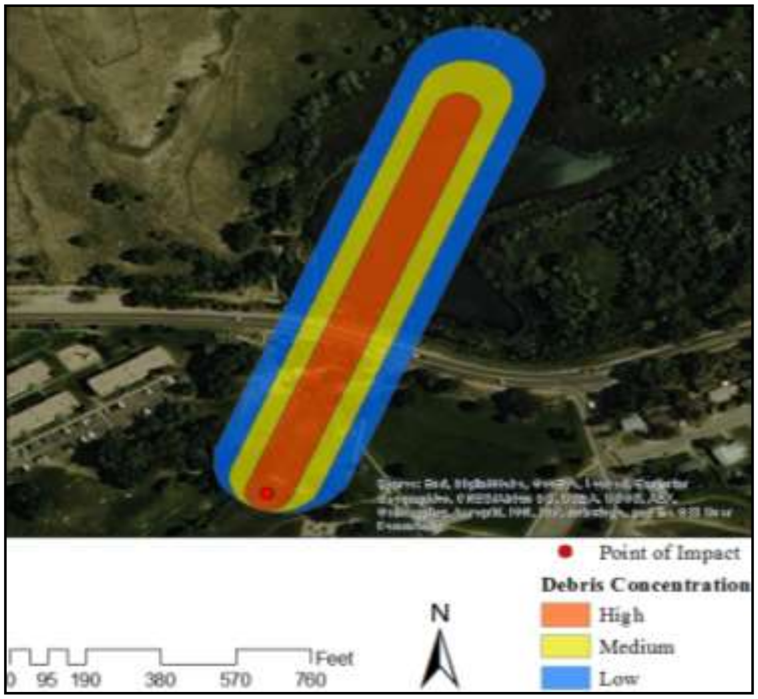


Figure 5-3: Output of RoCS Python Tool.

First, a second line was generated by moving the first line (impact distance line) in the direction of debris area by the distance the same as the wingspan. Using the same process, a third line was generated from the first one with the distance twice the wingspan. These three lines were saved in the *in_memory* feature class with a string field *buffDist*. This field stores wingspan, doubled wingspan ($2 \times wingspan$), and tripled wingspan ($3 \times wingspan$) as the buffer distances required in the next step. To get the area with low, medium, and high concentration of debris, the ArcMap 10.2 Buffer tool was run with the line feature class as input and its *buffDist* field as the buffer distance. Figure 5-3 shows the output generated by the RoCS Python tool. For detailed code, please refer to Appendix B.

Outputs generated by the RoCS Python script like debris area, velocity, time, debris distance, and angle of impact are listed in Table 5-2. Apart from the debris area, all other outputs are non-spatial and are produced as messages of the ArcMap 10.2 Python tool.

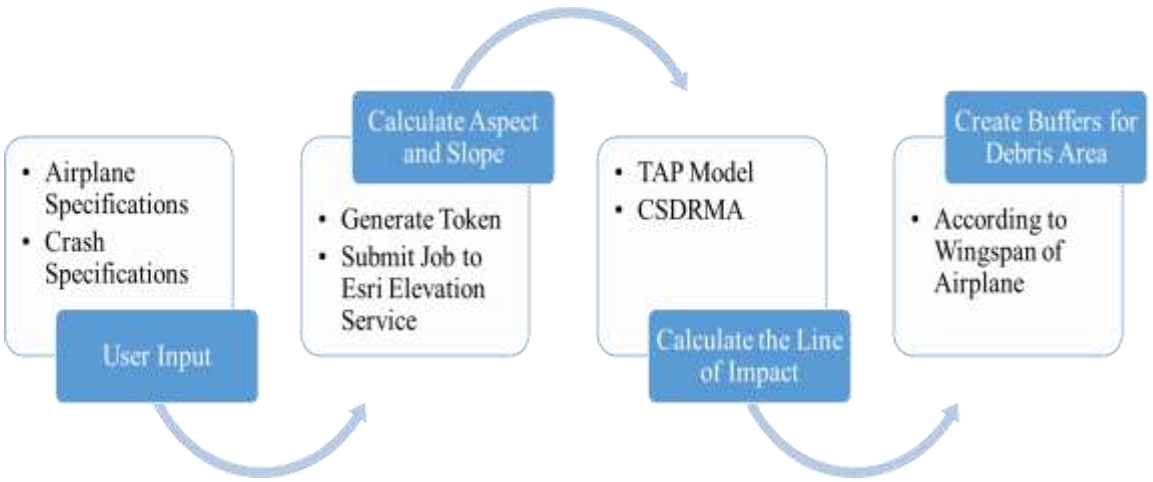
Table 5-2: Output Parameters of RoCS Geoprocessing Service.

Output Parameter	Format	Description
Output Debris Field	Polygon Feature Class	The output feature class comprising of three polygons displaying different debris concentration areas. Location and name of feature class have to be entered while using the service in ArcMap.
Debris Terminal Velocity	knot	Terminal Velocity of debris.
Time of Impact	seconds	Time of debris fall.
Debris Throw Distance	feet	Horizontal distance of expected debris spread.
Angle of Impact	degree	Angle of impact of aircraft.
Speed of Impact	knot	Total speed calculated from horizontal and vertical speeds.
Maximum Altitude of Thrown Debris	feet	Vertical distance that the expected debris flew.

Once the RoCS Python script was completed in ArcGIS for Desktop, it was published as a geoprocessing service on ArcGIS for Server and can be accessed from the URL http://74.208.69.168:26080/arcgis/rest/services/CrashApp/RoCS/GPServer/RoCS.

5.2 Extracting Terrain Property

The RoCS geoprocessing service takes optional parameters: terrain angle and its characteristic (none/uphill/downhill) from the user. This means that if terrain angle and characteristic are not entered, the service will assume that the terrain is flat. This might affect the resultant debris area as discussed in Section 2.2. To handle this drawback, further modifications were made in the RoCS Python tool to use the slope and aspect generated by Esri Summarize Elevation Service in degrees (Esri, 2015d). Because this service is Esri premium content, it cannot be accessed without logging into ArcGIS Online. To use a premium content in Python tool, an access token has to be generated dynamically using *client_id* and *client_secret* of a registered application (Esri, 2015e). This token expires after 120 minutes by default. Since Mojave Desert Ecosystem Program (MDEP) has not adopted Esri’s current model of credits, the extracting terrain characteristics function was not incorporated in the RoCS Python script that was published for the RoCS web application. Instead, a revised Python script was developed to demonstrate that this function could be used in the future when necessary. Figure 5-4 explains the workflow of the revised RoCS Python script. The inputs required for this script are the same as shown in Table 5-1, except for terrain angle and its characteristic, because the terrain aspect and slope were calculated from the service.



Airplane Specifications
Crash Specifications
User Input
Calculate Aspect and Slope
Generate Token
Submit Job to Esri Elevation Service
TAP Model
CSDRMA
Calculate the Line of Impact
Create Buffers for Debris Area
According to Wingspan of Airplane

Figure 5-4: Revised Python Script with Automated Terrain.

The application was registered on ArcGIS Online and an additional function *getToken()* was added in the RoCS Python script to generate a dynamic token using the *client_id* and *client_secret*. A request was sent to the Summarize Elevation Service using the crash point as an input feature and the token. In response, the service sent a job ID, with which slope and aspect at the crash location were determined. These values were converted to terrain angle and none/uphill/downhill using the following procedure.

While using the web application, the user has to select the heading of the aircraft from a drop down menu. It includes the values North, North East, East, South East, South, South West, West, and North West. As described in Figure 5-5, these values were converted to degrees, 0^{o} being north and increasing in clockwise direction.

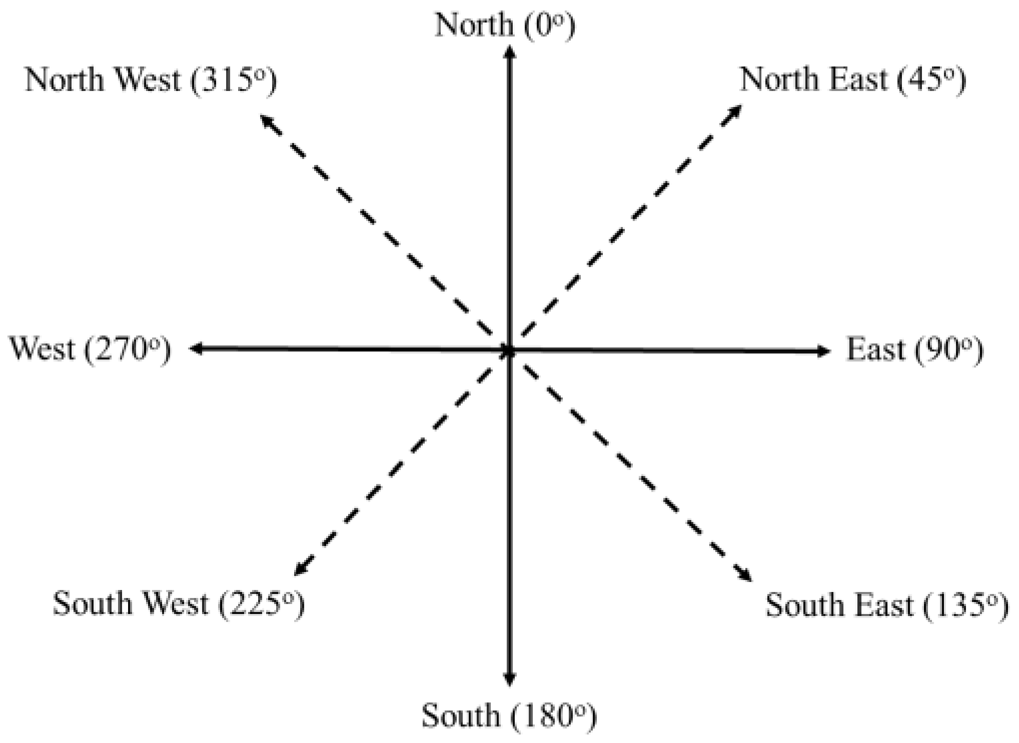


Figure 5-5: Heading of Aircraft

Slope extracted by the Summarize Elevation Service was considered as the terrain angle. The characteristic of terrain (none/uphill/downhill) was determined using the aspect which gives the direction of the terrain. The terrain is flat if the aspect returned by the service is -1. If the aspect of terrain and heading were in opposite directions, as shown in Figure 5-6 (a), the terrain was considered uphill and the absolute difference between aspect and slope was greater than 90^o. However, if the aspect and heading were not in opposite directions as shown in Figure 5-6 (b), the terrain was considered downhill and the absolute difference was less than 90^o.

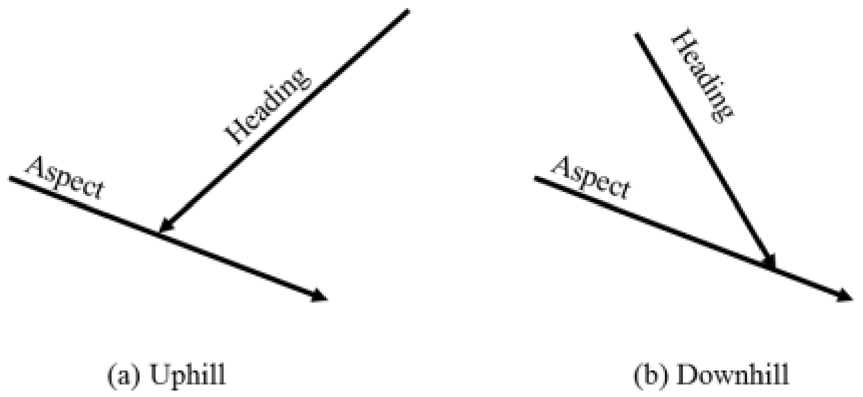


Figure 5-6: Determining Terrain Characteristic (Uphill/Downhill)

The slope and none/uphill/downhill values were used further to calculate the impact angle as mentioned in Section 2.2. For detailed code, refer to Appendix C.

5.3 Developing RoCS Web Application

Since the primary goal of this project was to facilitate the users to access this tool from various platforms, a RoCS web application was developed using HTML, CSS, JavaScript, and PHP. The user interface of the application was designed such that it can be used easily through any mobile device which has an Internet connection, a GPS service, and a browser. Figure 5-7 shows the RoCS web application interface design.

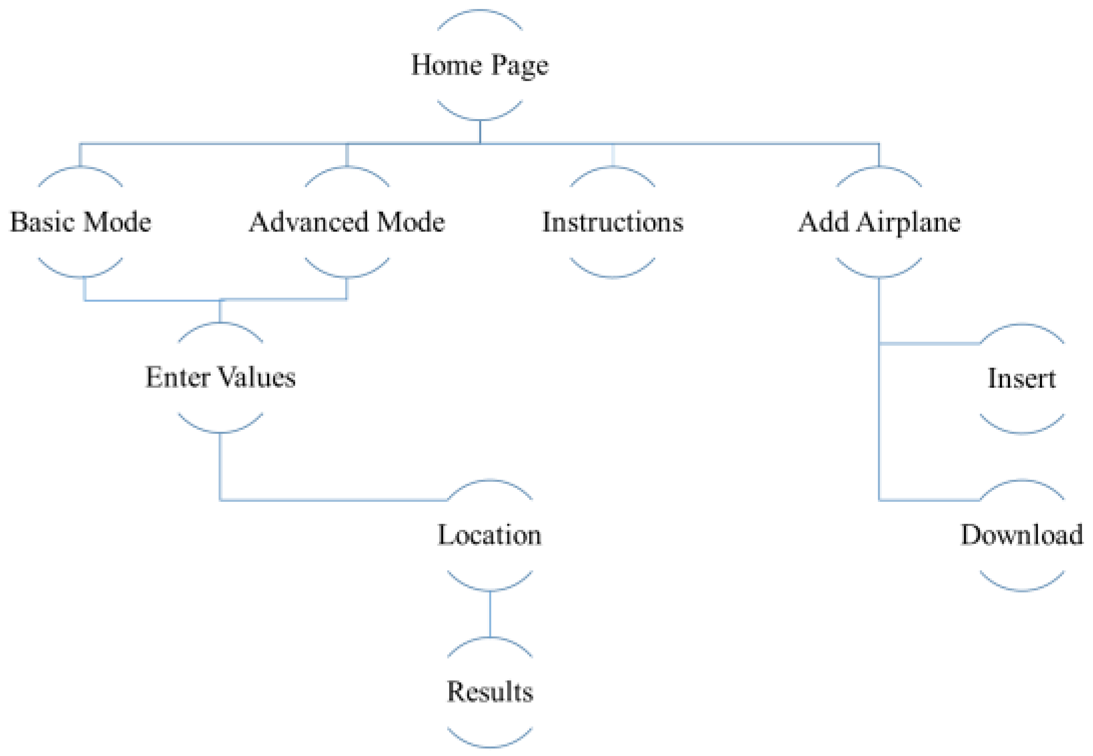


Figure 5-7: RoCS Web Application Interface.

