

Operator Interface for Autonomous Unmanned Aircraft Mission Feasibility and Control

by

Sean R Perry

Bachelor of Science in Electrical Engineering, UNB, 2006

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF**

Master of Science

In the Graduate Academic Unit of Electrical Engineering

Supervisor(s): James H Taylor, Ph.D., Electrical and Computer Engineering
Examining Board: Christopher P. Diduch, Ph.D., Electrical and Computer Engineering
Howard Li, Ph.D., Electrical and Computer Engineering
Bradford G. Nickerson, Ph.D., Computer Science

This thesis is accepted by the

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

December, 2009

©Sean R Perry, 2009

Dedication

To my father, mother, brother and sister. Without your love, guidance and support this never would have been possible.

Abstract

This thesis details development of a software package which provides a unique service to unmanned aerial vehicle operators. The software has two modes of operation, Planning Mode and Execution Mode. In planning mode the operator can easily simulate a desired flight course to determine mission feasibility. A key feature of this mode is the optional integration of forecast numerical weather data, weather charts, and interactive trajectory planning. In Execution mode the operator can deploy a flight, change the route “on the fly”, hold the aircraft over a point of interest, and return the aircraft to home base.

Any data required for these functions is automatically acquired from appropriate sources (NAV Canada, Environment Canada) via the internet. Operation of the software, including interactive trajectory planning, weather observing, and mission progress, is achieved through a user-friendly graphical user interface. Simulation results, flight trajectories, and weather data can be saved for further analysis. By comparing missions both with and without weather information, it was determined that meteorological information can have a very significant impact on the mission planning duration and fuel consumption. In an example mission, including numerical weather model data increased the UAV’s fuel consumption by 18% and mission time by 25%. Finally, basic software documentation is provided as an appendix.

Acknowledgements

Dr. James H. Taylor, University of New Brunswick - Who I have had the privilege of being both my instructor and supervisor. As a teacher you have provided me with an in-depth understanding and appreciation of advanced level topics. As a supervisor your dedication, keen eye for detail, and patience has led to a fine-tuned project. I would sincerely like to thank you for all your guidance and encouragement throughout this project. You have been an excellent mentor, but most importantly a great friend.

Dr. Siu O'Young and the RAVEN Team, Memorial University - For providing the simulation control model and the opportunity to contribute to the Raven project.

Dr. Brian Blanton, Renaissance Computing Institute, University of North Carolina at Chapel Hill - For developing the `read_grib` function.

Maira Omana - For your help with various latex problems.

Table of Contents

Dedication	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	viii
List of Tables	ix
List of Figures	xiii
1 Introduction	1
1.1 Problem Description	2
1.2 Background	4
1.3 Literature Review	7
1.4 Goals/Objectives	8

2	Requirements and Methodology	10
2.1	Requirements	10
2.2	Methodology	11
2.2.1	Numerical Model Weather Data	11
2.2.1.1	Gridded Numerical Weather Data Processing	14
2.2.1.2	Data Interpolation	16
2.2.1.3	Graphical Overlay Product	28
2.2.2	Graphical Weather Charts	28
2.2.3	Map Processing	29
2.2.3.1	Vector Data Imagery	30
2.2.3.2	Raster Data Imagery	32
2.2.4	Simulink [®] Models	35
2.2.4.1	Aerosim Model	35
2.2.4.2	Planning Simulink [®] Model	37
2.2.4.3	Execution Simulink [®] Model	39
2.2.4.4	GUI Software Flow	41

3	Implementation	43
3.1	Numerical Model Weather Data	43
3.1.1	Numerical Model Weather Data Processing	45
3.1.2	Weather Data Interpolation	45
3.1.3	Graphical Overlay Product	50
3.2	Graphical Weather Charts	51
3.3	Map Processing	55
3.3.1	Displaying a Map	55
3.3.1.1	Quick Zoom Tool	58
3.3.2	Simulink [®] Models	59
3.3.2.1	Planning Simulink [®] Model	61
3.3.3	Execution of the Simulink [®] Model	63
3.4	Graphical User Interface	63
4	Presentation of Using the System for Planning	67
4.1	Initialize	67
4.2	Planning	68
4.3	Simulate UAV Flight	69
4.4	UAV Simulation Results	70

5	Presentation of Using the System for Execution	73
6	Contributions, Future Work and Conclusion	75
6.1	Contributions	75
6.2	Future Work	76
6.3	Conclusion	77
	Bibliography	80
	Appendices	81
A	Software Manual	81
B	Grib Data Set Parameters	89
C	Various Weather Charts	91
	Vita	

List of Tables

2.1	Aircraft Model Inputs[1]	35
2.2	Aircraft Model Outputs[1]	36
2.3	Components of the Route Planning Model	37
2.4	Variables that are Logged During Simulation	39
2.5	Variables of the Route Planning Model	40
2.6	Real-time UAV Route Control Options	40
3.1	Input/Output Table	46
3.2	Weather Interpolation Files	50
4.1	Rough Waypoints for an Example Mission	69
4.2	Attributes of the Simulations	71
B.1	GRIB Data Set Parameters	89
C.1	Various Weather Charts	91

List of Figures

1.1	Overview of EFOP Operation	3
2.1	Low Resolution Polar Stereographic Grid [2]	13
2.2	Directory Structure of Downloaded Weather Files	14
2.3	Gridded Numerical Weather Model Data Structure	15
2.4	Data Interpolation Overview	17
2.5	Latitude/Longitude Lines and Polar Stereographic Grid Lines (This image is an illustration and not to scale)	18
2.6	Pressure Levels Above the Polar Stereographic Grid (this image is an illustration and not to scale)	19
2.7	Regions in the Polar Stereographic Grid (This image is an illustration and not exactly topologically correct)	21
2.8	Interpolation Within a Grid Region	21
2.9	Pressure Levels above the Polar Stereographic Grid (This image is an illustration and not to scale)	23
2.10	Single Wind Barb with Temperature	28

2.11	Wind Barb Values	29
2.12	An Example of Vector and Raster Data from Google Maps TM	31
2.13	Low Resolution Satellite Imagery of Earth	32
2.14	Reconstruction of Satellite Amaru Image of Earth	34
2.15	Aircraft Navigation from Fine Waypoint to Waypoint	38
2.16	Software Flow Chart	42
3.1	Code used to Download the GRIB Data	44
3.2	Code used to Read GRIB Data into Memory	45
3.3	Code used to Process the Weather Data in a Data Structure	45
3.4	Binary Search Algorithm used to Determine the Height Index	47
3.5	Code used to Interpolate each Corner of Weather Block Based on Height Index	48
3.6	Code used to Combine Each Corner into the Weighted Point Value	48
3.7	Code used to Combine each Time Value into the Weighted Value	49
3.8	Code used to Rotate between Two Frames of Reference using Small Perturbation Linearization	49
3.9	Graphical Weather Overlay Initialization Screenshot	51
3.10	Graphical Weather Overlay Over Newfoundland	52
3.11	Method of Selecting Relevant Weather Charts	53

3.12	Downloading Weather Charts	53
3.13	Method of Viewing Relevant Weather Charts	54
3.14	Code used to Display the Raster Map Data	55
3.15	Code used to Crop the Image	56
3.16	Code used to Create Reference Matrix R	57
3.17	Code used to Read and Plot Shape Data	57
3.18	Screen-shot of a Map	58
3.19	Screen-shot of the Quick Zoom Tool	59
3.20	Planning or Executing Simulink [®] Model	60
3.21	Code used to Track a Route	62
3.22	Code used to Log Data from the Simulink [®] Model	63
3.23	Code used to Read and Plot Shape Data	64
3.24	Code used to Change Flight Mode of Aircraft	65
3.25	Screen-shot of EFOP software GUI in MATLAB's GUIDE Development Environment (Some GUI elements overlap, since their visibility is toggled on and off as desired)	66
4.1	Plot of Planned Latitude/Longitude while in Planning Mode	69
4.2	Plot of Planned Altitude while in Planning Mode	70
4.3	Screen-capture of EFOP while in Planning Mode	71

4.4	Plot of Logged Lat/Lon while in Planning Mode	72
4.5	Plot of Logged Altitude while in Planning Mode	72
5.1	Plot of a Logged Data while in Execution Mode	74
A.1	Screen-capture of EFOP while in Planning Mode	83

Chapter 1

Introduction

The ability of an operator of an **unmanned aerial vehicle (UAV)** to plan and assess the feasibility of a proposed flight plan and then to execute it benefits the Canadian public in both economic and security terms. UAV surveillance can be highly beneficial for security issues. Security does not specifically refer to military threats, but refers to protecting the Canadian Public against situations that jeopardize their safety, resources, and the environment. Three situations where UAV surveillance would be extremely beneficial are finding and tracking icebergs, detecting illegal fishing and monitoring oil spills. Being able to monitor and react to these situations increases our security.