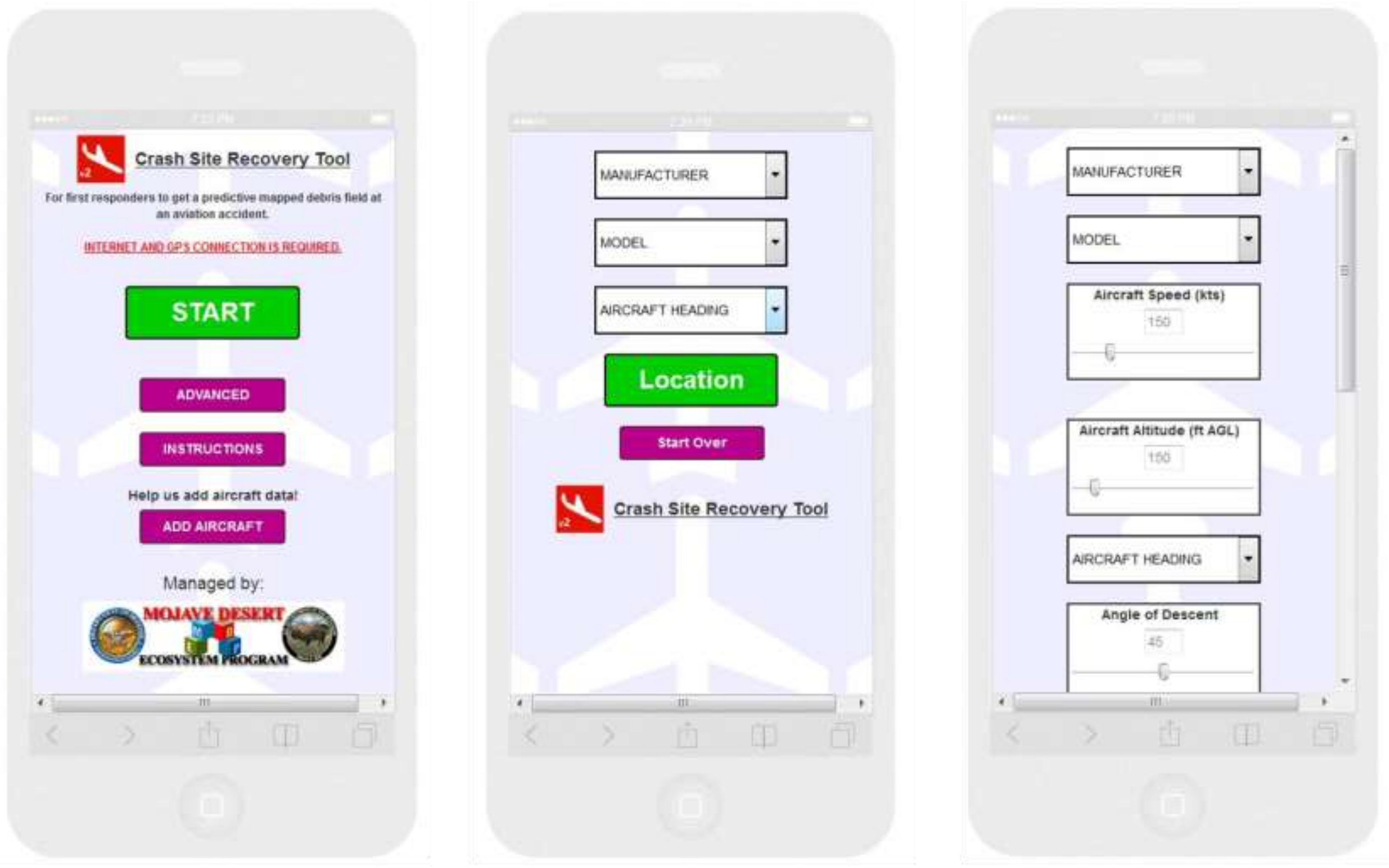
The home page of the application gives four options to the user: basic mode, advanced mode, instructions, and add airplane. Basic mode and advanced mode calculate the debris area based on the parameters entered by the user. Various checks are performed on these parameters to make sure they are valid. For example, it is verified that the aircraft heading is between 0^o and 360^o, and the descent angle of the aircraft and terrain angle is between 0^o and 90^o. Then the web application executes the geoprocessing service by sending the parameters shown in Table 5-1.

The basic mode is for users who do not have all the information required to calculate the debris area. These users only need to enter manufacturer, model, and heading of the aircraft. The remaining parameters will take default values hardcoded based on the input of Fon Duke, Manager at Mojave Desert Ecosystem Program (MDEP). The hardcoded parameters were adopted to be similar to what an aircraft would experience during a landing maneuver, which covered a majority of crash incidents and delivered reasonable results with minimal user input. Table 5-4 shows the default values considered in basic mode which include speed, angle of descent and altitude of aircraft, ground wind speed, ground wind direction, terrain angle, and terrain characteristic. Since the majority of the parameters are hardcoded, it is likely that the results of the basic mode will not be especially accurate. However, it will give a direction for first responders to begin their work.

Table 5-3: Default Parameters of Basic Mode in RoCS Web Application.

Parameter	Unit	Default Value
Speed of Aircraft	Knot	1/4th of maximum speed of aircraft
Angle of Descent	Degree	60
Ground Wind Speed	Knot	0
Ground Wind Direction	degree	0
Terrain Angle	degree	0
Terrain Characteristic	None/Uphill/Downhill	None
Altitude of Aircraft	feet AGL	1000

The advanced mode is for users who know all the required parameters. For this reason, the advanced mode generates results that are more realistic. Figure 5-8 shows the home page, basic mode, and advanced mode of the RoCS web application.


Crash Site Recovery Tool

For first responders to get a predictive mapped debris field at an aviation accident.

INTERNET AND GPS CONNECTION IS REQUIRED.

Help us add aircraft data!

Managed by:

(a) Home Page (b) Basic Mode (c) Advanced Mode

Figure 5-8: Screen Shots of RoCS Web Application

After clicking the *Location* button on the basic or advanced page, the user is sent to a map with current location zoomed in. The user can pan, zoom-in, and zoom-out to find and select the crash location. Figure 5-9 shows the location page in the RoCS web application. The red dot on the screen means the location has been captured by the application. If the user clicks on Finish without selecting the location, a blank map is displayed in the results page.

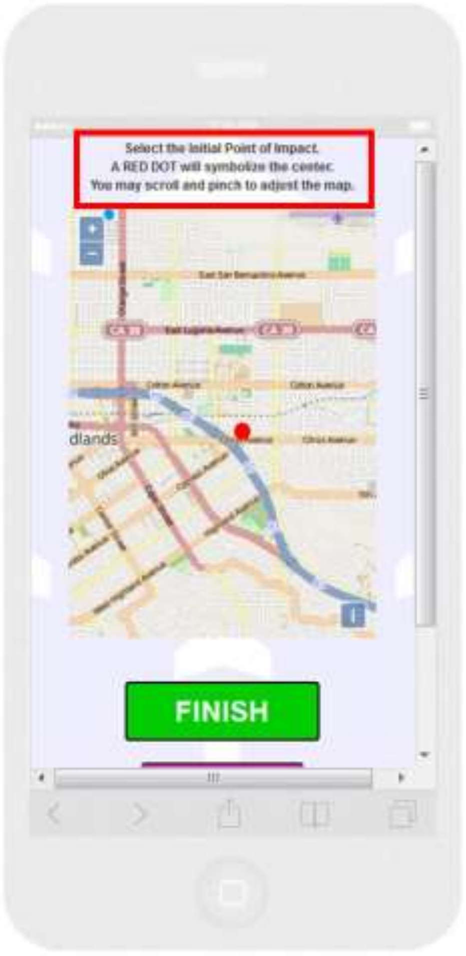

Select the Initial Point of Impact.
A RED DOT will symbolize the center.
You may scroll and pinch to adjust the map.
FINISH

Figure 5-9: Location page of RoCS

Once the location of a crash is selected, the *Finish* button sends all parameters to the RoCS geoprocessing service and takes the user to the results page where the debris area is displayed. Figure 5-10 shows the results page of the application where the high, medium, and low concentration areas are displayed on a basemap, which can be panned, zoomed in, or zoomed out.

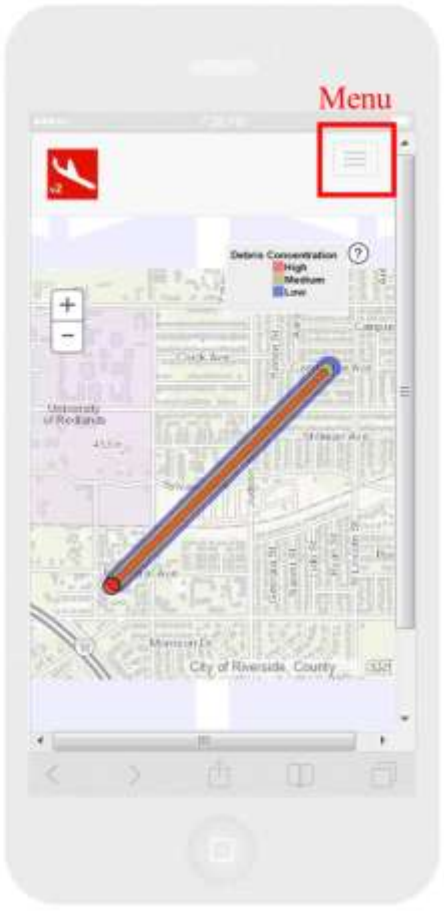


Figure 5-10: Results page of RoCS

A menu is provided on this page for the user to toggle basemap, save the map as a JPG image, email a link to the map image, or toggle the US National Grid as a layer on the map. Figure 5-11 shows a screen shot of the menu on the results page.

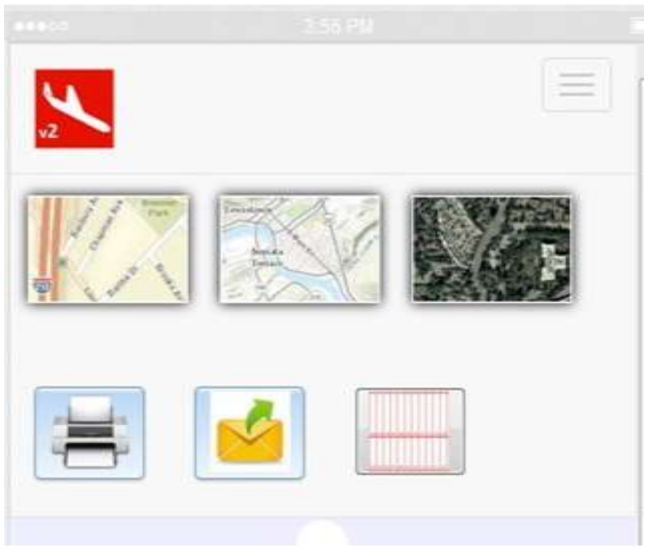


Figure 5-11: Menu of RoCS

The instructions page is a systematic guide for how to use the web application. Finally, Figure 5-12 shows the Add Airplane page that lets the user add a new airplane to the database or download the data of existing airplanes.

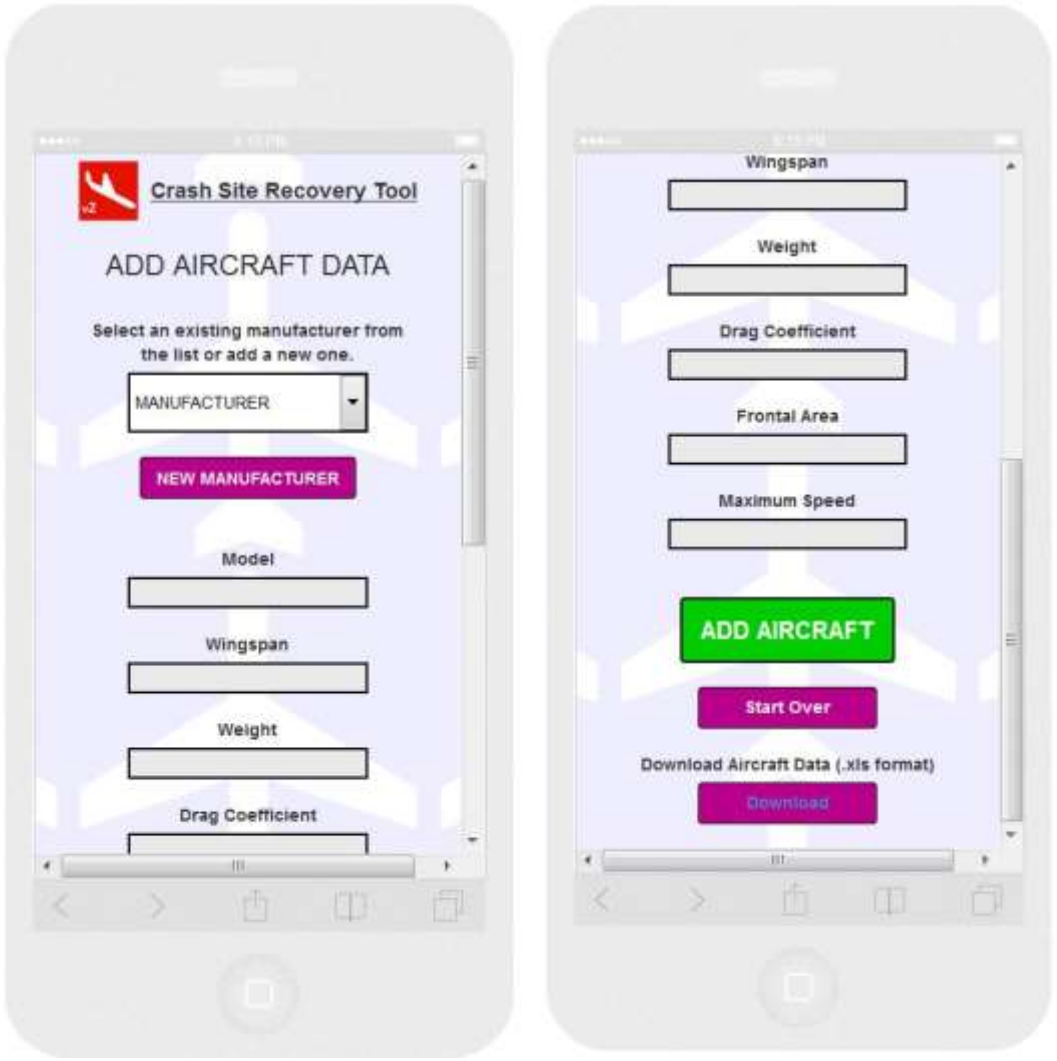


Figure 5-12: Add/Download Aircraft page of RoCS

The web application is published on the Mojave Desert Ecosystem Program (MDEP) server and is available for public use. It can be accessed from URL http://crash.mojavedata.gov.

5.4 Summary

A Python script tool was developed using Python and the ArcMap 10.2 toolbox. This tool was published on ArcGIS for Server as a geoprocessing service and used in the web application to generate the debris area. The web application was designed using HTML, CSS, PHP, and JavaScript so that it can be accessed from any mobile device. PHP ensured secure connection between the application and the aircraft database. The database also acts as an initiative to crowdsource details of various aircrafts.

The geoprocessing service considers the terrain as flat if the user does not enter terrain characteristics. Therefore, a revised Python script tool was developed to consider the terrain parameters from the Esri elevation service instead of values entered by users.