For UAV research and development to be sustainable, UAV operation must be economically feasible. This entails incurring minimal unnecessary losses of unmanned aircraft, as may happen in adverse weather conditions or on overly ambitious missions leading to fuel becoming exhausted. A system to minimize losses needs to include an aircraft model, aircraft simulator, aircraft automated control system,

weather forecasting system, and a user interface. Together, these components allow an operator to simulate an anticipated mission and determine the likelihood of success or failure. The significance of the predictive nature of such a feasibility study is compounded, since there are exogenous factors that strongly determine the probability of the success or failure. Meteorological conditions are the key factor when considering if a proposed flight plan will succeed or fail. Accounting for weather effects can drastically change the planning and outcome of a trip. This research will address the design and implementation of an operator interface for UAV mission planning and execution. It will be used as part of the RAVEN project [3], an AIF-funded program led by Dr. Siu O'Young of the Memorial University, under which UNB has mission control responsibility.

1.1 Problem Description

The mission of this project is to develop and implement a software package that is capable of simulating and controlling a UAV. The software will have two primary modes of operation. The first mode is called **feasibility/planning**, in which a mission can be planned and simulated. The simulation uses forecast gridded weather data from Environment Canada's **Global Environmental Multiscale** (GEM) model. Time variant forecast weather data can be extracted from the GEM model data using a complex method of interpolation. By reading this environmental data into a simulation, the user is able to determine if a particular mission is feasible or not given the anticipated weather conditions. The second mode of operation is called **execution**. In this mode the software is connected to and remotely controls an actual UAV. Flight trajectory changes are permitted on the fly. This software is also designed to both acquire and display all pertinent weather information.

The end product of this project is the **Environmental Feasibility Operations Package (EFOP)**, which is a prototype software package that allows a trained operator to plan, determine feasibility, and execute a UAV mission. Figure 1.1 depicts a typical scenario of EFOP in action. The operator is located at a remote terminal, which has the ability to communicate with the UAV. Before the UAV is deployed, the operator defines and simulates a proposed mission trajectory to determine feasibility, based on forecast weather. Once the optimal or acceptable route has been determined, the operator proceeds to execute the mission. The mission route is shown by the red line in figure 1.1. The UAV proceeds to track this route while constantly scanning for an event. Two different points of detection have been shown along the aircraft's route. These points are shown in figure 1.1 by the yellow stars. Once an event has been detected, the operator has the option to investigate these events by changing the flight trajectory, or to proceed as originally planned.

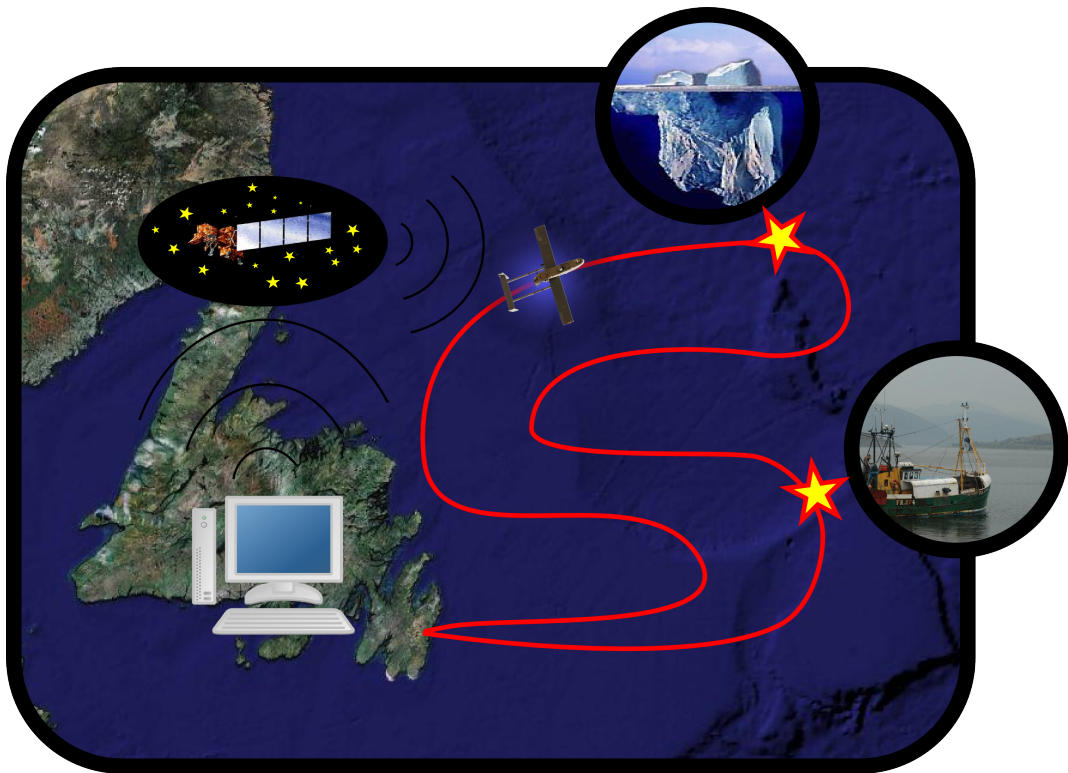


Figure 1.1: Overview of EFOP Operation

1.2 Background

This project incorporates both technology and data from many different sources. Connecting data and tools into a software package requires knowing the main players. The following sections briefly describe the groups, software, and technologies that were used to create EFOP.

RAVEN [3]

Manned aerial surveillance is currently performed off the coast of Newfoundland by Provincial Aerospace Limited (PAL). Using a Beechcraft KingAir 200 aircraft, PAL's missions are approximately 1000 nautical miles, last 6 to 8 hours, and are typically flown at an altitude of 1000 feet. The main goal of the **R**emote **A**erial **V**ehicles for **E**Nviromental monitoring project (RAVEN) is to supplement PAL's manned missions through the use of UAVs. In order to successfully develop this product, the harsh maritime operating environment needs to be accurately approximated for the purpose of testing proposed flight plans using high fidelity unmanned aerial vehicle models and accurately forecast weather. By supplementing PAL's current maritime surveillance regime with an operational UAV, increased cost efficiency and improved surveillance could possibly be achieved.

MATLAB[®] [4]

MATLAB[®] is a numerical computing environment and programming language that is created and maintained by The MathWorks. Matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs in other languages are all easily accomplished in

MATLAB[®]. It uses the MATLAB[®] programming language called M-code. MATLAB[®] provides the user with a full development environment. Commands can be run via the command window, variables can be displayed in the workspace and scripts can be created/edited in the editor. MATLAB[®] also includes facilities to debug m-code.

MATLAB[®] specializes in numerical computing, but is fortunately not limited to that alone. Toolboxes, a grouping of highly specialized functions, have been developed for various needs. Some examples are the signal processing toolbox, the Mapping Toolbox[™], and the Image Processing Toolbox[™]. These toolboxes give MATLAB[®] the ability of accomplish highly specific tasks quite efficiently. The Mapping Toolbox[™] and Image Processing Toolbox[™] are particularly useful in this project. The Mapping Toolbox[™] provides the ability to easily display different mapping data and images, translate between mapping projections, and perform geo-spatial translations. The Image Processing Toolbox[™] allows MATLAB[®] to properly read, process, and render image files.

MATLAB[®] also has the ability to create reasonably complex graphical user interfaces. Development of these interfaces is completed in MATLAB[®]'s **GUI Design Environment** or **GUIDE**. GUIDE allows GUI's to be created and edited interactively from figures. Callback commands are located in an accompanying M-file. GUIDE developments can include a menu system for easy program navigation. Through GUIDE it is also possible to interact with the MATLAB[®] workspace in a normal manner. Variables can be written to MAT-files and loaded back into memory at a later time. GUIDE also utilizes MATLAB's method of handling graphics. Handles are used to make alterations to objects in the figure.

Simulink[®] [5]

Simulink[®] was developed by MathWorks and is used for modeling, simulating, and analyzing multi-domain dynamic systems. Simulink[®] uses interconnected graphical objects or blocks. Simulink[®] comes with a basic blockset that is capable of most basic tasks. Specialized blocksets are developed for particular tasks. An example of the latter would be a blockset designed specifically to model an aircraft. The integration between MATLAB[®] and Simulink[®] is very tight, and it is possible to program Simulink[®] blocks using MATLAB[®] code. These blocks are called S-functions.

Aerosim Blockset [1]

The Aerosim blockset is a complete set of aeronautical simulation tools that provide the user with the ability to rapidly develop a nonlinear 6-degree-of-freedom aircraft dynamic model. Aerosim includes both basic aircraft dynamics blocks and complete aircraft models. It is possible to customize these models through the use of parameter files. Interfacing with Flightgear is accomplished using one of Aerosim's flightgear blocks. Flightgear is a sophisticated open-source flight simulation packages that is released under the GNU General Public License.

Environment Canada [6]

Environment Canada is a department of the Government of Canada responsible for coordinating environmental policies and programs as well as preserving and enhancing the natural environment and conserving wildlife. Environment Canada is also responsible for meteorology through the Meteorological Service of Canada, which

produces weather forecasts, public meteorological information, and severe weather warnings. Environment Canada is also responsible for environmental research, and they produce numerical weather models.

1.3 Literature Review

Due to the innovative nature of this software project, a traditional literature review on a specific topic is not possible. Certain components and technical aspects of this project have been researched.