However, the revised tool was not used in the web application because each time the elevation service is executed, it consumes credits from the associated ArcGIS Online account, which is not desired by the client.

Chapter 6 – Results and Analysis

This chapter discusses the results obtained from the Recovery of Crash Site (RoCS) geoprocessing service and the revised Python tool. Section 6.1 explains the effect of input parameters on the estimated debris field. Section 6.2 compares the actual debris area with the area generated by the RoCS geoprocessing tools. Model accuracy and efficiency measures were developed and compared among four test cases. The impact of terrain on debris estimation is discussed in Section 6.3 with a focus on the revised Python tool. Section 6.4 explains the various tests conducted to validate the web application.

6.1 Effects of Parameters on the Estimated Debris Field

Various parameters related to the airplane, crash, and terrain are given as inputs to the debris model. This section discusses the effect of each parameter on the estimated debris field. Table 5-1 indicate whether the respective parameter is entered by the user or extracted from the airplane database.

Speed of aircraft at the time of impact is required in knots (kts). One knot is a unit of speed equivalent to 1.151 miles per hour. Speed of aircraft is directly proportional to the resultant debris field. If all the other parameters are considered constant, increase in speed increases the debris area, whereas if speed is reduced, debris area decreases.

Altitude of aircraft before the initial impact is required in feet Above Ground Level (ft AGL). As the altitude is increased, the estimated debris field increases as well.

Direction of the aircraft at the time of initial impact is called the heading. It is required in degrees, 0^{o} being North and increasing in a clockwise direction (Figure 5-5). The heading of the aircraft does not affect the debris throw distance, but changes the direction in which it is thrown.

Descent is the angle between the horizontal ground and the path of the aircraft at the point of initial impact. The descent angle must be between 0^{o} and 90^{o}. Assuming all the other parameters are constant, increase in descent angle reduces the debris throw distance, and vice versa.

Frontal area is the measurement of the area presented to the airflow in square feet (Oldham, 1990). Frontal area of aircraft and the estimated debris field are inversely proportional. Therefore, with increase in frontal area, the debris field reduces.

Drag coefficient (C_d) refers to the horizontal and vertical aerodynamic force experienced by the aircraft. As the coefficient increases, the estimated debris field decreases because of the force experienced by the aircraft.

There are multiple weight values associated with an aircraft, such as maximum weight while takeoff, maximum weight while landing, weight of empty aircraft, weight of aircraft with fuel, and more. Based on the client's recommendation, the maximum weight of aircraft during takeoff was considered for the adapted debris model. The debris throw distance is correlated positively with the weight of the aircraft.

Wingspan is the length from the tip of one wing to the tip of the other wing, in feet. Wingspan does not change the debris throw distance, but it affects the estimated spread area of the debris. An increase in wingspan will result in an increase in the width of the debris throw area.

Ground level is the elevation of initial impact point relative to mean sea level. It is measured in feet Mean Sea Level (ft MSL).

Ground level wind speed is considered in knots and wind direction is measured in degrees, with 0^o being North (Figure 5-5). If the direction of wind is the same as the aircraft heading, the debris throw distance increases. However, the distance is reduced if the direction of wind is against the heading of aircraft.

Terrain angle must be between 0^o and 90^o. If terrain is uphill, an increase in terrain angle reduces the debris throw distance. However if terrain is downhill, the debris throw distance increases. Refer to Section 2.2 for the effect of terrain on debris area.

6.2 Model Validation

Adapted from the Crash Site Debris Recovery Mobile Application (CSDRMA) (Janzen, 2012), the model implemented by the RoCS geoprocessing service predicts the possible debris fields from an airplane crash by considering speed, altitude, aircraft heading, descent, frontal area, drag coefficient, weight, wingspan of the aircraft, ground level, ground level wind speed, ground level wind direction, terrain angle, and terrain characteristic. To evaluate model performance, the predicted debris fields need to be compared to actual debris fields. One approach is to compare the percentage of the area shared by both the types of fields. However, the actual debris fields were not very accurate as not all debris were recorded. To represent the better geographic extent of actual debris fields, a convex hull of the actual debris fields may be used for comparison. Both approaches were implemented in this study with a focus on the latter.

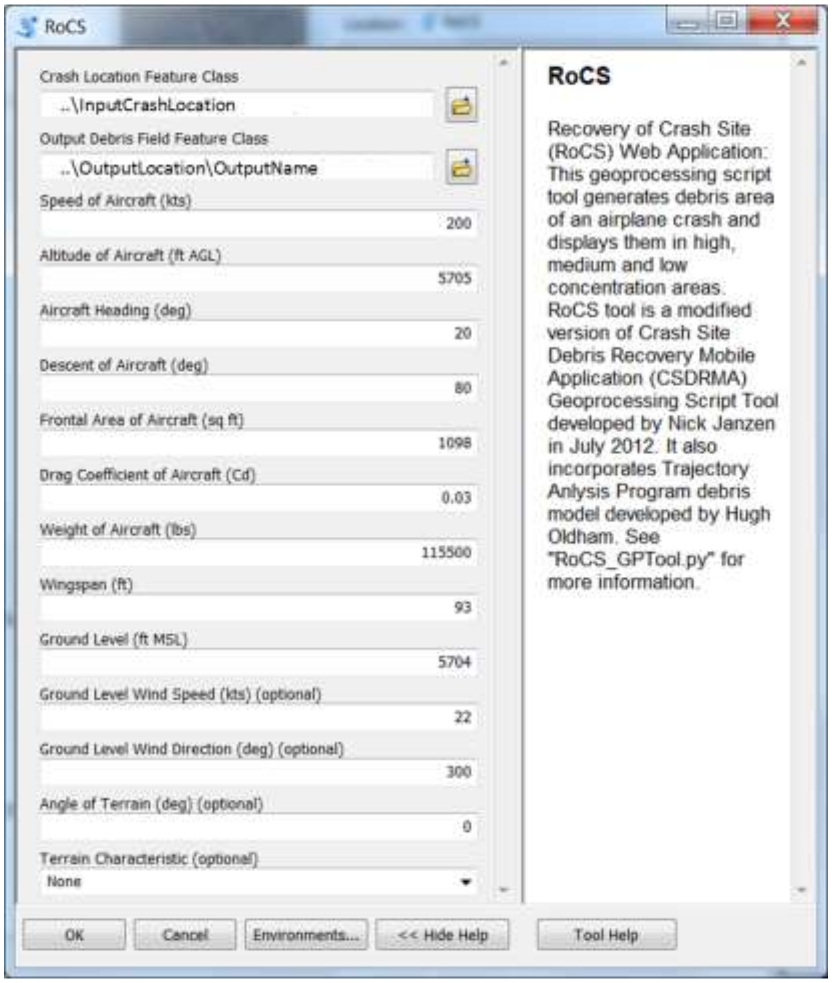
Model performance was evaluated from two perspectives: model accuracy and model efficiency. The model accuracy measure assesses how accurately the model output is when compared to the ground truth. This measure is calculated as the percentage of area in common between the convex hull of the actual debris field and the RoCS generated debris field. The model performs well if most of the actual debris fields fall into the predicted area. The model efficiency measure evaluates how much the model over-predicts the debris area. An accurate model output might not be efficient if a large portion of the predicted debris area does not contain any actual debris observations. The measure is calculated as the percentage of the RoCS generated debris area that overlaps with the actual debris field. The model is more efficient when there is less over-prediction.

Four test datasets described in Section 4.3 were used to assess model accuracy and efficiency. Table 6-1 summarizes the input parameters for the four test cases including the 1991 Colorado crash, the 2008 London crash, the 2011 New Mexico crash, and the 2013 Alabama crash. The terrains were assumed flat in this stage as no specific terrain parameters were provided.

Table 6–1: Input for Test Cases

	Colorado Crash, 1991	Alabama Crash, 2013	London Crash, 2008	New Mexico Crash, 2011
Speed of Aircraft (kts)	200	120	106	145
Altitude of Aircraft (ft AGL)	5,705	1,000	200	1,000
Aircraft Heading (degree)	20	175	270	270
Descent of Aircraft (degree)	80	60	45	60
Frontal Area of Aircraft (sq ft)	1,098	2,799	525	500
Drag Coefficient	0.03	0.03	0.03	0.02
Weight of Aircraft (lbs)	115,500	308,650	460,000	99,600
Wingspan (ft)	93	147.11	199.92	99.6
Ground Level (ft MSL)	5,704	650	77	999
Ground Level Wind Speed (kts)	22	0	0	0
Ground Level Wind Direction (degree)	300	0	0	0
Angle of Terrain (degree)	0	0	0	0
Terrain Characteristic (None/Uphill/Downhill)	None	None	None	None

Figure 6-1 shows an example of how to use the RoCS tool in ArcGIS 10.2 to calculate the predicted debris areas for the 1991 Colorado crash. First, the point of contact of airplane during the crash was given as an input. Location and name of the output feature class was then specified. All the remaining values such as speed, altitude, heading, descent, frontal area, drag coefficient, weight, and wingspan of aircraft, ground level, ground level wind speed, ground level wind direction, and angle of terrain with upslope or downslope were entered based on the specifications in Table 6-1.


RoCS

Crash Location Feature Class
..\InputCrashLocation

Output Debris Field Feature Class
..\OutputLocation\OutputName

Speed of Aircraft (kts)
200

Altitude of Aircraft (ft AGL)
5705

Aircraft Heading (deg)
20

Descent of Aircraft (deg)
80

Frontal Area of Aircraft (sq ft)
1098

Drag Coefficient of Aircraft (Cd)
0.03

Weight of Aircraft (lbs)
115500

Wingspan (ft)
93

Ground Level (ft MSL)
5704

Ground Level Wind Speed (kts) (optional)
22

Ground Level Wind Direction (deg) (optional)
300

Angle of Terrain (deg) (optional)
0

Terrain Characteristic (optional)
None

RoCS

Recovery of Crash Site (RoCS) Web Application: This geoprocessing script tool generates debris area of an airplane crash and displays them in high, medium and low concentration areas. RoCS tool is a modified version of Crash Site Debris Recovery Mobile Application (CSDRMA) Geoprocessing Script Tool developed by Nick Janzen in July 2012. It also incorporates Trajectory Anlysis Program debris model developed by Hugh Oldham. See "RoCS_GPTool.py" for more information.

Figure 6-1: RoCS Geoprocessing Tool

Once the tool ran, the debris area was calculated and displayed on the map in blue, yellow, and orange, showing the low, medium, and high debris concentrations respectively (Figure 6-2).

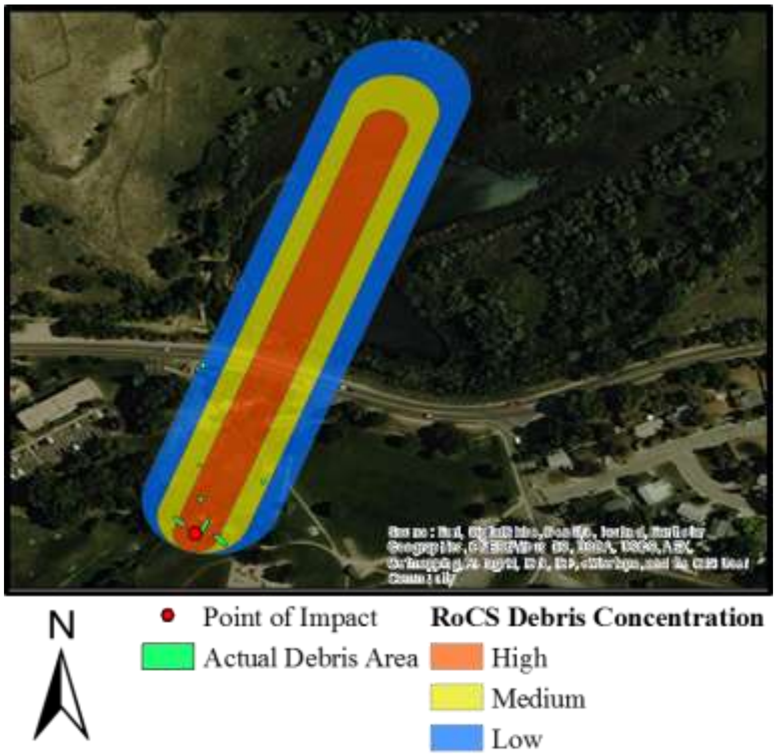
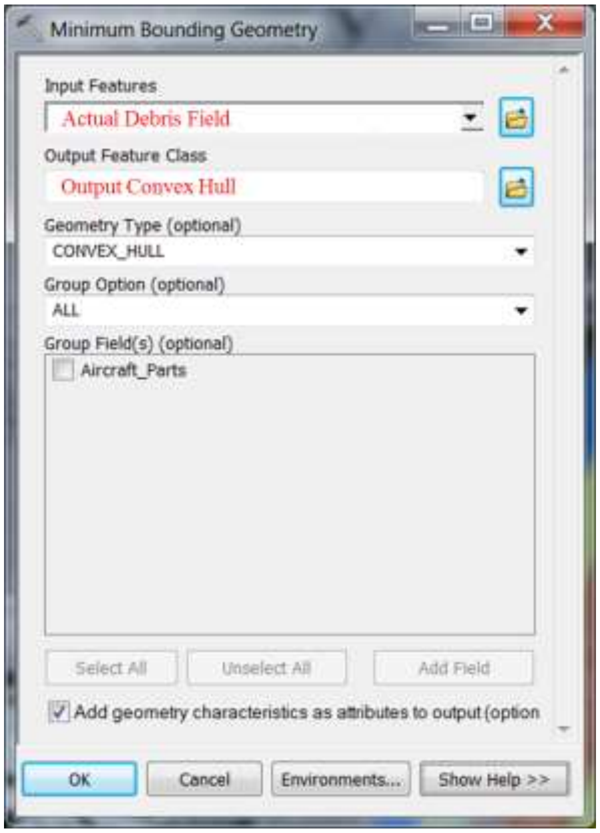


Figure 6-2: RoCS Generated Debris Area

A convex hull of the actual debris fields was constructed to represent the extent of debris. Figure 6-3 shows the *Minimum Bounding Geometry* tool used to generate the convex hull of the actual debris fields, where the feature class of the actual debris field was given as the input, and geometry type was entered as CONVEX_HULL.


Minimum Bounding Geometry

Input Features: Actual Debris Field

Output Feature Class: Output Convex Hull

Geometry Type (optional): CONVEX_HULL

Group Option (optional): ALL

Group Field(s) (optional): Aircraft_Parts

[✓] Add geometry characteristics as attributes to output (option

Figure 6-3: Minimum Boundary Geometry tool (ArcMap 10.2)

The convex hull of the actual debris fields of the Colorado Crash is shown in Figure 6-4 (a). Almost all of actual debris fields fall into the predicted debris area, while there is still a large portion of the predicted area not containing any debris observations. Using the same procedure, the map outputs of the other three test cases were developed (Figure 6-4 b, c, and d). It appears that the model over-predicts debris fields for both Colorado and Alabama Crash, but under-predicts debris field for New Mexico Crash. The estimation of the London Crash seems to fall in between.