The Raven Project is detailed in reference [3]. In order to develop this project, both Matlab and Simulink were used. Reference [4] and [5] both point to basic information on The Mathworks' website. There are several interesting sources with respect to mapping and map generation. Raster data is obtained from reference [15] (Blue marble next generation), and many vector datasets were easily obtained from Natural Resources Canada [13]. All vector information was stored in a shape file [14]. Additionally a series of weather charts from NAV Canada are routinely downloaded [12].

The World Meteorological Organization [8] promotes standardization of meteorological data and helps facilitate worldwide cooperation. Environment Canada's numerical weather data is stored in a GRIB Edition 1 format which is described in reference [10]. The most common format to distribute gridded forecast data is called GRIB or **GR**Idded **B**inary. Environment Canada [6] provides its gridded numerical weather prediction models, as well as documentation [2] [7] [16], in GRIB format for free. Fortunately software exists for reading numeric GRIB data directly into

Matlab. A function called read-grib [9] has been well developed and was used in this work. An efficient binary searching routine was found in reference [17]. The basic bilinear interpolation used to interpolate this weather is described in reference [11]. The aircraft model that was chosen for this projects is called the Aerosim Blockset produced by Unmanned Dynamics. Reference [1] is the manual for the Aerosim Blockset. Overall the software was not as responsive as desired; hence, a method to speed up the software was researched [18].

1.4 Goals/Objectives

This work is a unique type of masters research project. It involves the aggregation and integration of various data sets, software packages, and technologies to produce an interactive utility which ties together and processes all relevant information necessary to plan and operate a UAV mission through the development of a custom software package, EFOP. One of the goals of the project is to link to the various technologies and information sources with software that is both easy to use and robust. In particular, the following attributes are critical:

- This software must be user friendly.
- It should be interactive and also allow the user to monitor and control real time situations.
- It must include a seamless method for acquiring and using relevant environmental data.
- It should be easily maintainable and expandable.

This project is an integral part of the Raven project, as it contributes valuable information for UAV planning and execution which is generated by EFOP. This project

focuses on the graphical interface, the effect of weather, and flight trajectories. Furthermore, it uses this information to determine the feasibility of proposed missions and greatly simplifies executing these trajectories.

This thesis builds directly upon work being completed by other Raven members, since it is designed to work with the Raven aircraft control model. With future development, Raven may be able to use this software for all their mission planning and execution needs. This thesis will attempt, through the use of developed software, to determine whether incorporating meteorological information into a mission planning procedure will result in either more refined flight trajectories or a more practical mission.

Chapter 2

Requirements and Methodology

2.1 Requirements

These are the requirements that the software development process was governed by:

- This developed software should be applicable anywhere in Canada.
- The UAV operator should have easy access to relevant forecast environmental data and charts that are required to make educated mission decisions.
- Such data used to form an educated decision should be utilized to provide a rigorous verification by high fidelity simulation.
- All interaction with the software should be through an effective and intuitive graphical user interface.
- Mission routes should be easy to define and modify in all three dimensions.
- Gridded numerical weather models should be the source of weather information.
- Graphical overlays and weather charts should be presented and formatted such that they are easily understood by a person familiar with aviation practices.

2.2 Methodology

The following describes how each section of this software package functions, how these sections combine to produce EFOP, and how an operator would use this software package.

2.2.1 Numerical Model Weather Data

In this project a robust source of numerical weather data was needed to assess the flight conditions for an entire mission. The optimal meteorological numeric model would cover a large area, be presented over a spatial grid, include various meteorological variables, and span a suitable time frame. After considerable research a suitable model from Environment Canada was found; it is called the low-resolution **Canadian Meteorological Centre (CMC) GRIdded Binary (GRIB)** database [7].

Environment Canada offers several models at different resolutions. The low resolution regional model has a 60 km central core resolution while the high resolution regional model has a 15 km central core resolution. There is also a global model with a 220 km or two degree by two degree grid. The size of the gridded numerical weather model was a concern while developing this project. If the gridded numerical weather data files are too large it will cause the data retrieval, data processing, and data interpolation to require an unreasonable amount of time. Likewise, if the data's resolution is too low it will not provide adequate information. A requirement was that the software be at least capable of operating within Canada, so considering the very poor resolution of the global model, it was dismissed as unnecessary. When processing the high and low resolution regional data-sets into `mat`

files (MATLAB[®] specific data storage files) it was determined that the low resolution data-set needed approximately 50 megabytes of storage and the high resolution data contained nearly 20 times more data. A subset of the high resolution model was not used, since all data needs to be extracted from binary files before it can be used. Converting from the binary format can be time consuming. For these reasons the low resolution regional model was chosen. It was also felt that a 60 km grid had enough information to offer a meaningful interpolation over the coverage of an anticipated mission.

This low resolution model's grid, shown in figure 2.1, is based on a polar stereographic mapping projection. This polar stereographic mapping projection grid consists of $n_i = 135$ grid lines horizontally and $n_j = 95$ grid lines vertically. The resolution at sixty degrees north is 60 km, and the coordinates of the first grid point (bottom left point) are 27.203 degrees north and 135.213 degrees west, somewhere in the Pacific Ocean. The north pole is at grid index (50.6, 111.3) and the orientation of this polar stereographic grid is rotated -111 degrees with respect to the j axis. Unfortunately the lines of the grid do not correspond with the lines of latitude and longitude on the earth. Working with and translating between two different coordinate systems can be very tricky and requires exact calculation. Translating between latitude and longitude coordinates to grid indices becomes a very mathematically intense problem. Fortunately algorithms to translate between different projections have been built into the Mathworks Mapping Toolbox.

It was not possible to acquire the gridded numerical weather data from Environment Canada directly via MATLAB[®] since there are no direct MATLAB[®] routines for downloading files. MATLAB[®] does have the ability to concatenate strings and execute operating system command. Command line software, called CURL, is capable

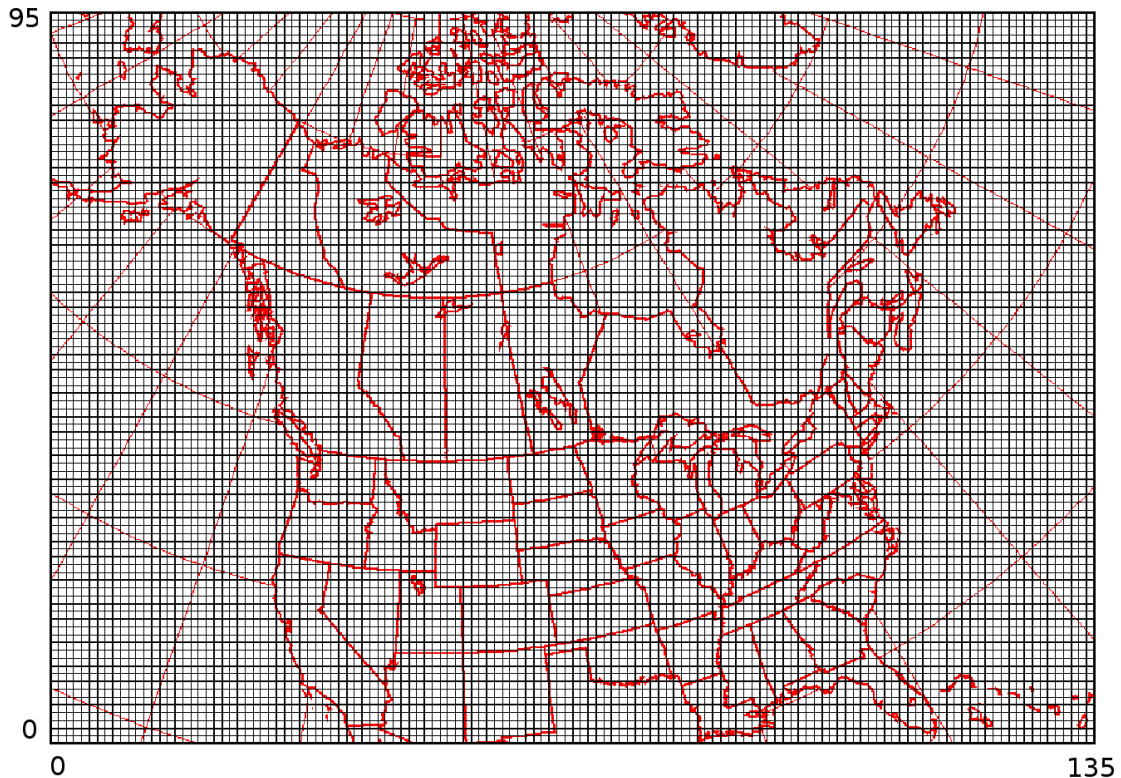


Figure 2.1: Low Resolution Polar Stereographic Grid [2]

of downloading large lists of data files and can thus be run from the MATLAB[®] command prompt. The complete dataset for the low resolution model consists of 1008 files that are approximately 16 KB each, altogether totaling 14.5 megabytes of binary data. Download times can be lengthy even with a high speed connection. The use of wireless internet is strongly discouraged, as it tends to drop the occasional data file.

The directory structure of the data files can be seen in figure 2.2. There are four meteorological variables that are downloaded (HGT, TMP, UGRD and VGRD, i.e. height, temperature, and x, y wind speed components). A directory is created for each variable and each of these directories is filled with nine time directories. Accordingly each time directory is named P000, P006, P012, ... P048 since they represent

$T_1 = 0$ (time of forecast) and $T_9 = 48$ hours later. Each time directory contains twenty-eight grib data files and an example name of one file is `CMC_HGT_50.grib`. In this file name “HGT” represents the meteorological variable and the number “50” represents the isobaric pressure level of the gridded data in that file.

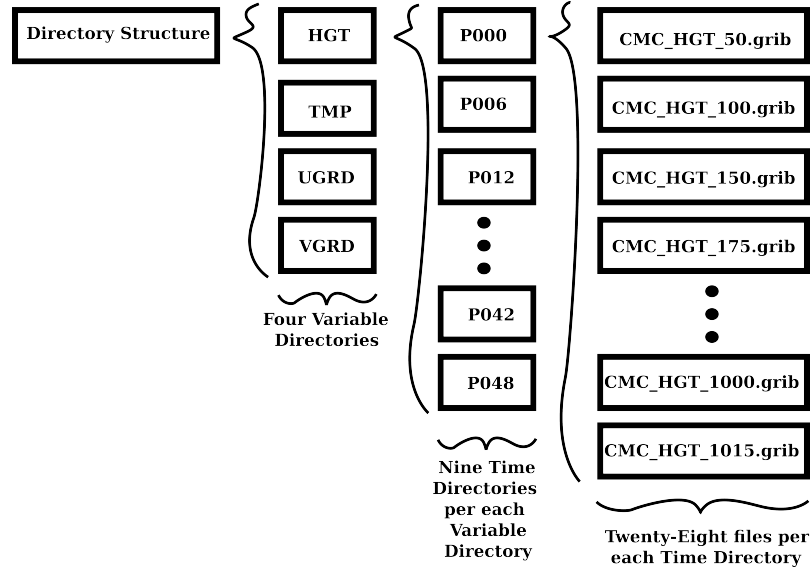


Figure 2.2: Directory Structure of Downloaded Weather Files

2.2.1.1 Gridded Numerical Weather Data Processing

Importing the data into MATLAB[®] is not a trivial task. Each binary data file is encrypted using a compressed wavelet format; it is a special data encryption scheme called GRIB. The GRIB format was introduced by the World Meteorological Organization (WMO) to be a uniform data format for all countries to use. GRIB files can come in three different flavours, GRIB edition 0, GRIB edition 1, and GRIB edition 2. The GRIB 0 format has been retired and all of Environment Canada’s Models are output in the GRIB 1 format. This is an open formatting scheme with full documentation available on WMO’s webpage [8].

By skimming the GRIB 1 manual, it became immediately evident that program-

ming a GRIB reader into MATLAB[®] would be a very large task. Fortunately a MATLAB[®] GRIB reader is under development called `read_grib` [9]. At the time of writing this document the latest version is 1.4.1. Through trial and error it was determined that this reader did not function properly with Environment Canada’s GRIB files. By comparing the GRIB manual [10] to the source code provided with it, `read_grib` was adjusted by the author to properly read Environment Canada’s GRIB files.

Storing data in the GRIB format is extremely efficient when the goal is to archive and share model data, but it is not efficient if the data is needed to be dynamically called into computer memory by MATLAB[®]. For this reason a routine was developed by the author that would systematically extract all pertinent information from the GRIB files and write it into a MATLAB[®] data structure called a MAT file. The data is stored in a format that is detailed in figure 2.3.

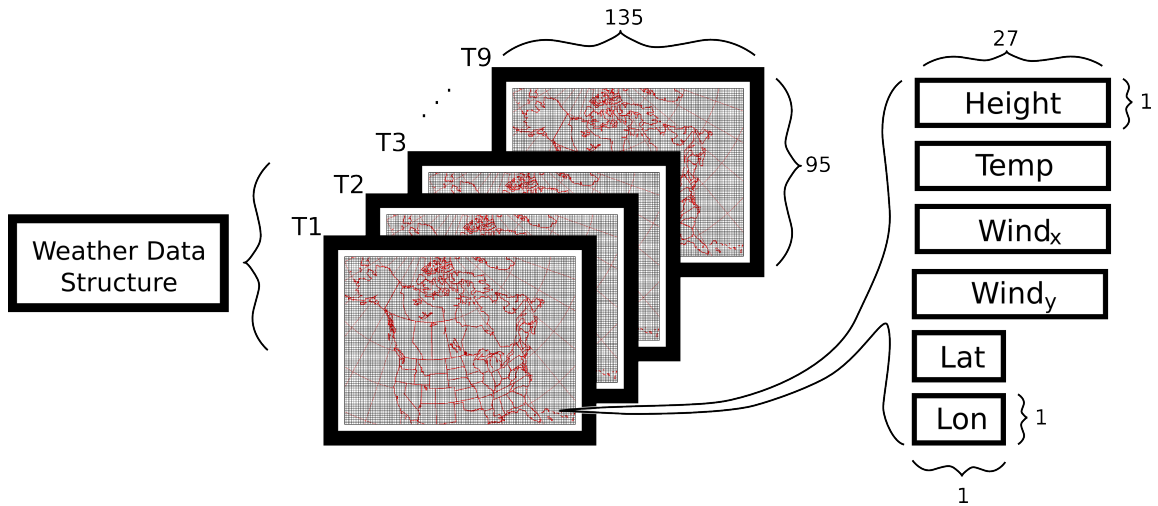


Figure 2.3: Gridded Numerical Weather Model Data Structure

Updated gridded numerical weather data is available for processing every day. Figure 2.3 helps illustrate the complexity of the weather data structure that is constructed from the GRIB files. As shown in figure 2.3, the weather data structure is

broken up into nine grids or MATLAB[®] cell arrays. Each grid represent a different instant in time. There are 9 snapshots, in 6 hour increments, giving a 48 hour forecast. For example, the grid T0 contains weather data from 00 hours GMT, T1 contains weather data from 06 hours GMT, and T9 contains weather data from 48 hours GMT. Each grid has a dimension of 135 by 95 and every grid point represents a geographic location where various geographic and environment datasets are stored (figure 2.1). The geographic data stored at each grid point is the latitude and longitude of the point. The environment datasets that are stored at each grid point contain data pertaining to height, temperature, X component of wind and Y component of wind. While the geographic data elements are a number, the environmental data elements are a set of numbers. Each environmental data element consists of 27 numbers which represent the value at a different isobaric level. The 27 isobaric levels are 1015, 1000, 985, 970, 950, 925, 900, 875, 850, 750, 700, 650, 600, 550, 500, 450, 400, 350, 300, 275, 250, 225 200, 175, 150, 100 and 50 hPa.

2.2.1.2 Data Interpolation

Once the data has been processed and stored in the MAT-file weather data structure, a complex interpolation scheme is used to efficiently calculate specific data for the proposed flight trajectory. This interpolation routine must consider geographic location (latitude and longitude), altitude and time when calculating the specific weather information. The data interpolation is broken down into six steps. The interconnection of these steps and their description is detailed in figure 2.4.

Step 1 - Determine the index of geographic point in weather grid

When a grid is specified it is described using a mapping projection. By translating between different mapping projections, this project can incorporate data from many