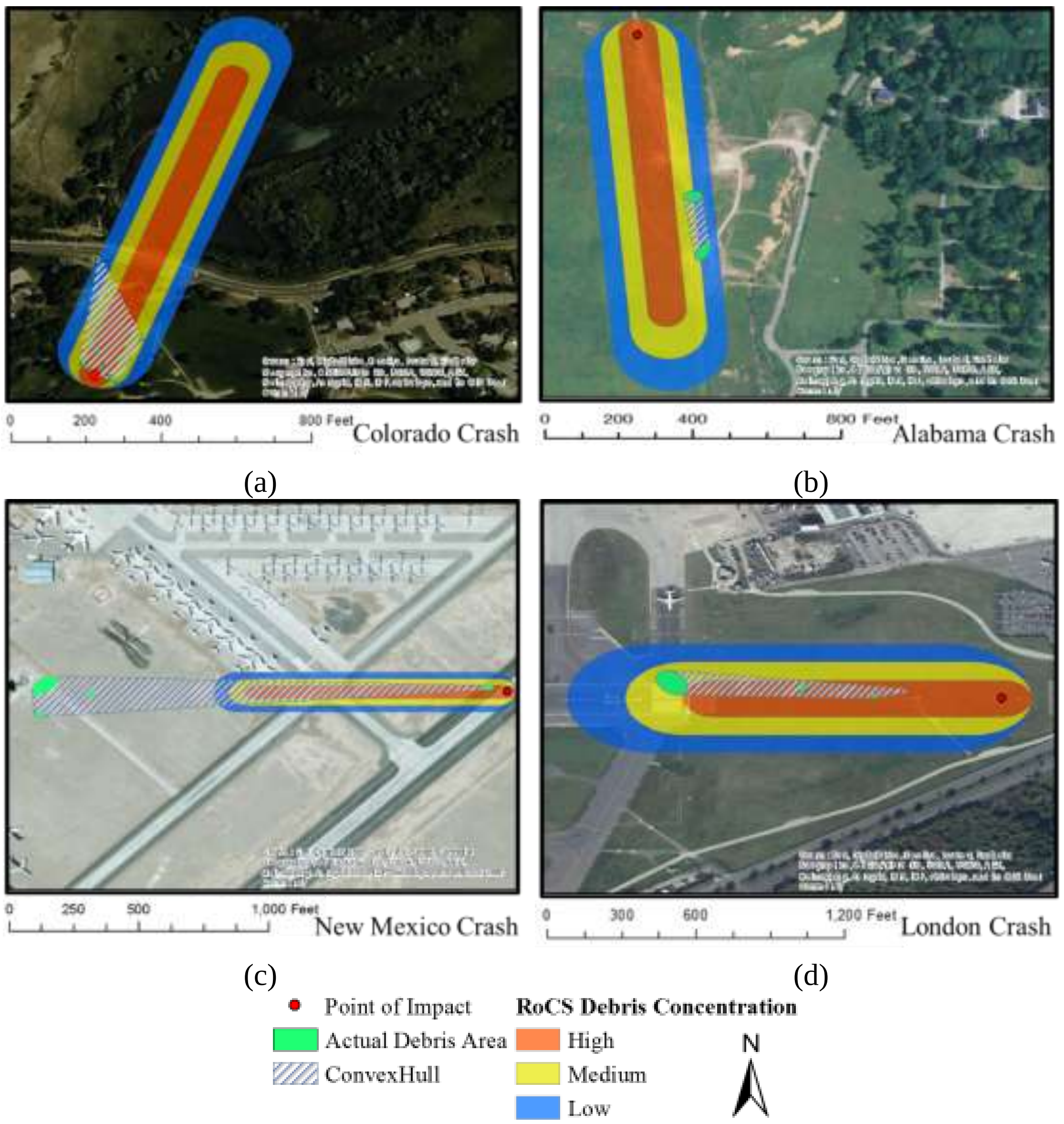


Figure 6-4: Comparison of Predicted and Actual Debris Fields

To quantify the comparison, the aforementioned two measures, accuracy and efficiency measures, were calculated using a few common GIS operations, such as intersection and field calculation. The accuracy measure, the percentage of the convex hull overlapping the predicted areas, was calculated as follows:

$$Accuracy\ Measure = \frac{Intersect\ Area}{Convex\ Hull\ Area} \times 100$$

The efficiency measure, the percentage of RoCS area overlapping the convex hull, was calculated as follows:

$$Efficiency\ Measure = \frac{Intersect\ Area}{RoCS\ Debris\ Field\ Area} \times 100$$

Table 6-2 presents the two measures for all four test cases. For the Colorado, Alabama, and London crashes, the predicted areas cover most of the convex hulls of the actual debris fields. This indicates that the model accurately predicts the debris fields. However, the efficiency of the model is rather low because at most 13% of the predicted area contains any debris observations. In contrast, accuracy and efficiency of the model are comparable for the New Mexico Crash. About 40% of the actual debris field is covered by the predicted area and around 40% of the predicted area contains debris observation. These four test cases suggest that the debris trajectory model has a reasonable accuracy but poor efficiency.

Table 6–2: Model Accuracy and Efficiency Measures

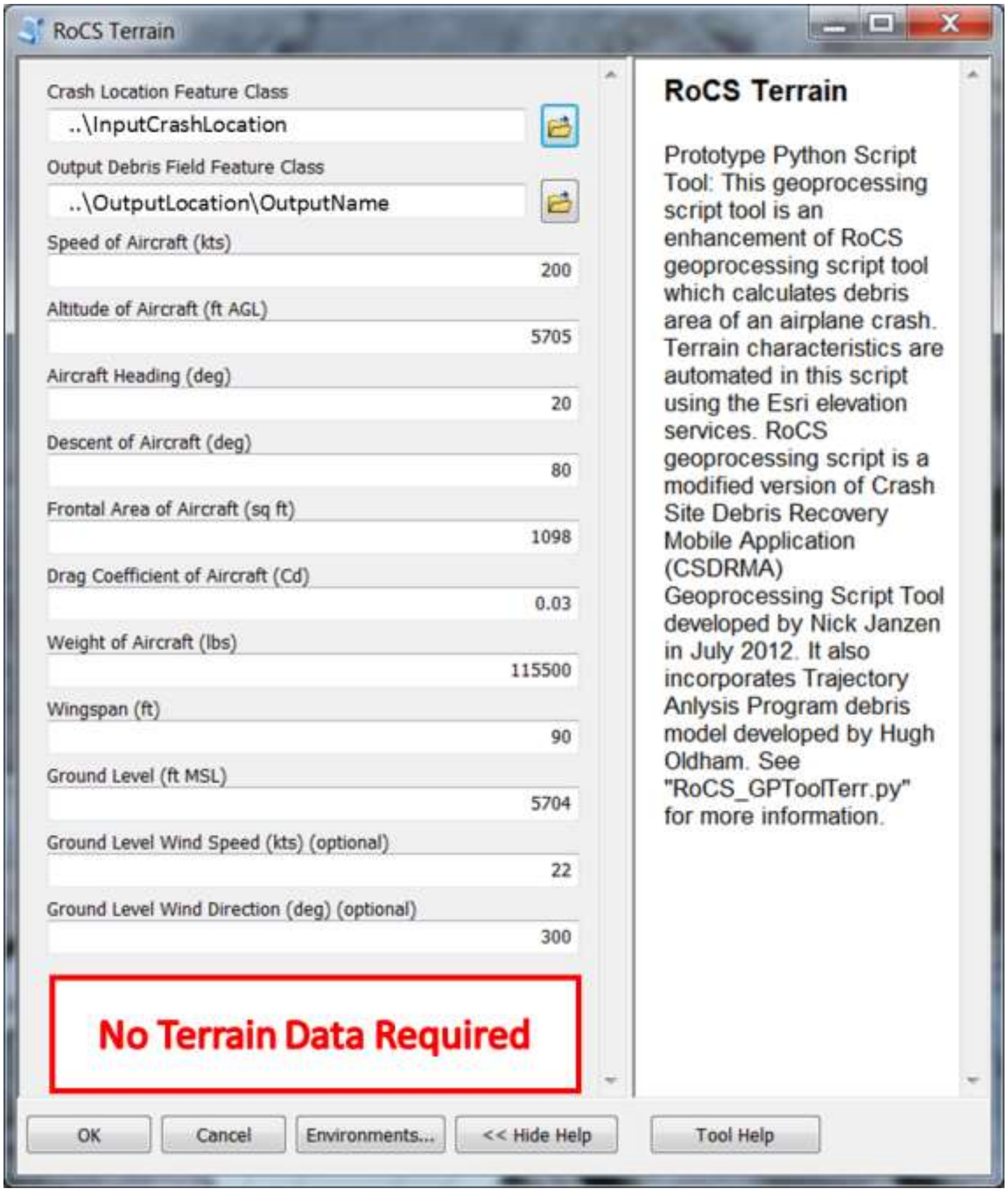
Crash	Accuracy Measure (%)	Efficiency Measure (%)
Colorado Crash	100.00	12.93
Alabama Crash	100.00	2.87
London Crash	100.00	9.03
New Mexico Crash	41.18	36.66

As mentioned in the beginning of this section, accuracy can also be measured using the area of actual debris field instead of its convex hull. This approach reduces the accuracy of the model to a great extent; for example, the accuracy of the New Mexico crash changed from 41% down to about 10% when the area of actual debris fields was considered. This is because the convex hull represents the geographic spread of the actual debris fields, not the actual area where debris was found.

6.3 Terrain Impact on Debris Prediction

It would be desirable if the tool can automatically extract terrain characteristics when terrain parameters are not available. This function was implemented in a revised Python tool (Section 5.2), but was not published due to the ArcGIS Online credit concern.

Compared to the RoCS geoprocessing tool, the input parameters required for this tool are the same except the terrain parameters (Figure 6-5). In the RoCS tool, if the user does not enter terrain parameters, the terrain is considered to be flat. The revised tool overcomes this drawback by extracting the terrain parameters from the Esri Summarize Elevation Service (Esri, 2015d). Note that the slope extracted from the Esri Elevation Service only refers to the slope at the point of impact, which does not consider slope variations along the distance of throw. Not being able to accommodate the terrain characteristics along the trajectories of debris remains one of the limitations of the current debris model.


RoCS Terrain

Crash Location Feature Class
..\InputCrashLocation

Output Debris Field Feature Class
..\OutputLocation\OutputName

Speed of Aircraft (kts)
200

Altitude of Aircraft (ft AGL)
5705

Aircraft Heading (deg)
20

Descent of Aircraft (deg)
80

Frontal Area of Aircraft (sq ft)
1098

Drag Coefficient of Aircraft (Cd)
0.03

Weight of Aircraft (lbs)
115500

Wingspan (ft)
90

Ground Level (ft MSL)
5704

Ground Level Wind Speed (kts) (optional)
22

Ground Level Wind Direction (deg) (optional)
300

No Terrain Data Required

RoCS Terrain

Prototype Python Script Tool: This geoprocessing script tool is an enhancement of RoCS geoprocessing script tool which calculates debris area of an airplane crash. Terrain characteristics are automated in this script using the Esri elevation services. RoCS geoprocessing script is a modified version of Crash Site Debris Recovery Mobile Application (CSDRMA) Geoprocessing Script Tool developed by Nick Janzen in July 2012. It also incorporates Trajectory Anlysis Program debris model developed by Hugh Oldham. See "RoCS_GPToolTerr.py" for more information.

Figure 6-5: Revised Python Tool

Using this tool, the debris fields of the four test aircraft crashes were re-calculated and the outputs were compared to the predicted areas generated under the assumption of flat terrain (Figure 6-5).

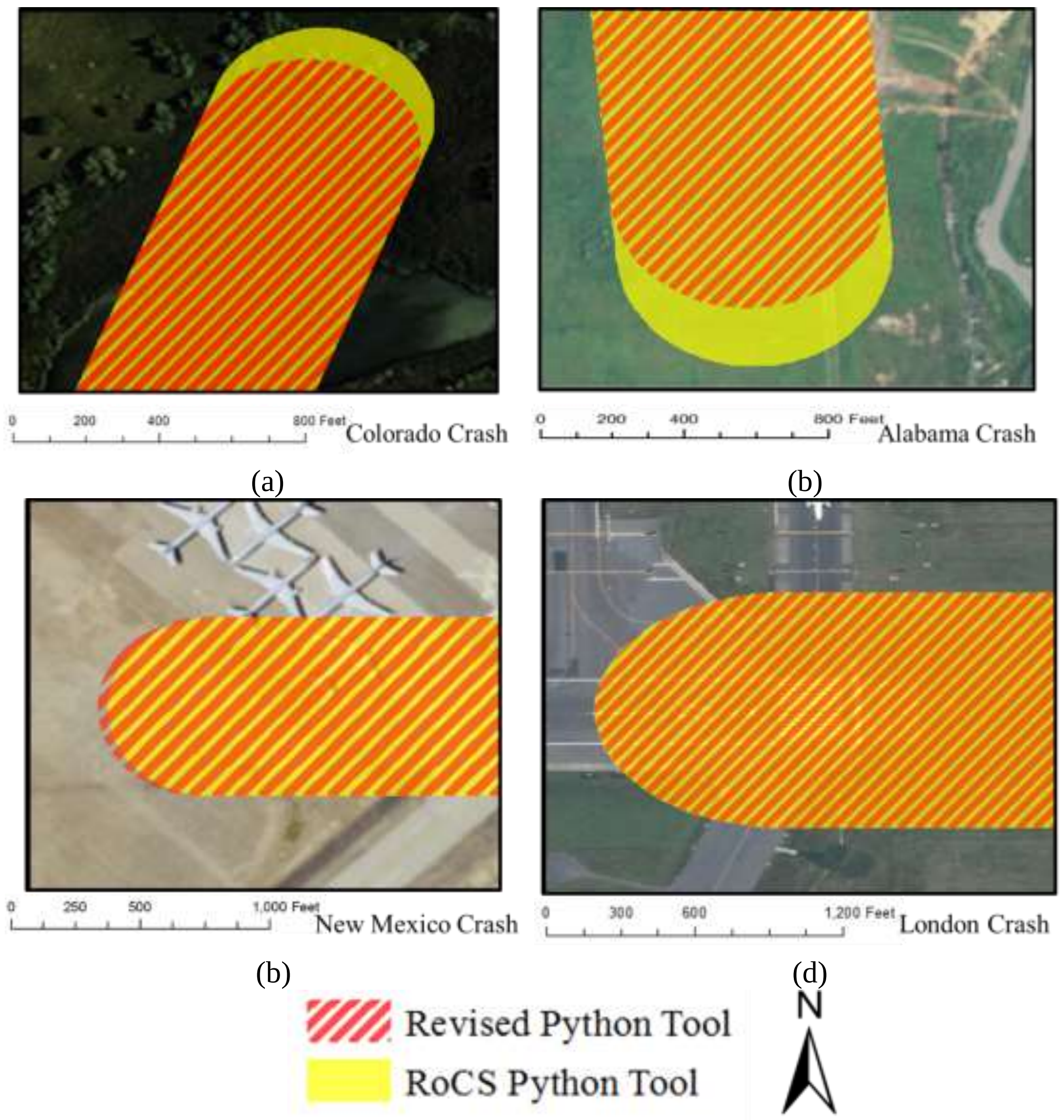


Figure 6-6: Difference in Outputs of the RoCS and the Revised Python Tool

When the tool was executed for the four datasets, the slope was calculated as 0.44^o uphill for the Colorado Crash, 3.66^o uphill for the Alabama Crash, 0.41^o downhill for the New Mexico Crash, and 0.44^o uphill for the London Crash. Since the calculated slope of the Alabama Crash is the largest as compared to the other three terrain settings, the generated debris area differs the most as compared to the RoCS generated area as shown in Figure 6-6 (b). In addition, the uphill terrains reduced the distance of throw (Colorado and Alabama crashes) while the downhill terrain enlarged the distance of throw (New Mexico crash). The London and Colorado crashes have similar terrain characteristics, however the other parameters, such as the angle of descent and the type of airplane, were different, which resulted in dissimilar decreases in debris fields.