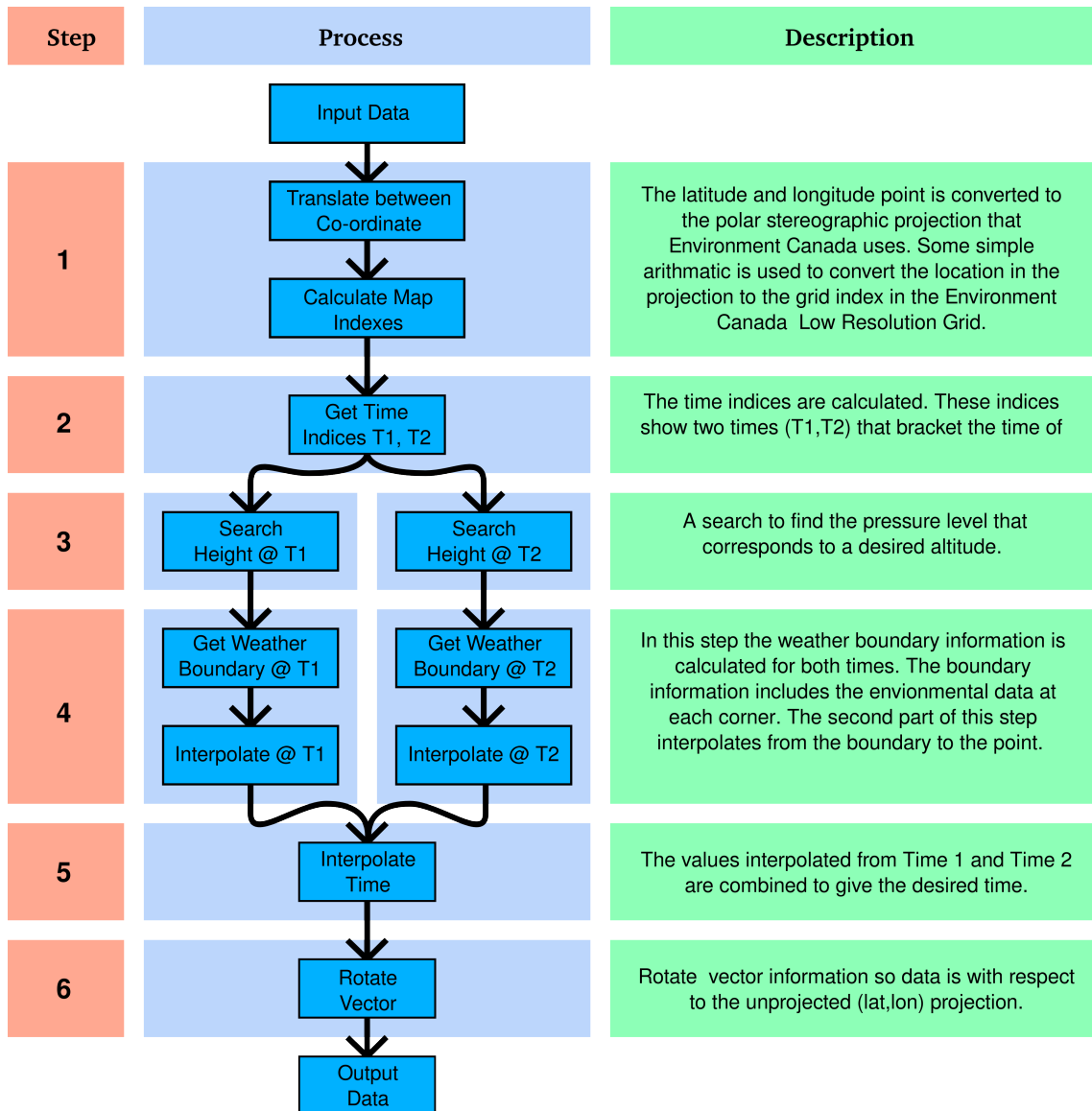


Figure 2.4: Data Interpolation Overview

different sources. The Environment Canada grib numerical model data is stored on a polar stereographic grid. Stereographic projections are beneficial in polar regions. In a polar stereographic projection directions are true from the center point, scale increases away from the center point and distortion increases in area and shape.

Originally a complex search method was used to determine the index of the latitude and longitude in the polar stereographic grid. Since the polar stereographic

projection is mathematically defined, it is possible to translate between these two frames of reference. Using this coordinate system translation scheme and some simple calculations, the exact indexed position in the polar stereographic grid can be determined. The ability to translate between the polar stereographic grid coordinate system and the unprojected coordinate system allows the user to obtain the exact grid index coordinate for any desired latitude and longitude. From this point onward it is only necessary to perform calculations with respect to the polar stereographic grid frame of reference. Figure 2.5 illustrates the differences between these two frames of reference.

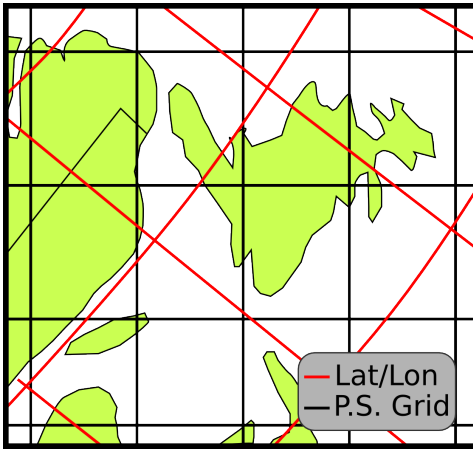


Figure 2.5: Latitude/Longitude Lines and Polar Stereographic Grid Lines (This image is an illustration and not to scale)

Step 2 - Get Time Index

All interpolated data is interpolated at a particular time. For example, assume that weather data was required at 9:00. Environment Canada's numerical model is a discrete model and therefore provides a model snapshot at six-hour intervals. For interpolation purposes, the proportional temporal location between two of these snapshots is required. The model snapshots have 6 hours intervals starting at 00h; therefore, the time index at 9:00 would be 2.5. In order to include time in the

interpolation process, the next two weather interpolation steps need to be completed at each of the boundary snapshots, i.e., both $t_2 = 06h$ and $t_3 = 12h$ in this example.

Step 3 - Searching Isobaric Pressure Levels

All the data in the weather data structure is vertically sorted by pressure levels and not altitudes. Unfortunately all data interpolations must be based on altitude and not pressure level. The relationship between pressure levels and altitude is not constant and depends on the characteristic of the modelled air mass; therefore, at each (lat, lon) grid point there is a different altitude/pressure relationship. Since the data is pre-sorted, a binary search algorithm was used to determine the pressure level of a desired altitude. An illustration of the pressure levels and a constant altitude is depicted in figure 2.6. When viewing this illustration it is important to remember that the z-axis is pressure, not altitude.

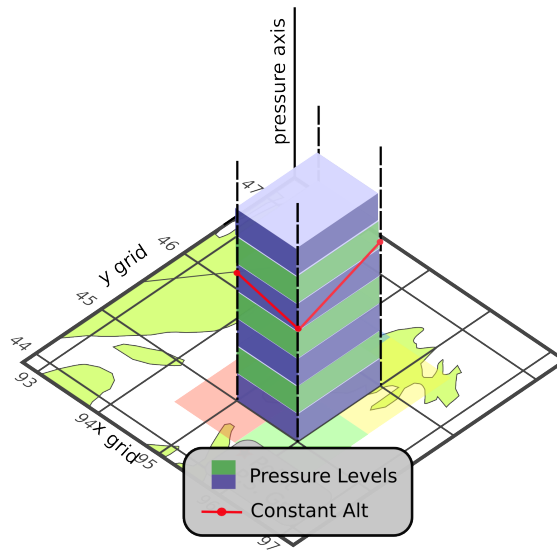


Figure 2.6: Pressure Levels Above the Polar Stereographic Grid (this image is an illustration and not to scale)

Step 4 - Interpolate Location

To successfully interpolate an exact location within the numerical model's grid, two steps must be taken. First the weather boundary must be calculated and second these boundary values must be interpolated for a given location, altitude and time.

Although the search method for determining the pressure/altitude relationship is very efficient, it is computationally intensive. Therefore this is the logical place to trim computational time. A simplifying assumption was made when determining the location of the height index : The index used to interpolate height was calculated at each of the four corners. from the closest grid point. Several factors contributed to this assumption being reasonable: First, the interpolation process weights the closest index point highest, and second, the variation in the height index appears to be minimal. Finding the closest grid point was accomplished by rounding off the precise grid location of the converted latitude/longitude. For example, assume that the grid index of the point of interest is (95.45, 45.78); therefore, the closest grid coordinates would be (95, 46). In Figure 2.7 the different corners can be seen. The blue area corresponds to the upper left grid point, the red area to the lower left grid point, the green area to the lower right grid point, and the yellow area to the upper right grid point. The location in this figure falls in the upper left grid's area. Simple linear interpolation with respect to height was used to calculate the four weather variable values surrounding the location of the desired point. The assumption that we can use only the closest corner was verified by calculating some example height index values using GRIB data. The difference in height indexes between the corners tended to vary less then 5 percent.

Once the isobaric index of the closest corner has been determined, the values at each corner for each weather variable, V_{ij} , can be interpolated. Using these four corner values it is possible to interpolate the value, V_{xy} , in the grid. This is accomplished

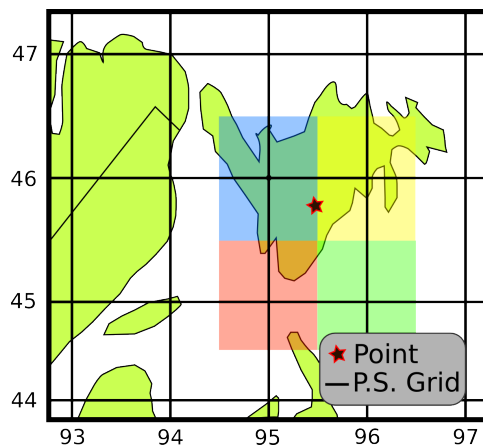


Figure 2.7: Regions in the Polar Stereographic Grid (This image is an illustration and not exactly topologically correct)

using the methodology described in figure 2.8 and in equation 2.1. The methodology described in equation 2.1 is essentially a bilinear interpolation that is commonly used in digital image processing [11].

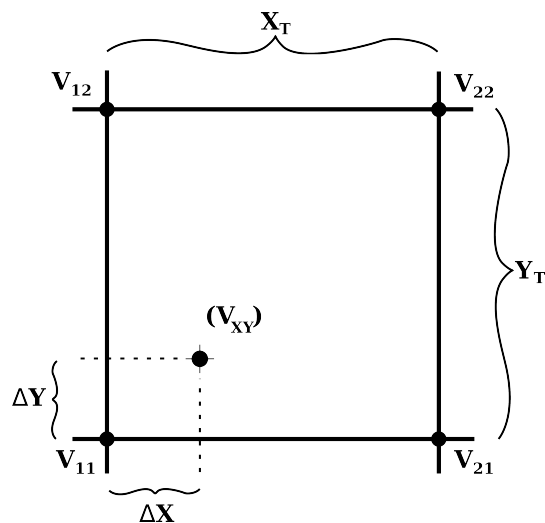


Figure 2.8: Interpolation Within a Grid Region

$$V_{XY} = \frac{V_{11} \Delta X}{X_T Y_T} \left(\frac{X_T}{\Delta X} + \frac{V_{21}}{V_{11}} - 1 \right) (1 - \Delta Y) + \frac{V_{12} \Delta X}{X_T Y_T} \left(\frac{X_T}{\Delta X} + \frac{V_{22}}{V_{12}} - 1 \right) (\Delta Y) \quad (2.1)$$

Using equation 2.1, the value at the point may be quickly interpolated based on the four boundary corner values.

Step 5 - Interpolate Time

The fifth step of this interpolation process is with respect to time. The spatial interpolation for each weather variable, described in steps 3 and 4, was performed twice. The time indices calculated in step two can be used to linearly interpolate these values. This results in one value per environmental variable.

Step 6 - Rotate Vectors

Most of the data interpolation takes place in the polar stereographic grids frame of reference. Scalar data is transferred between the polar stereographic grid frame of reference and the unprojected frame of reference. Transferring vectors between these two frame of reference is not trivial because the grids, as seen in figure 2.5, do not align. The wind data extracted from the weather data structure is broken into the W_X and W_Y components on the polar stereographic grid, as shown. The UAV model (section 2.2.4) accepts wind data that is in an unprojected frame of reference or resolved into latitude and longitude components, W_{Lat} and W_{Lon} .

Figure 2.9 illustrates the basic problem. The blue arrows, W_x , W_y , represent the vector components of the wind, W , with respect to the polar stereographic grid. The green arrow is the resultant of the two polar stereographic vector components. The desired frame of reference translation should not change the magnitude or direction of the resultant wind vector; it should break the resultant vector down into components, W_{Lat} and W_{Lon} ; these components are displayed in purple in figure 2.9.

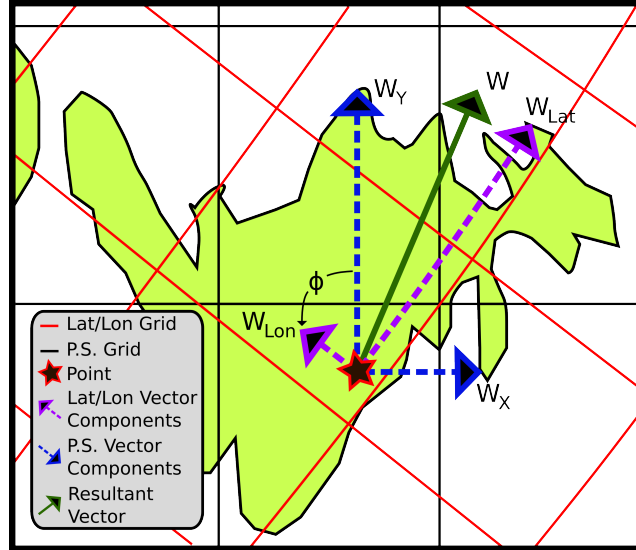


Figure 2.9: Pressure Levels above the Polar Stereographic Grid (This image is an illustration and not to scale)

In order to make this translation the angle ϕ between the two frames of reference is needed. Note that the (Lat, Lon) axes are not straight lines on the polar stereographic grid, as shown in figure 2.1, so the local angle ϕ is obtained using small perturbation linearization. Small perturbation linearization is accomplished by determining the polar stereographic grid location of a point where the longitude has been had a small change from the desired latitude & longitude. By comparing the desired point and the perturbed point in both frames of reference, the local angle, ϕ , can be calculated. The vector transform is:

$$\begin{aligned}
 W_{Lat} &= W_x \cos(\phi) + W_y \sin(\phi) \\
 W_{Lon} &= -W_x \sin(\phi) + W_y \cos(\phi)
 \end{aligned}
 \tag{2.2}$$

Example

To illustrate the mechanics of the weather data interpolation, an example will be generated. It is important to remember in this interpolation, accuracy is being

traded off with run time. The goal was to find the most accurate method that is still sufficiently fast. In this example the latitudinal component of wind is to be found at the Pippy Park Golf Course in St. John's, Newfoundland, Canada.

Location: Pippy Park Golf Course
Latitude: 47.585094 Degrees
Longitude: -52.757721 Degrees
Altitude: 500 Meters
Time: 7:30 GMT

In this example real weather values are used throughout the calculations. To verify that the weather algorithm works properly, the results of these manual calculations are compared to the result obtained from the software in chapter 3. Although the goal is to independently show that both the computer and hand calculations reach the same result, it is necessary to use some higher MATLAB[®] functions within these hand calculation. For this example the weather GRIB data set was acquired and processed into a MAT data file.

Step 1 - Obtain the grid indices

Using the MATLAB[®] `mfwdtran` function and some simple calculations the latitude and longitude values were converted to an index in the polar stereographic grid. In this example Pippy Park Golf Course corresponds to the grid location:

$$x = 115.9720 \quad y = 70.8340$$

Step 2 - Obtain the time indices

The time index is calculated for 7:30 am:

$$\begin{aligned} index_{time} &= \left(\frac{7.5}{6}\right) + 1 \\ &= 2.25 \end{aligned}$$

An index time value of 2.25 means the time is a quarter of the way between $T_2=6h$ and $T_3=12h$ (6h time steps).

Step3 - Obtain the height index

The closest corner needs to be selected. This is found by rounding both the x and y indices:

$$\begin{aligned} \text{round}(115.97) &= 116 \\ \text{round}(70.83) &= 71 \end{aligned}$$

The closest corner is at grid index (116,71).

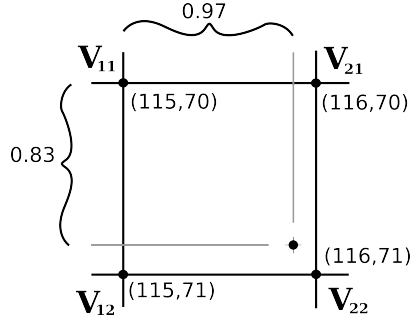
In step 5 of the algorithm it is necessary to combine results at different times. The two sets of calculation are shown side by side.

<u>Time at 6h (T_2)</u>	<u>Time at 12h (T_3)</u>
The value of 500 meters was found	The value of 500 meters was found
between pressure indices 4 and 5;	between pressure indices 4 and 5;
$H_4 = 370.93$ m	$H_4 = 385.08$ m
$H_5 = 548.70$ m	$H_5 = 562.74$ m

Linear interpolation is used to determine to location between index 4 and 5.

$$h_{index}(T_2) = 4 + \frac{(500-370.93)}{(548.70-370.93)} = 4.73 \quad \left| \quad h_{index}(T_3) = 4 + \frac{(500-385.08)}{(562.74-385.08)} = 4.65$$

Step 4 - Calculate weather at each time



In order to calculate the weather, the weather variables isobaric vector at each of the four surrounding corners the must be found. Each of these vectors must be interpolated based on the height index. This step is done for each of the desired weather variables.

Each value at 500m must be interpolated. Below are examples of the W_X at corner V_{11} . All other value have been filled in the table.

$$\begin{array}{l|l}
 W_X(T_2) = 3.66 + (5.53 - 3.66)(4.73 - 4) & W_X(T_3) = 3.65 + (4.41 - 3.65)(4.65 - 4) \\
 = 5.02 \text{ m/s} & = 4.14 \text{ m/s}
 \end{array}$$

	Corner V_{11}	Corner V_{21}	Corner V_{12}	Corner V_{22}
$T_2 W_X@I_5$	5.53	4.52	1.92	4.52
$T_2 W_X@I_4$	3.66	2.96	0.68	4.26
$T_2 W_X@500m$	5.02	4.10	1.59	4.45
$T_2 W_Y@5$	8.49	8.04	7.34	5.54
$T_2 W_Y@4$	7.88	7.18	6.98	6.18
$T_2 W_Y@500m$	8.33	7.81	7.24	5.71
$T_3 W_X@I_5$	4.41	5.31	6.11	5.91
$T_3 W_X@I_4$	3.65	4.35	4.65	5.25
$T_3 W_X@500m$	4.14	4.97	5.60	5.68
$T_3 W_Y@5$	6.88	8.08	7.08	7.48
$T_3 W_Y@4$	6.95	7.45	7.81	7.94
$T_3 W_Y@500m$	6.90	7.86	7.34	7.64

Using equation 2.1, $W_X(x,y)$ and $W_Y(x,y)$ are solved for both the X and Y component of the wind. Each of these are done at both T_2 and T_3 .