6.4 Web Application Testing

Since the main goal of the project was to increase the target audience, the RoCS web application was developed in HTML, CSS, JavaScript, and PHP. To make sure that the application was platform independent, features like button click, drop down load, pan of map, zoom-in of map, zoom-out of map, print as JPG, email, toggle of basemap, adding aircraft, and saving airplane data as Excel were tested on different operating systems and devices. These operating systems included iOS 8.3, Blackberry 10.3.1.2576, Android 5.0.2, and Android 4.2.2 JDQ39. In addition, Mojave Desert Ecosystem Program (MDEP) tested the application with Android 4.0, Android 4.4, and iOS 6. The RoCS web application was tested on devices iPhone, iPad, iPod, Motorola, Samsung, and LG. The same features were tested successfully on Internet Explorer 11, Mozilla Firefox 38.0.5, Google Chrome 39, Opera 30, and Safari 5.1.7.

To test the user interface of the web application, the client conducted a test survey to make sure that all the features are easy to locate and use. The users involved in this survey were professional first responders and investigators. Based on the feedback received in the test survey, modifications were made to the user interface.

6.5 Summary

Fon Duke, Manager of the Mojave Desert Ecosystem Program (MDEP), provided the data required for testing. The data were originally in World Geodetic System (WGS) 1984 and was projected to WGS 1984 World Mercator to calculate the area. The percentage of actual debris fields covered by the RoCS generated field was around 81%, but the percentage of RoCS debris field that matched the actual debris field was much less. Therefore, the tool provides a direction for the first responders and the investigators, but they cannot rely on it completely. More parameters can be incorporated in this model to make it more accurate.

The web application was accessible from different devices with varying operating systems. As a result, the target audience is expected to increase significantly.

Chapter 7 – Conclusions and Future Work

The objectives of this project were to develop a database for airplanes, modify the existing geoprocessing script, and develop a web application that can be used by any device which has an Internet connection, a GPS service, and a compatible browser. Therefore, a database was developed in Microsoft SQL 2008 R2, the Python script was modified and published on ArcGIS for Server, and a platform independent web application was developed. The application was tested successfully on devices with different operating systems. In addition, a revised Python script was developed to include terrain parameters more efficiently. The database, the geoprocessing service, and the web application were published on the client's server and are available for public use.

7.1 Future Work

Although many parameters were considered in the debris model implemented in this project, the model itself could be improved in the future by adding other parameters, such as land cover.

A revised Python script was developed to improve results of the effect of terrain on the calculated debris area. Since this script uses ArcGIS Online credits (Esri, 2015d), it was not merged with the delivered solution. This script could be modified to calculate the terrain parameters without utilizing the credits. This could be done by using or developing an open source tool that extracts terrain slope and aspect of a point, line, or polygon feature from a Digital Elevation Model (DEM).

To validate the debris model, a convex hull was created for the actual debris area data of various crashes provided by the Mojave Data Ecosystem Program (MDEP). These convex hulls indicate a fan-shaped debris area rather than the oval debris area considered for this project. Therefore, modifications can be made in the RoCS geoprocessing service to generate fan-shaped debris areas.

The estimated debris area is shown on a topographic basemap with an option to toggle the basemap between Esri Imagery without labels, Esri Topographic, or Esri Streets. This estimated area is in the form of three 2-dimensional ovals displaying high, medium, and low concentrations of debris. Using the upcoming version of ArcGIS API for JavaScript, this output can be modified to 3-dimensional space to provide better visualization.

The model of the Recovery of Crash Site (RoCS) web application could be modified so that it can work in an offline mode. This would help the first responders, cleanup crews, and investigators use the application even if an internet connection is unavailable.

The geoprocessing service uses various ArcGIS 10.2 tools and is published on ArcGIS for Server. The script can be modified to use QGIS tools and can be published on GeoServer to make the application 100% open source.

The web application can be modified to include editing features, which can be used by the first responders and the investigators to edit the actual debris field in real time. In addition, more layers such as police stations, hospitals, fire stations, and airports can be added to the results page with toggle functionality.

7.2 Summary

In summary, the Recovery of Crash Site (RoCS) web application was successful as it will assist first responders, cleanup crews, and investigators to initiate their work at an airplane crash site. It can be accessed from any device which has an internet connection, a GPS, and any compatible browser. The application results in an estimated debris field area with high, medium, and low concentrations that can be emailed or saved as a JPG file. In addition, a database is used as an initiative to crowdsource airplane information.

Works Cited

aerospaceweb.org. (2011, May 30). *Aircraft Museum*. Retrieved from http://www.aerospaceweb.org/aircraft/

City of Rancho Cucamonga, California. (n.d.). *GIS Maps and Data*. Retrieved June 9, 2015, from City of Rancho Cucamonga, California: http://www.cityofrc.us/cityhall/admin/gis/mapsdata/default.asp

Coltman, J., Ingen, C., Johnson, N., & Zimmerman, R. (1989). *Aircraft Crash Survival Design Guide. Volume 2. Aircraft Design Crash Impact Conditions and Human Tolerance.* Phoenix, Arizona: SIMULA INC. Retrieved from http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA218435

Department for Transport (Air Accidents Investigation Branch). (2010). *Report on the accident to Boeing 777-236ER, G-YMMM, at London Heathrow Airport on 17 January 2008* (Report EW/C2008/01/01). United Kingdom.

Esri. (2015a). *GIS Supports Gov 2.0*. Retrieved June 9, 2015, from http://www.esri.com/industries/gov20

Esri. (2015b). *GIS for Business*. Retrieved June 9, 2015, from http://www.esri.com/industries/business

Esri. (2015c). *GIS Education Research*. Retrieved June 9, 2015, from http://edcommunity.esri.com/educational-roles/Researchers

Esri. (2015d). *ArcGIS REST API: Elevation Analysis Services*. Retrieved June 16, 2015, from https://developers.arcgis.com/rest/elevation/api-reference/summarize-elevation.htm

Esri. (2015e). *Accessing ArcGIS Online Services*. Retrieved June 19, 2015, from https://developers.arcgis.com/authentication/accessing-arcgis-online-services/#getting-a-token

Federal Aviation Administration. (2009, December 11). *Aircraft Characteristics Database.* Retrieved from http://www.faa.gov/airports/engineering/aircraft_char_database/media/aircraft_char_122009.xls

Fu, P., & Sun, J. (2011). *Web GIS, Principles and Applications.* Redlands, California: ESRI Press.

Janzen, N. R. (2012, July). *Crash Site Debris Recovery Mobile Application (Master's thesis, University of Redlands).* Retrieved from http://inspire.redlands.edu/gis_gradproj

Lin, M. Y., Larson, E. W., & Collins, J. D. (2003, June). *Determination of Debris Risk to the Public Due to the Columbia Breakup During Reentry*. Retrieved from Accident Investigation Board: http://caib1.nasa.gov/news/report/pdf/vol2/part16.pdf

Mojave Desert Ecosystem Program. (n.d.). *About*. Retrieved November 2014, from https://www.mojavedata.gov/about.html

National Transportation Safety Board. (2001). *Uncontrolled Descent and Collision With Terrain, United Airlines Flight 585, Boeing 737-200, N999UA, 4 Miles South of Colorado Springs, Municipal Airport, Colorado Springs, Colorado, March 3, 1991* (Aircraft Accident Report NTSB/AAR-01/01). Washington D.C.: National Transportation Safety Board.

National Transportation Safety Board. (2012). *Crash During Experimental Test Flight, Gulfstream Aerospace Corporation GVI (G650), N652GD, Roswell, New Mexico, April 2, 2011* (Aircraft Accident Report NTSB/AAR-12/02). Washington D.C.: National Transportation Safety Board.

National Transportation Safety Board. (2014). *Crash During a Nighttime Nonprecision Instrument Approach to Landing, UPS Flight 1354, Airbus A300-600, N155UP, Birmingham, Alabama, August 14, 2013* (Aircraft Accident Report NTSB/AAR-14/02). Washington D. C.: National Transportation Safety Board.

Oldham, H. (1990). *Aircraft debris trajectory analysis.* Retrieved from Proairshow, LLC: http://proairshow.com/aircraft_debris.htm

Palt, K. (2015, June 28). *Aircraft Encyclopedia*. Retrieved from http://www.flugzeuginfo.net/acdata_en.php

Point Blue Conservation Science. (2014). *Whale Aware*. Retrieved June 9, 2015, from Download Whale mAPP: http://www.whaleaware.org/index.php?page=download-whale-mapp-coming-soon

Sarconi, M. (2013). *A Prototype System for Simulating the Risks of Sub-Orbital Space Flight for Commercial Aviation.* Retrieved from University of Glasgow: http://www.dcs.gla.ac.uk/~pat/4yProjects/HallOfFame/reports2013/SarconiMarco.pdf

Söylemez, E., & Usul, N. (2006). *Utility Of GIS In Search And Rescue Operations*. Retrieved November 2014, from http://proceedings.esri.com/library/userconf/proc06/papers/papers/pap_1908.pdf

U.S. Bank. (2015). *U.S. Bank Locations*. Retrieved June 9, 2015, from https://www.usbank.com/locations/

Appendix A. Crash Site Debris Recovery Mobile Application (CSDRMA) Python Script

CSDRMA Python script that was developed by Nicholas Janzen in 2012.

```
#Crash Site Debris Recovery Mobile Application (CSDRMA) Geoprocessing Script Tool
#Trajectory Analysis for Aircraft Debris
#   for use as a tool script in ArcGIS 10.0
#   (requires feature class input of crash location)
#Nick Janzen
#MS GIS Program
#University of Redlands
#Redlands, California, USA
#July 2012
#Based on the "TAPS" program for GW-BASIC, Hugh Oldham, the
#  "Thrown Rotor Blade Trajectory Analysis" for Microsoft
#  Excel, T Watson (http://proairshow.com/aircraft_debris.htm),
#  and excerpts from the book "Aircraft Accident Investigation -
#  2nd Edition", Richard Wood & Robert Sweginnis

#Import Modules
import math, sys, arcpy, os

#ArcGIS Desktop Input
inputcrash = arcpy.GetParameterAsText(0)
workspace = arcpy.GetParameterAsText(1)
VEL = float (arcpy.GetParameterAsText(2))
ALT = float (arcpy.GetParameterAsText(3))
Course = float (arcpy.GetParameterAsText(4))
ANGA = float (arcpy.GetParameterAsText(5))
FAREA = float (arcpy.GetParameterAsText(6))
CD = float (arcpy.GetParameterAsText(7))
WT = float (arcpy.GetParameterAsText(8))
wingspan = float (arcpy.GetParameterAsText(9))
Groundlevel = float (arcpy.GetParameterAsText(10))
SWIND = float (arcpy.GetParameterAsText(11))
DWIND = float (arcpy.GetParameterAsText(12))
TANGA = float (arcpy.GetParameterAsText(13))
SLOPE = arcpy.GetParameterAsText(14)

#Input Checks

#Course Check I
if Course >= 1 and Course <= 360:
    Course = Course
```

```
else:
    arcpy.AddError("Error: Flight Path Course input must be 1-360")
    sys.exit()

#ANGA Check
if ANGA >= 0 and ANGA <= 90:
    ANGA = ANGA
else:
    arcpy.AddError("Error: Descent of Aircraft input must be 0-90")
    sys.exit()

#Course Check II
if Course < DWIND:
    WINDC = (Course - DWIND) * (math.pi/180) #convert to radians
    WINDC = math.cos(WINDC) * SWIND
else:
    WINDC = (DWIND-Course) * (math.pi/180) #convert to radians
    WINDC = math.cos(WINDC) * SWIND

#TANGA Check
if TANGA >= 0 and TANGA <=90:
    TANGA = TANGA
else:
    arcpy.AddError("Error: Angle of Terrain for Aircraft-Ground Impact input must be 1-
90")
    sys.exit()

#SLOPE Check
if SLOPE == "Upslope":
    ANGA = ANGA + TANGA
elif SLOPE == "Downslope":
    ANGA = ANGA - TANGA
elif SLOPE == "None":
    ANGA = ANGA
else:
    arcpy.AddError("Error: Aircraft-Ground Impact Characteristic input must be 'None',
'Upslope' or 'Downslope'")
    sys.exit()

#Preliminary Data Processing
wingspan = wingspan/2
CDS = CD * FAREA
SLUGS = 0.002378 * (1-(6.875**-6*ALT))**4.2561
GSLUGS = 0.002378 * (1-(6.875**-6*Groundlevel))**4.2561
TVEL = (2*WT/(CDS*SLUGS))**0.5
TVELKTS = TVEL * 0.5921052
```

```
GLTVEL = (2*WT/(CDS*GSLUGS))**0.5
GLTVELKTS = GLTVEL * 0.5921052
DT = 0.05
WINDone = abs(WINDC * 6080/3600)
T = 0
X = 0
Z = ALT
DTtwo = DT * DT
UVprep = ANGA * (math.pi/180) #convert to radians
U = 1.69 * VEL * math.cos(UVprep)
V = 1.69 * VEL * math.sin(UVprep)
W = WINDone * (Z/30) ** 0.26
UO = U
VO = V
Zmax = 0
FPEANG = 0

#Data Processing
while Z > Groundlevel:
   VELtwo = U*U+V*V
   if U == 0:
      U=.01
   else:
      U = U
   FP = math.atan(V/U)
   if U<0 and V<0:
      K = -1
   else:
      K = 1
   DRAG = (SLUGS/2)*VELtwo*CDS
   AX = -DRAG * math.cos(FP) * 32.2 * K/WT
   AZ = -DRAG * math.sin(FP) * 32.2/WT - 32.2
   UO = UO + AX * DT
   V = V + AZ * DT
   U = UO + W
   VO = V
   FPE = math.atan(VO/UO)
   X = X + UO * DT + 0.5 * AX * DTtwo
   Z = Z + VO * DT + 0.5 * AZ * DTtwo
   if Z > Groundlevel:
      T = T + DT
   else:
      T = T
   if abs(Z)>abs(Zmax):
      Zmax = Z  ,m * (180/math.pi) #convert to degrees
   else:
    Zmax = Zmax

#Output Processing
impactkts = (((UO*UO+VO*VO)**0.5)*0.68182)
FPE = FPE * (180/math.pi) #convert to degrees

#Spatial Processing
WKID = 4326 #WGS84
sr = arcpy.SpatialReference()
sr.factoryCode = WKID
arcpy.env.outputCoordinateSystem = sr
arcpy.env.workspace = "in_memory"
arcpy.CreateFeatureclass_management ("in_memory", "Bearing")
arcpy.MakeTableView_management ("Bearing","in_memory/BearingTable")
arcpy.AddField_management ("in_memory/BearingTable", "x_lon", "DOUBLE")
arcpy.AddField_management ("in_memory/BearingTable", "y_lat", "DOUBLE")
arcpy.AddField_management ("in_memory/BearingTable", "length", "DOUBLE")
arcpy.AddField_management ("in_memory/BearingTable", "bearing", "DOUBLE")
cur = arcpy.InsertCursor("in_memory/BearingTable")
row = cur.newRow()
arcpy.AddXY_management(inputcrash)
icrows = arcpy.SearchCursor(inputcrash)
for icrow in icrows:
  xlon = icrow.getValue("POINT_X")
  ylat = icrow.getValue("POINT_Y")
row.x_lon = xlon
row.y_lat = ylat
row.length = X
row.bearing = Course
cur.insertRow(row)
del cur, row
arcpy.BearingDistanceToLine_management("in_memory/BearingTable", "BearingLine",
"x_lon", "y_lat", "length","FEET", "bearing","", "","", sr)

#Spatial Outputs
buffer = arcpy.Buffer_analysis("BearingLine", workspace, str(wingspan) + " feet",
"FULL", "ROUND", "NONE")

#Output
arcpy.AddMessage("Debris Terminal Velocity (kts): " + str(TVELKTS))
arcpy.AddMessage("Time to Impact (sec): " + str(T))
arcpy.AddMessage("Debris Throw Distance (ft): " + str(X))
arcpy.AddMessage("Angle of Impact (deg): " + str(FPE))
arcpy.AddMessage("Speed of Impact (kts): " + str(impactkts))
arcpy.AddMessage("Max Altitude of Debris (ft): " + str(Zmax))
arcpy.AddMessage("Analysis Complete")
```