$ \begin{aligned} W_X(x, y) &= \\ (5.02)(0.97) \left(\frac{1}{0.97} + \frac{4.10}{5.02} - 1 \right) (1 - 0.83) + \\ (1.59)(0.97) \left(\frac{1}{0.97} + \frac{4.45}{1.59} - 1 \right) (0.83) \\ &= 4.32 \\ W_Y(x, y) &= \\ (8.33)(0.97) \left(\frac{1}{0.97} + \frac{7.81}{8.33} - 1 \right) (1 - 0.83) + \\ (7.24)(0.97) \left(\frac{1}{0.97} + \frac{5.71}{7.24} - 1 \right) (0.83) \\ &= 6.11 \end{aligned} $	$ \begin{aligned} W_X(x, y) &= \\ (4.14)(0.97) \left(\frac{1}{0.97} + \frac{4.97}{4.14} - 1 \right) (1 - 0.83) + \\ (5.60)(0.97) \left(\frac{1}{0.97} + \frac{5.68}{5.60} - 1 \right) (0.83) \\ &= 5.55 \\ W_Y(x, y) &= \\ (6.90)(0.97) \left(\frac{1}{0.97} + \frac{7.86}{6.90} - 1 \right) (1 - 0.83) + \\ (7.34)(0.97) \left(\frac{1}{0.97} + \frac{7.64}{7.34} - 1 \right) (0.83) \\ &= 7.66 \end{aligned} $
--	--

Step 5 - Linear interpolation over time

There are two values of each wind component, one at time $T_2 = 6h$ and another at $T_3 = 12h$. A linear time interpolation combines these two values using the time index calculated in step 2.

$$\begin{aligned}
W_X(x, y) &= 4.32 + (5.55 - 4.32)(2.25 - 2) \\
&= 4.63
\end{aligned}$$

$$\begin{aligned}
W_Y(x, y) &= 6.11 + (7.66 - 6.11)(2.25 - 2) \\
&= 6.50
\end{aligned}$$

Step 6 - Rotate vectors

The wind values calculated in step 5 are with respect to the polar stereographic grid. These two vectors need to be converted into a latitude/longitude frame of reference, using eqn. (2.2), as shown in Figure 2.6. The value of ϕ was calculated to be 58.27°

$$\begin{aligned}
W_{Lat} &= W_x \cos(\phi) + W_y \sin(\phi) = -0.529m/s \\
W_{Lon} &= -W_x \sin(\phi) + W_y \cos(\phi) = 7.959m/s
\end{aligned}$$

2.2.1.3 Graphical Overlay Product

When specifying the route of the aircraft it is important to have quick access to wind and temperature information at any altitude. Convenient methods of displaying important information have been developed by the meteorological community. Wind barbs are a preferred method of overlaying wind speeds and directions on a map. Wind speed information is given in knots, where 1 knot = 1.852 km/h. The wind direction is represented by the rotational direction of the wind barb. In Figure 2.10 a single wind direction toward the north east can be seen, along with the temperature indication, $35^{\circ}C$.

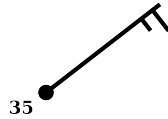


Figure 2.10: Single Wind Barb with Temperature

Different symbols on the barb represent different wind speeds. A short bar indicates a value of 5 knots, a long bar indicates 10 knots, and a triangle represents 50 knots. A summation of all bars and triangles indicates the wind speed at that location. Examples can be seen in figure 2.11; the barb in figure 2.10 corresponds to 15 knots.

Sometimes it is beneficial to display additional information. In this project it was determined that temperature should also be given. As seen in figure 2.10, temperature values are located directly beside each wind barb. Temperatures are given in Celsius.

2.2.2 Graphical Weather Charts

A requirement of this project is that the UAV operator has the ability to access all pertinent aviation weather data. Even though overlaying wind and temperature in-

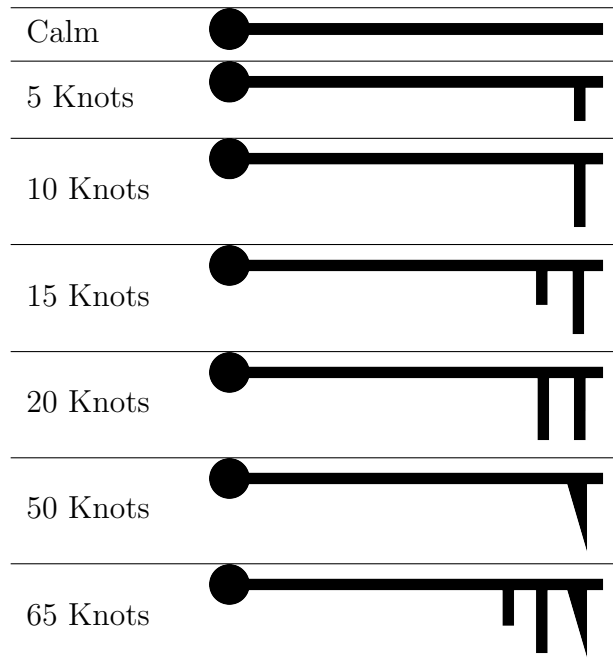


Figure 2.11: Wind Barb Values

formation is very important for the operator to understand the weather patterns, it does not give all the important information the operator might need to make confident decisions. There are many other products, such as satellite and radar imagery and turbulence and icing forecasts, that would greatly help the operator analyse possible risk to the aircraft. Fortunately, many useful products have been grouped onto NAV Canada’s website [12]. To ease the operator’s job of sorting through on-line information, all current pertinent weather analysis charts are accessible through EFOP’s graphical user interface. A list of these charts, with corresponding descriptions, can be found in table C.1 of Appendix C.

2.2.3 Map Processing

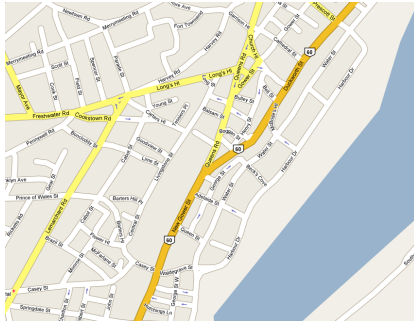
Raster and vector data are the two primary methods of storing images on a computer. Raster data is a digital image in pixel form. The data for a raster image is

stored in a grid where each grid value represents a pixel of the image. A picture from a digital camera is an example of a raster image. Raster data is stored in many different file types; some example file formats are JPEG (Joint Photographic Experts Group), TIFF (Tagged Image File Format), and PNG (Portable Network Graphics) files. Geo-spatial vector data differs from raster data in that vector data does not contain data for every pixel. Instead data is only recorded for each object, a point or line, in the dataset. CAD drawings and roads on a GPS navigation system are good examples of vector data. When looking at the same area, vector data files tend to be much smaller than raster data files. When vector data are displayed, they are actually shown as a raster image.

Figures 2.12(a), 2.12(b), and 2.12(c) contrast the difference between vector and raster data using examples from Google MapsTM. In figure 2.12(a) vector data describes the routes of roads and in figure 2.12(b) the raster satellite imagery can be seen for the same location. Figure 2.12(c) ties the vector data from figure 2.12(a) and raster data from figure 2.12(b) together and overlays them on one plot. Displaying both types of imagery data is desired in this project.

2.2.3.1 Vector Data Imagery

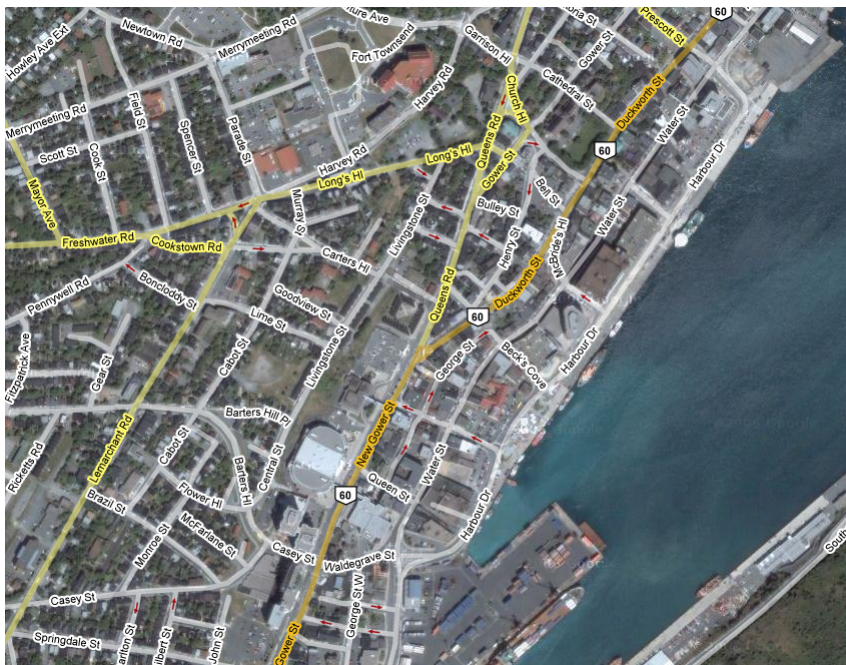
The vector data used in this project is obtained free from the GeoGratis website [13] which is maintained by Natural Resources Canada, a branch of the Canadian Government. Most significantly North American boundary data, basically an outline of North America, was acquired. Additional to the boundary shape data files, files describing transportation routes, rail routes, populated areas, and hydrology was downloaded. Shape data files were developed by Environmental Systems Research Institute (ESRI) and are used as an open specification or publicly available format



(a) Vector Data Imagery



(b) Raster Data Imagery



(c) Vector and Raster Overlay

Figure 2.12: An Example of Vector and Raster Data from Google Maps™

to store vector data [14]. Shape files contain geo-spatial vector data that is formatted for geographic information systems software. Fortunately MATLAB[®] is capable of reading this information.