Appendix B. Recovery of Crash Site (RoCS) Python Script

RoCS Python script that is based on Crash Site Debris Recovery Mobile application (CSDRMA) by Nicholas Janzen in 2012 and Trajectory Analysis Program (TAP) by Oldham in 1990.

```
#Recovery of Crash Site (RoCS) Web Application Geoprocessing Script Tool
#Shilpi Jain
#MS GIS Program
#University of Redlands
#Redlands, California, USA
#August 2015
#Based on the Crash Site Debris Recovery Mobile Application (CSDRMA)
Geoprocessing Script Tool
#   Trajectory Analysis for Aircraft Debris
#   for use as a tool script in ArcGIS 10.0, Nick Janzen, July 2012
#   (https://www.mojavedata.gov/data_crash/MIP_NickJanzen.pdf), and
#Trajectory Analysis Program for GW-BASIC, Hugh Oldham,
#   (http://proairshow.com/aircraft_debris.htm)

#Import Modules
import math, sys, arcpy, os

#ArcGIS Desktop Input
inputcrash = arcpy.GetParameterAsText(0)
workspace = arcpy.GetParameterAsText(1)
VEL = float (arcpy.GetParameterAsText(2))
ALT = float (arcpy.GetParameterAsText(3))
Course = float (arcpy.GetParameterAsText(4))
ANGA = float (arcpy.GetParameterAsText(5))
FAREA = float (arcpy.GetParameterAsText(6))
CD = float (arcpy.GetParameterAsText(7))
WT = float (arcpy.GetParameterAsText(8))
wingspan = float (arcpy.GetParameterAsText(9))
Groundlevel = float (arcpy.GetParameterAsText(10))
SWIND = float (arcpy.GetParameterAsText(11))
DWIND = float (arcpy.GetParameterAsText(12))
TANGA = float (arcpy.GetParameterAsText(13))
SLOPE = arcpy.GetParameterAsText(14)

#Input Checks

#Course Check I
if Course >= 0 and Course <= 360:
    Course = Course
```

```
else:
    arcpy.AddError("Error: Flight Path Course input must be 0-360")
    sys.exit()

#ANGA Check
if ANGA >= 0 and ANGA <= 90:
    ANGA = ANGA
else:
    arcpy.AddError("Error: Descent of Aircraft input must be 0-90")
    sys.exit()

#Course Check II
if Course < DWIND:
    WINDC = (Course - DWIND) * (math.pi/180) #convert to radians
    WINDC = math.cos(WINDC) * SWIND
else:
    WINDC = (DWIND-Course) * (math.pi/180) #convert to radians
    WINDC = math.cos(WINDC) * SWIND

#TANGA Check
if TANGA >= 0 and TANGA <=90:
    TANGA = TANGA
else:
    arcpy.AddError("Error: Angle of Terrain for Aircraft-Ground Impact input must be 0-
90")
    sys.exit()

#SLOPE Check
if SLOPE == "Upslope":
    ANGA = ANGA + TANGA
elif SLOPE == "Downslope":
    ANGA = ANGA - TANGA
elif SLOPE == "None":
    ANGA = ANGA
else:
    arcpy.AddError("Error: Aircraft-Ground Impact Characteristic input must be 'None',
'Upslope' or 'Downslope'")
    sys.exit()

#Preliminary Data Processing
wingspan = wingspan/2
CDS = CD * FAREA
SLUGS = 0.002378 * (1-(6.875**-6*ALT))**4.2561
GSLUGS = 0.002378 * (1-(6.875**-6*Groundlevel))**4.2561
TVEL = (2*WT/(CDS*SLUGS))**0.5
TVELKTS = TVEL * 0.5921052
```

```
GLTVEL = (2*WT/(CDS*GSLUGS))**0.5
GLTVELKTS = GLTVEL * 0.5921052
DT = 0.05
WINDone = abs(WINDC * 6080/3600)
T = 0
X = 0
Z = ALT
DTtwo = DT * DT
UVprep = ANGA * (math.pi/180) #convert to radians
U = 1.69 * VEL * math.cos(UVprep)
V = 1.69 * VEL * math.sin(UVprep)
W = WINDone * (Z/30) ** 0.26
UO = U
VO = V
Zmax = 0
FPEANG = 0

#Data Processing
while Z > Groundlevel:
   VELtwo = U*U+V*V
   if U == 0:
      U=.01
   else:
      U = U
   FP = math.atan(V/U)
   if U<0 and V<0:
      K = -1
   else:
      K = 1
   DRAG = (SLUGS/2)*VELtwo*CDS
   AX = -DRAG * math.cos(FP) * 32.2 * K/WT
   AZ = -DRAG * math.sin(FP) * 32.2/WT - 32.2
   UO = UO + AX * DT
   V = V + AZ * DT
   U = UO + W
   VO = V
   FPE = math.atan(VO/UO)
   X = X + UO * DT + 0.5 * AX * DTtwo
   Z = Z + VO * DT + 0.5 * AZ * DTtwo
   if Z > Groundlevel:
      T = T + DT
   else:
      T = T
   if abs(Z)>abs(Zmax):
      Zmax = Z * (180/math.pi) #convert to degrees
   else:
        Zmax = Zmax

#Output Processing
impactkts = (((UO*UO+VO*VO)**0.5)*0.68182)
FPE = FPE * (180/math.pi) #convert to degrees

#Spatial Processing
WKID = 4326 #WGS84
sr = arcpy.SpatialReference()
sr.factoryCode = WKID
arcpy.env.outputCoordinateSystem = sr
arcpy.env.workspace = "in_memory"
arcpy.CreateFeatureclass_management ("in_memory", "Bearing")
arcpy.DeleteRows_management("in_memory/Bearing")
arcpy.MakeTableView_management ("in_memory/Bearing","in_memory/BearingTable")
arcpy.AddField_management ("in_memory/BearingTable", "x_lon", "DOUBLE")
arcpy.AddField_management ("in_memory/BearingTable", "y_lat", "DOUBLE")
arcpy.AddField_management ("in_memory/BearingTable", "length", "DOUBLE")
arcpy.AddField_management ("in_memory/BearingTable", "bearing", "DOUBLE")
arcpy.AddField_management ("in_memory/BearingTable", "buffDist", "TEXT")

arcpy.AddXY_management(inputcrash)
icrows = arcpy.SearchCursor(inputcrash)

for icrow in icrows:
    xlon = icrow.getValue("POINT_X")
    ylat = icrow.getValue("POINT_Y")


#Shilpi Start
R = 6378.1   #Radius of Earth
d = wingspan * 0.0003048 #distance km
d2 = 2 * d

lat1 = math.radians(ylat)
lon1 = math.radians(xlon)

#Second Line
lat2 = math.asin( math.sin(lat1)*math.cos(d/R) +
     math.cos(lat1)*math.sin(d/R)*math.cos(math.radians(Course)))

lon2 = lon1 + math.atan2(math.sin(math.radians(Course))*math.sin(d/R)*math.cos(lat1),
             math.cos(d/R)-math.sin(lat1)*math.sin(lat2))

#Third Line
lat3 = math.asin( math.sin(lat2)*math.cos(d/R) +
    math.cos(lat2)*math.sin(d/R)*math.cos(math.radians(Course)))

lon3 = lon2 + math.atan2(math.sin(math.radians(Course))*math.sin(d/R)*math.cos(lat2),
            math.cos(d/R)-math.sin(lat2)*math.sin(lat3))

#Converting to degrees
lat2 = math.degrees(lat2)
lon2 = math.degrees(lon2)
lat3 = math.degrees(lat3)
lon3 = math.degrees(lon3)

#Updating buffer distance
cur = arcpy.InsertCursor("in_memory/BearingTable")
row = cur.newRow()

row.x_lon = lon3
row.y_lat = lat3
row.length = X
row.bearing = Course
row.buffDist = str(3 * wingspan) + " Feet"
cur.insertRow(row)

row.x_lon = lon2
row.y_lat = lat2
row.length = X
row.bearing = Course
row.buffDist = str(2 * wingspan) + " Feet"
cur.insertRow(row)

row.x_lon = xlon
row.y_lat = ylat
row.length = X
row.bearing = Course
row.buffDist = str(wingspan) + " Feet"
cur.insertRow(row)
del cur, row
#Shilpi End

arcpy.BearingDistanceToLine_management("in_memory/BearingTable",
"in_memory/BearingLine", "x_lon", "y_lat", "length","FEET", "bearing","", "","", sr)

#Shilpi Start
arcpy.AddField_management ("in_memory/BearingLine", "buffDist", "TEXT")
typeCursor = arcpy.da.UpdateCursor("in_memory/BearingLine", ["buffDist"])
cnt = 0
for row1 in typeCursor:
    if cnt == 0:
        row1[0] = str(3 * wingspan) + " Feet"
        cnt += 1
    elif cnt == 1:
        row1[0] = str(2 * wingspan) + " Feet"
        cnt += 1
    else:
        row1[0] = str(wingspan) + " Feet"
    typeCursor.updateRow(row1)
#Shilpi End

#Spatial Outputs
buffer = arcpy.Buffer_analysis("in_memory/BearingLine", workspace, "buffDist",
"FULL", "ROUND", "NONE")

#Output
arcpy.AddMessage("Debris Terminal Velocity (kts): " + str(TVELKTS))
arcpy.AddMessage("Time to Impact (sec): " + str(T))
arcpy.AddMessage("Debris Throw Distance (ft): " + str(X))
arcpy.AddMessage("Angle of Impact (deg): " + str(FPE))
arcpy.AddMessage("Speed of Impact (kts): " + str(impactkts))
arcpy.AddMessage("Max Altitude of Debris (ft): " + str(Zmax))
arcpy.AddMessage("Analysis Complete")
```

Appendix C. Automated Terrain Python Script

Modified RoCS Python script, used to extract terrain slope and aspect from the Summarize Elevation Service.

```
#Recovery of Crash Site (RoCS) web application Geoprocessing Script Tool

#Shilpi Jain
#MS GIS Program
#University of Redlands
#Redlands, California, USA
#August 2015
#Based on the Trajectory Analysis Program for GW-BASIC, Hugh Oldham, the
#  "Thrown Rotor Blade Trajectory Analysis" for Microsoft
#  Excel, T Watson (http://proairshow.com/aircraft_debris.htm),
#Crash Site Debris Recovery Mobile Application (CSDRMA) Geoprocessing Script Tool
#Trajectory Analysis for Aircraft Debris
#   for use as a tool script in ArcGIS 10.0, Nick Janzen, July 2012
# (https://www.mojavedata.gov/data_crash/MIP_NickJanzen.pdf)

#Import Modules
import math, sys, arcpy, os, urllib, urllib2, json, requests, time

def main():
    #ArcGIS Desktop Input
    inputcrash = arcpy.GetParameterAsText(0)
    workspace = arcpy.GetParameterAsText(1)
    VEL = float (arcpy.GetParameterAsText(2))
    ALT = float (arcpy.GetParameterAsText(3))
    Course = float (arcpy.GetParameterAsText(4))
    ANGA = float (arcpy.GetParameterAsText(5))
    FAREA = float (arcpy.GetParameterAsText(6))
    CD = float (arcpy.GetParameterAsText(7))
    WT = float (arcpy.GetParameterAsText(8))
    wingspan = float (arcpy.GetParameterAsText(9))
    Groundlevel = float (arcpy.GetParameterAsText(10))
    SWIND = float (arcpy.GetParameterAsText(11))
    DWIND = float (arcpy.GetParameterAsText(12))
    #Input Checks
    #Course Check I
    if Course >= 0 and Course <= 360:
        Course = Course
    else:
        arcpy.AddError("Error: Flight Path Course input must be 0-360")
        sys.exit()
    #ANGA Check
    if ANGA >= 0 and ANGA <= 90:
        ANGA = ANGA
    else:
        arcpy.AddError("Error: Descent of Aircraft input must be 0-90")
        sys.exit()

    #Course Check II
    if Course < DWIND:
        WINDC = (Course - DWIND) * (math.pi/180) #convert to radians
        WINDC = math.cos(WINDC) * SWIND
    else:
        WINDC = (DWIND-Course) * (math.pi/180) #convert to radians
        WINDC = math.cos(WINDC) * SWIND

    #Shill new start
    fc = arcpy.AddXY_management(inputcrash)
    token = getToken()
    print token
    arcpy.AddMessage("Tooken = " + token)
    ip = arcpy.FeatureSet()
    ip.load(fc)

    data =
requests.post('http://elevation.arcgis.com/arcgis/rest/services/Tools/Elevation/GPServer/S
ummarizeElevation/submitJob', params={
      'f': 'json',
      'token': token,
      'InputFeatures': ip.JSON,
      'IncludeSlopeAspect': 'true'
    })

    jsonData = data.json()
    print(jsonData)
    arcpy.AddMessage(jsonData)

    time.sleep(5)

    OP =
requests.post('http://elevation.arcgis.com/arcgis/rest/services/Tools/Elevation/GPServer/S
ummarizeElevation/jobs/'+ jsonData["jobId"] +'/results/OutputSummary?token='+ token
+'&f=json')
    opJason = OP.json()
    TANGA = opJason["value"]["features"][0]["attributes"]["MeanSlope"]
    print(TANGA)
    arcpy.AddMessage("Mean Slope = " + str(TANGA))
```

```
Aspect = opJason["value"]["features"][0]["attributes"]["MeanAspect"]
arcpy.AddMessage("Mean Aspect = " + str(Aspect))

if (Aspect == -1):                 #Flat
   ANGA = ANGA
elif ((Aspect - Course) <= 90):    #DownHill
   ANGA = ANGA - TANGA
elif ((Aspect - Course) > 90):     #UpHill
   ANGA = ANGA + TANGA
#Shill new end


#Preliminary Data Processing
wingspan = wingspan/2
CDS = CD * FAREA
SLUGS = 0.002378 * (1-(6.875**-6*ALT))**4.2561
GSLUGS = 0.002378 * (1-(6.875**-6*Groundlevel))**4.2561
TVEL = (2*WT/(CDS*SLUGS))**0.5
TVELKTS = TVEL * 0.5921052
GLTVEL = (2*WT/(CDS*GSLUGS))**0.5
GLTVELKTS = GLTVEL * 0.5921052
DT = 0.05
WINDone = abs(WINDC * 6080/3600)
T = 0
X = 0
Z = ALT
DTtwo = DT * DT
UVprep = ANGA * (math.pi/180) #convert to radians
U = 1.69 * VEL * math.cos(UVprep)
V = 1.69 * VEL * math.sin(UVprep)
W = WINDone * (Z/30) ** 0.26
UO = U
VO = V
Zmax = 0
FPEANG = 0

#Data Processing
while Z > Groundlevel:
   VELtwo = U*U+V*V
   if U == 0:
      U=.01
   else:
      U = U
   FP = math.atan(V/U)
   if U<0 and V<0:
```

```
            K = -1
        else:
            K = 1
        DRAG = (SLUGS/2)*VELtwo*CDS
        AX = -DRAG * math.cos(FP) * 32.2 * K/WT
        AZ = -DRAG * math.sin(FP) * 32.2/WT - 32.2
        UO = UO + AX * DT
        V = V + AZ * DT
        U = UO + W
        VO = V
        FPE = math.atan(VO/UO)
        X = X + UO * DT + 0.5 * AX * DTtwo
        Z = Z + VO * DT + 0.5 * AZ * DTtwo
        if Z > Groundlevel:
            T = T + DT
        else:
            T = T
        if abs(Z)>abs(Zmax):
            Zmax = Z * (180/math.pi) #convert to degrees
        else:
            Zmax = Zmax

    #Output Processing
    impactkts = (((UO*UO+VO*VO)**0.5)*0.68182)
    FPE = FPE * (180/math.pi) #convert to degrees

    #Spatial Processing
    WKID = 4326 #WGS84
    sr = arcpy.SpatialReference()
    sr.factoryCode = WKID
    arcpy.env.outputCoordinateSystem = sr
    arcpy.env.workspace = "in_memory"
    arcpy.CreateFeatureclass_management ("in_memory", "Bearing")
    arcpy.DeleteRows_management("in_memory/Bearing")
    arcpy.MakeTableView_management
("in_memory/Bearing","in_memory/BearingTable")
    arcpy.AddField_management ("in_memory/BearingTable", "x_lon", "DOUBLE")
    arcpy.AddField_management ("in_memory/BearingTable", "y_lat", "DOUBLE")
    arcpy.AddField_management ("in_memory/BearingTable", "length", "DOUBLE")
    arcpy.AddField_management ("in_memory/BearingTable", "bearing", "DOUBLE")
    arcpy.AddField_management ("in_memory/BearingTable", "buffDist", "TEXT")

    icrows = arcpy.SearchCursor(inputcrash)

    for icrow in icrows:
        xlon = icrow.getValue("POINT_X")
        ylat = icrow.getValue("POINT_Y")


    #Shill Start
    #rad = Course * (math.pi/180)    #converting Course frorm degree to radians
    R = 6378.1   #Radius of Earth
    d = wingspan * 0.0003048 #distance km
    d2 = 2 * d

    lat1 = math.radians(ylat)
    lon1 = math.radians(xlon)
    # x2 = xlon + (math.sin(math.radians(Course)) * (X/2))
    # y2 = ylat + (math.cos(math.radians(Course)) * (X/2))

    #Second Line
    lat2 = math.asin( math.sin(lat1)*math.cos(d/R) +
         math.cos(lat1)*math.sin(d/R)*math.cos(math.radians(Course)))

    lon2 = lon1 +
math.atan2(math.sin(math.radians(Course))*math.sin(d/R)*math.cos(lat1),
              math.cos(d/R)-math.sin(lat1)*math.sin(lat2))

    #Third Line
    # lat3 = math.asin( math.sin(lat1)*math.cos(d2/R) +
    #      math.cos(lat1)*math.sin(d2/R)*math.cos(math.radians(Course)))
    #
    # lon3 = lon1 +
math.atan2(math.sin(math.radians(Course))*math.sin(d2/R)*math.cos(lat1),
    #             math.cos(d2/R)-math.sin(lat1)*math.sin(lat2))

    lat3 = math.asin( math.sin(lat2)*math.cos(d/R) +
         math.cos(lat2)*math.sin(d/R)*math.cos(math.radians(Course)))

    lon3 = lon2 +
math.atan2(math.sin(math.radians(Course))*math.sin(d/R)*math.cos(lat2),
              math.cos(d/R)-math.sin(lat2)*math.sin(lat3))

    #Converting to degrees
    lat2 = math.degrees(lat2)
    lon2 = math.degrees(lon2)
    lat3 = math.degrees(lat3)
    lon3 = math.degrees(lon3)

    #Updating buffer distance
    cur = arcpy.InsertCursor("in_memory/BearingTable")
    row = cur.newRow()
    row.x_lon = lon3
    row.y_lat = lat3
    row.length = X
    row.bearing = Course
    row.buffDist = str(wingspan + wingspan/2 + wingspan/4) + " Feet"
    cur.insertRow(row)

    row.x_lon = lon2
    row.y_lat = lat2
    row.length = X
    row.bearing = Course
    row.buffDist = str(wingspan + wingspan/2) + " Feet"
    cur.insertRow(row)

    row.x_lon = xlon
    row.y_lat = ylat
    row.length = X
    row.bearing = Course
    row.buffDist = str(wingspan) + " Feet"
    cur.insertRow(row)
    del cur, row
    #Shill End

    arcpy.BearingDistanceToLine_management("in_memory/BearingTable",
"in_memory/BearingLine", "x_lon", "y_lat", "length","FEET", "bearing","", "","", sr)

    arcpy.AddField_management ("in_memory/BearingLine", "buffDist", "TEXT")
    typeCursor = arcpy.da.UpdateCursor("in_memory/BearingLine", ["buffDist"])
    cnt = 0
    for row1 in typeCursor:
        if cnt == 0:
            row1[0] = str(3 * wingspan) + " Feet"
            cnt += 1
            arcpy.AddMessage( "Test " + row1[0])
        elif cnt == 1:
            row1[0] = str(2 * wingspan) + " Feet"
            arcpy.AddMessage( "Test " + row1[0])
            cnt += 1
        else:
            row1[0] = str(wingspan) + " Feet"
            arcpy.AddMessage( "Test " + row1[0])
        typeCursor.updateRow(row1)

    #arcpy.CopyFeatures_management("in_memory/BearingLine", workspace2)
#Debris Line
```

```
    #arcpy.AddMessage("Trying...")

    #Spatial Outputs
    buffer = arcpy.Buffer_analysis("in_memory/BearingLine", workspace, "buffDist",
"FULL", "ROUND", "NONE")

    #arcpy.CopyFeatures_management(workspace,
"C:/Apache/htdocs/RocsLayout/tbx/New File Geodatabase.gdb/Buffer1")

    #Output
    arcpy.AddMessage("Debris Terminal Velocity (kts): " + str(TVELKTS))
    arcpy.AddMessage("Time to Impact (sec): " + str(T))
    arcpy.AddMessage("Debris Throw Distance (ft): " + str(X))
    arcpy.AddMessage("Angle of Impact (deg): " + str(FPE))
    arcpy.AddMessage("Speed of Impact (kts): " + str(impactkts))
    arcpy.AddMessage("Max Altitude of Debris (ft): " + str(Zmax))
    arcpy.AddMessage("Analysis Complete")



#Shill new start
def getToken():
    params = {}
    params['client_id'] = "WOqbbjf8pTaTC9l7"
    params['client_secret'] = "be16b44ef54e43cfb686e17bd53d21fe"
    params['grant_type'] = "client_credentials"
    params = urllib.urlencode(params)
    try:
        request = urllib2.Request('https://www.arcgis.com/sharing/oauth2/token/',params)
        response = urllib2.urlopen(request)
        response = response.read()
        response = json.loads(response)
        token = response["access_token"]
        return token
    except Exception:
        pass

if __name__ == "__main__":
    main()
#Shill new end
```

Appendix D. Measurement Units

The various measurement units required as input and generated as output.

Unit	Description
degree	Denoted by ° and is a measurement of an angle.
knot	Knot, a unit of speed. It is equivalent to 1.151 mph. It is used in meteorology and maritime and air navigation.
AGL	Above Ground Level, measurement with respect to the underlying ground surface.
square feet	Square foot, an imperial unit of area. Defined as the area of a square with sides of 1 foot.
pounds	Pound, an imperial unit of mass.
feet	Foot, a unit of length. It is subdivided into 12 inches.
feet MSL	Foot Mean Sea Level, a type of vertical datum.

Appendix E. Aircraft Values Used in the Microsoft SQL 2008 R2 Database

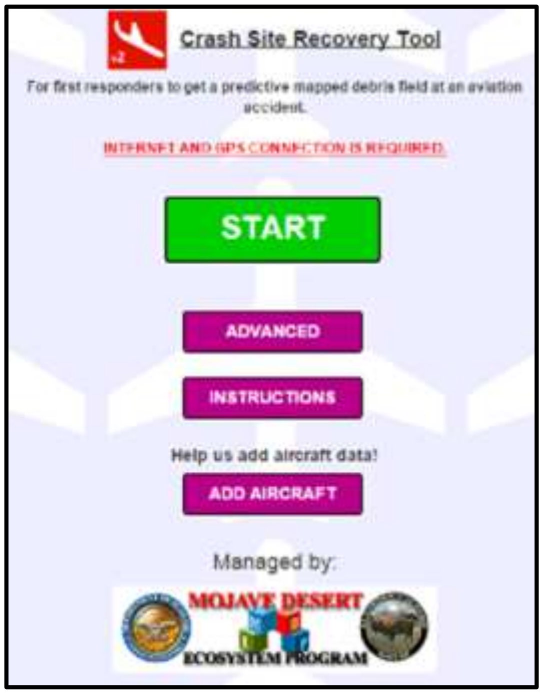
Manufacturer	Model	Drag Coefficient	Weight	Wingspan	Frontal Area	Cruise Speed
Aermacchi	S211	0.02	6050	27.67	135.63	360
Airbus	A300-F4	0.03	308650	147.11	2799	481
Beechcraft	Model 99	0.02	10400	45.88	279.7	205
Boeing	707-320	0.013	257000	130.8	2771	480
Boeing	747-200	0.03	883000	195.66	1098	481
Boeing	737-200	0.03	115500	93	1098	421
Cessna	172 Skyhawk	0.02	2450	36.08	174	122
Cessna	310	0.02	4600	35	175	178
Convair	800	0.02	193000	120	2000	535
Douglass	DC-8-32	0.01	310000	142.42	2771	511
Gates Learjet	Learjet 24	0.02	13500	35.58	231.8	481
Hawker Beechcraft	Hawker 800	0.02	28000	54.33	353	402
Lockheed	F-104G Starfighter	0.0172	29027	21.75	196.1	1154
Lockheed	JetStar	0.01	6050	54.41	840	438
Lockheed Martin	F-22 Raptor	0.02	83500	44.5	840	1304
McDonnell Douglas	F-4 Phantom	0.02	61795	38.38	530	506
Piper	PA-31 Navajo	0.02	6500	40.67	229	207
Ryan	Navion	0.05	2750	38.38	184	135
UNKNOWN	LIGHT	0.02	4000	38	200	150
UNKNOWN	MEDIUM	0.02	20000	45	250	300
UNKNOWN	HEAVY	0.03	300000	150	1800	410

Appendix F. Instructions to use the Rocs Application

Welcome screen: The Recovery of Crash Site Tool (RoCS) requires an active internet connection and geolocation enabled to process a request. The screen provides four main features:

1. START: A quick solution requiring minimal input to allow for rapid response in a time critical situation.
2. ADVANCED: Allows a user to input more details about the incident to obtain a more accurate response.
3. INSTRUCTIONS: This page. Provides the user additional information on this product.
4. ADD AIRCRAFT: Allows for a crowd sourcing effort to take place to keep the solution current and accurate.

Each of the following pages in the solution has a START OVER button that takes the user back to this welcome screen if they are interested in doing so.
The application DOES NOT WORK WITH INTERNET EXPLORER.

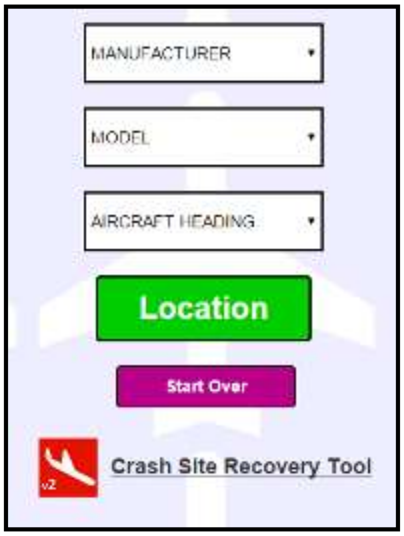


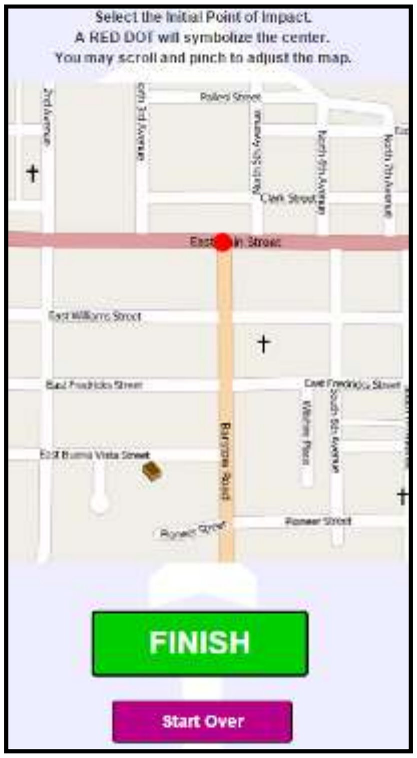
Crash Site Recovery Tool
For first responders to get a predictive mapped debris field at an aviation accident.
INTERNET AND GPS CONNECTION IS REQUIRED
START
ADVANCED
INSTRUCTIONS
Help us add aircraft data!
ADD AIRCRAFT
Managed by:
MOJAVE DESERT
ECOSYSTEM PROGRAM

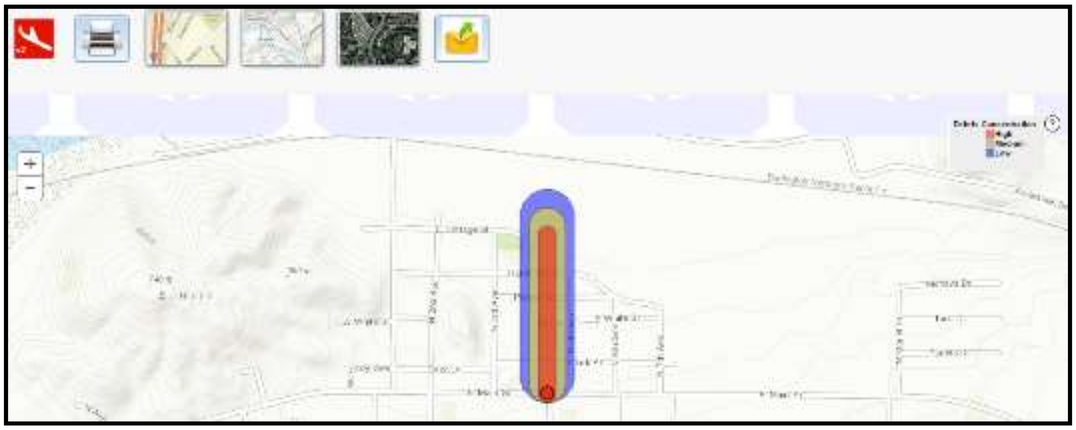
START: Once the user engages START they are taken to a page that asks them the following questions: Manufacture, Model, Aircraft heading and Location. The first three questions are all dropdowns allowing the user a rapid method to select the correct plane involved in the incident.

Location will take the user to a new page that will display a map and center on the device location of the user. The user can select with either their finger or mouse the center of impact on the map. If the map is in the wrong location the user can zoom in or out along with panning to the correct area. Once the user has selected the point of impact they can then select FINISH.

This will now allow the system to process the inputs to return a result. At this time, results show a general concentration of debris fields caused by the aircraft impact on the area. Users can zoom and pan the map to orientate themselves with the area. The buttons across the top allow the user to change map backgrounds, print and share these results via email with others who will be assisting in the incident.

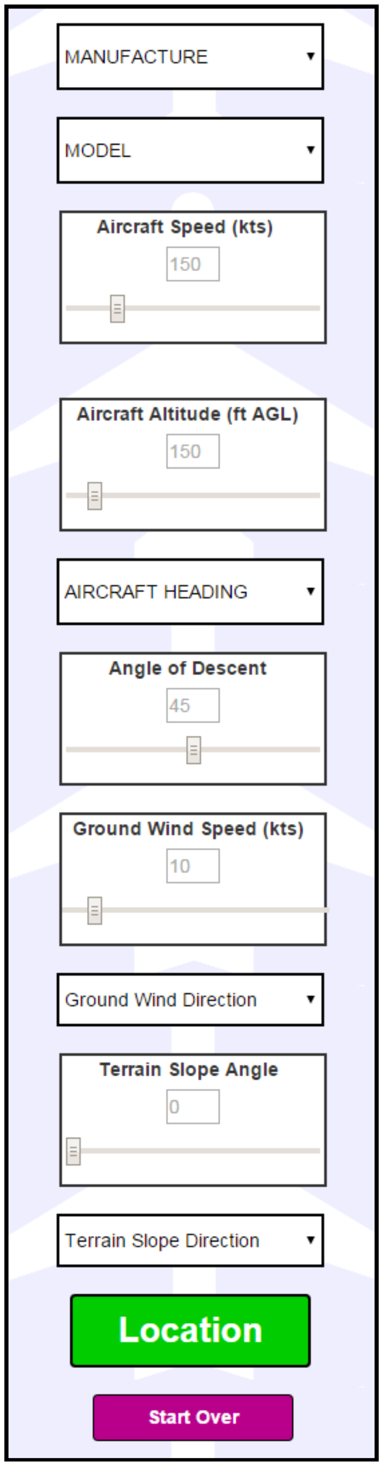



Select the Initial Point of Impact.
A RED DOT will symbolize the center.
You may scroll and pinch to adjust the map.

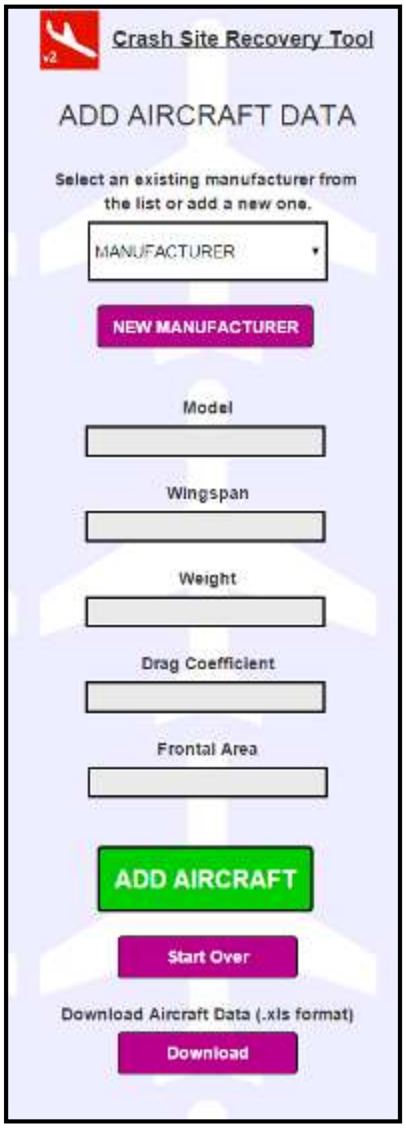


ADVANCED: This option is similar to the START option listed above. The following are additional factors the user can input if they are known about the incident. Aircraft Speed (kts), Angle of Descent, Ground Wind Speed (kts) and Terrain Slope Angle can be adjusted using the slide bar or entered into the number text box. Ground Wind Direction and Terrain Slope Direction are both dropdowns the user can select from the listed options.

The Location Selection and Results page are the same as when used with the START button, and include the same map, printing and sharing features.



ADD AIRCRAFT: This part of the tool allows interested parties to input facts about different models of planes that may not be available currently for analysis. Users can select from a manufacture or add a new manufacture if the one they are looking for is not listed. After the manufacture has been entered then the factory model, wingspan weight, drag coefficient and frontal area need to be entered to allow the model to compute the impact area correctly.



Background: The RoCS has been under development since 2011 to help support an underserved need. Many first responders approach an aircraft incident with little or no support resources and this solution was drafted to assist in these kinds of situations. An incident has many factors that come about over time. First and foremost is the recovery of survivors along with containing any dangerous situations like fire. Second comes analysis and material recovery and third is remediation and restoration. An

incident can take years to complete some or all of these steps and so many groups and individuals are involved but all need data to be able to be effective along the way.
In coordination with the University of Redlands, graduate students have been presented with the scope of this effort. The results have been finding proven models and updating them to work through mobile portals to give incident responders a tool to aid in their efforts. The results so far are promising but also need to continue to improve. As more individuals use this tool and provide results and feedback our hope is that over time the solution will increase in its accuracy and utility. To understand more of how the solution operates please see the completed papers and documentation by Shilpi Jain.

Appendix G. Crash Site Images

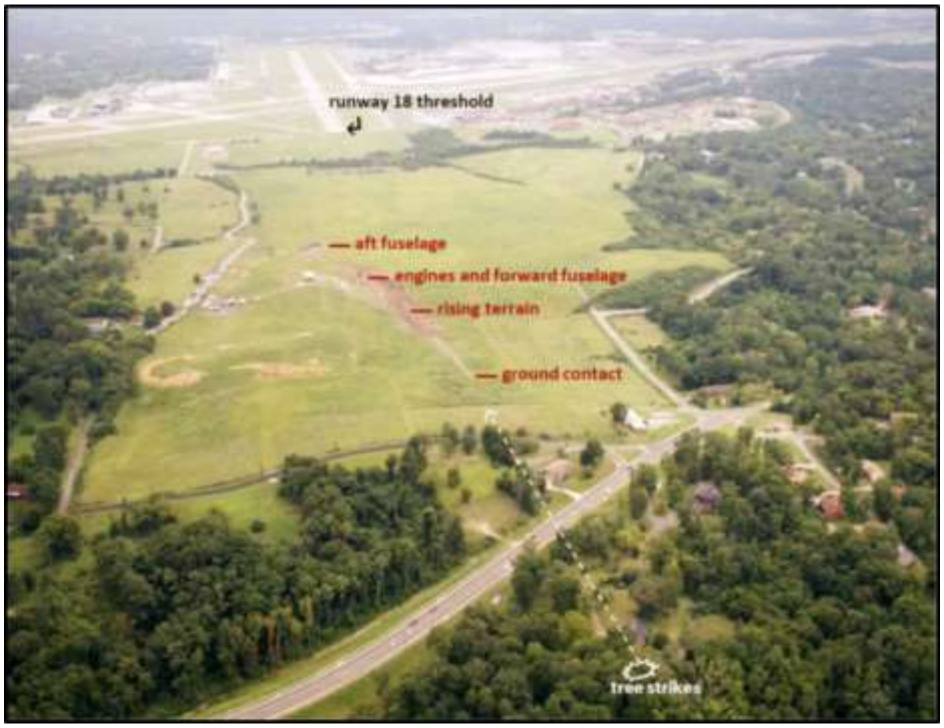


Figure: 2013 Alabama crash (National Transportation Safety Board, 2014)

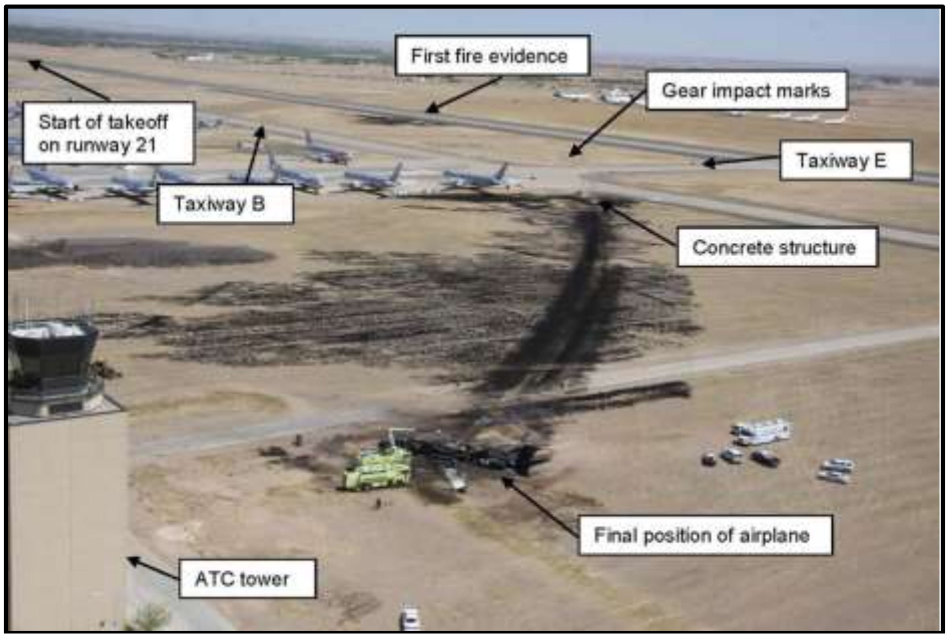


Figure: 2011 New Mexico crash (National Transportation Safety Board, 2012)

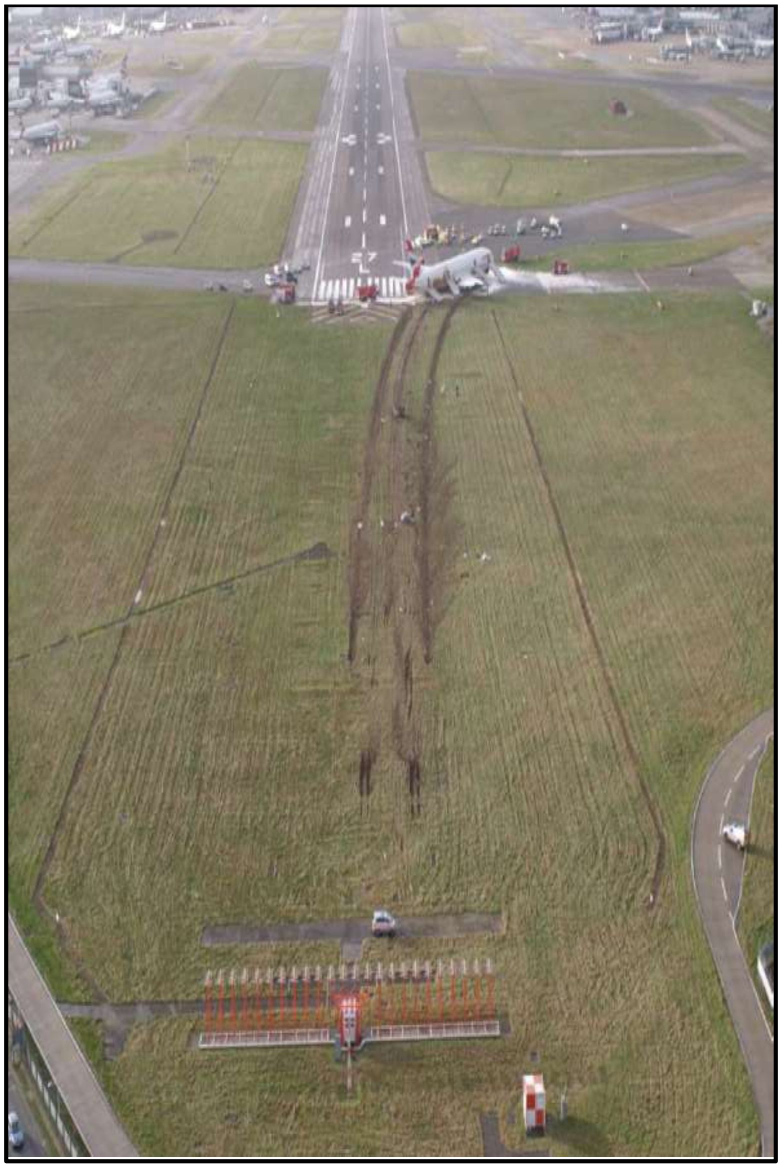


Figure: 2008 London crash (Department for Transport (Air Accidents Investigation Branch), 2010)

Appendix H. Formulae

Adapted from http://www.movable-type.co.uk/scripts/latlong.html, these formulae were used to move the crash location as described in Section 5.1.

$$newLat = \sin^{-1}\left(\sin(lat) * \cos\left(\frac{d}{R}\right) + \cos(lat) \times \sin\left(\frac{d}{R}\right) \times \cos(heading)\right)$$

$$newLon = lon + \tan^{-1}\left(\frac{\sin(heading) \times \sin\left(\frac{d}{R}\right) \times \cos(lat)}{\cos\left(\frac{d}{R}\right) - \sin(lat) \times \sin(newLat)}\right)$$

$$R = 6378.1\ km$$

$$d = wingspan\ (km)$$