2.2.3.2 Raster Data Imagery

The raster imagery used in this project was obtained from a set of satellite images that was produced by NASA [15]. The imagery was obtained by the MODIS satellite in 2004. An image was obtained for each month of the year. The June image was used in this software, since it offered the best contrast between the ocean, land, and ice. The original data set included both topographical as well as bathymetry (sea depth) data. Elevation data was stripped from the dataset since the development environment does not have a way of efficiently rendering this data. The image is saved in the GEODETIC map projection, revision WGS84. A low resolution sample of this satellite imagery can be seen in figure 2.13.

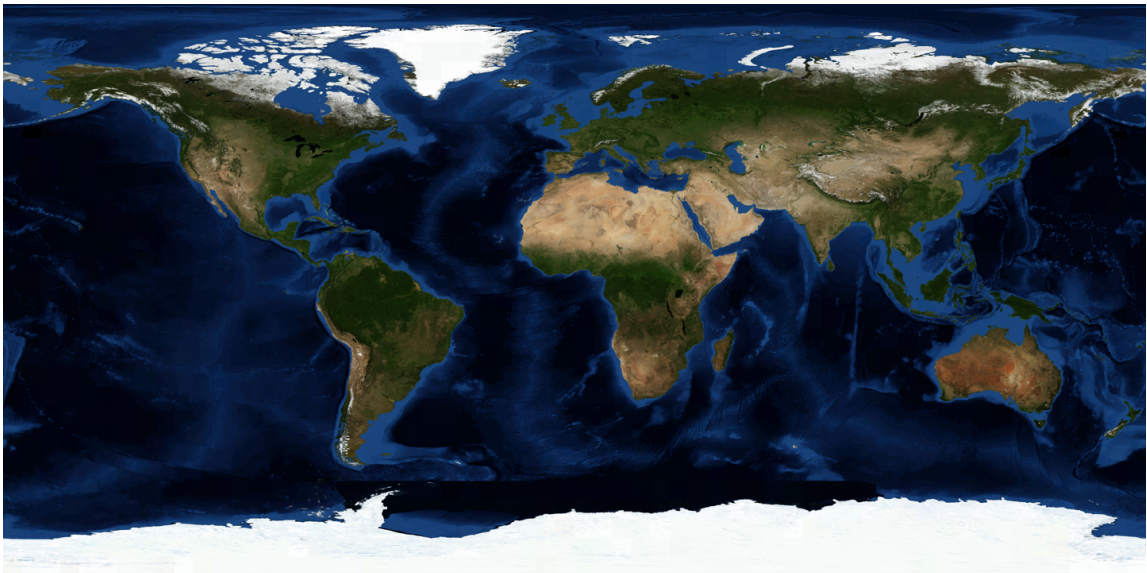


Figure 2.13: Low Resolution Satellite Imagery of Earth

Although the image shown in figure 2.13 is a low resolution rendering, the original image is very high resolution. The image is 86400 pixels wide and approximately 900 megabytes once being converted to a geotiff format, resulting in a resolution of approximately 500m per pixel at the equator. Managing and easily accessing desired data in a punctual manner is critical to ease of use for the software. Therefore, the high resolution satellite imagery was subdivided into a set of files. Each file represents a 3 degree by 3 degree portion of the satellite imagery, necessitating a total of 1800 images comprised of a grid 60 images horizontally and 30 images vertically. Figure 2.14 shows how these pieces are fit together to reconstruct an image.

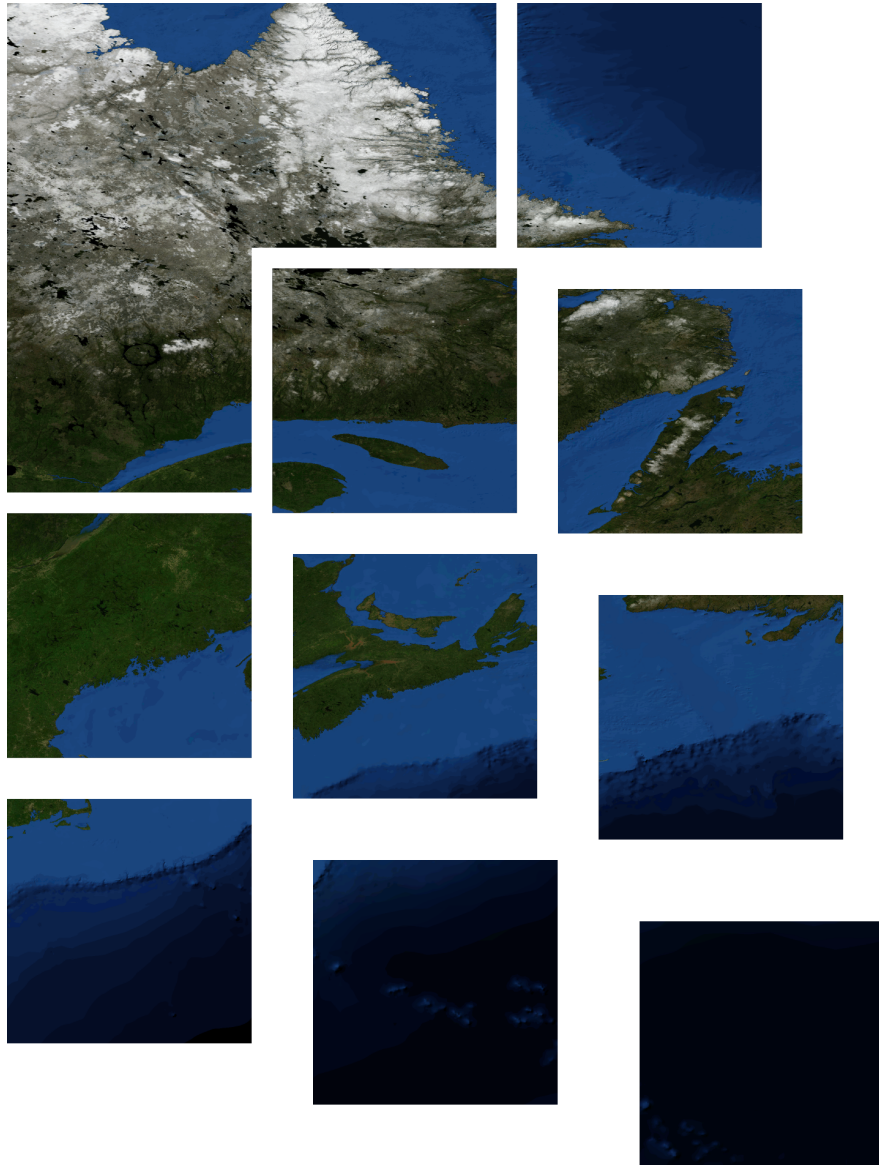


Figure 2.14: Reconstruction of Satellite Amaru Image of Earth

Quickly processing this imagery data requires executing the following four steps:

- Load desired image data into memory
- Stitch together images
- Crop this image
- Re-size the image

By performing these steps, a raster image can be generated for a precise area of the earth. These processed images can be used as a background in this software package.

2.2.4 Simulink[®] Models

In this project there are two distinct Simulink[®] simulations that have been developed, which are utilized during the planning and feasibility analysis and flight execution phases, respectively. The backbone of both of these simulations is the Aerosim Model produced by Unmanned Dynamics, LLC [1].

2.2.4.1 Aerosim Model

There are three complete models included in the Aerosim blockset. These models are basically the same except that one frame of reference is with respect to the aircraft's body, another frame of reference is geodetic with no effects from the variations in the earth's magnetic field, and the third model is also geodetic but includes the variation of the earth's magnetic field. The Aerosim body frame of reference model is used in this project.

This block is a complete six degree of freedom (DOF) or 12th order dynamic model. In this model there are three input classes and fifteen output classes. The three input classes are listed in table 2.1.

Table 2.1: Aircraft Model Inputs[1]

Input Variable	Variable Index	Variable Description
Controls [1:7,1]	C[1,1] = Flaps	Radian deflection of the flaps

Continued on next page

Input Variable	Variable Index	Variable Description
	C[2,1] = Elevator	Radian deflection of the elevator
	C[3,1] = Aileron	Radian deflection of the aileron
	C[4,1] = Rudder	Radian deflection of the rudder
	C[5,1] = Throttle	Fractional value of the throttle (0 to 1)
	C[6,1] = Mixture	Ratio value, air/fuel mixture
	C[7,1] = Ignition	1 \Rightarrow power-on or 0 \Rightarrow power-off.
Winds [1:3,1]	W[1,1] = W_N	Northerly background wind velocity, m/s
	W[2,1] = W_E	Easternly background wind velocity, m/s
	W[3,1] = W_D	Downward background wind velocity, m/s
Reset [1,1]	R[1,1] = Reset	Integrator reset flag (equals 0 or 1, all integrators reset on rising-edge)

The 15 output classes of the model are listed in table 2.2.

Table 2.2: Aircraft Model Outputs[1]

Output	Element Size	Element Description
States	[15x1]	The vector of aircraft states $[u \ v \ w \ p \ q \ r \ e_0 \ e_x \ e_y \ e_z \ Lat \ Lon \ Alt \ m_{fuel} \ \Omega_{eng}]^T$
Sensors	[18x1]	The vector of sensor data $[Lat \ Lon \ Alt \ V_N \ V_E \ V_D \ a_x \ a_y \ a_z \ p \ q \ r \ p_{stat} \ p_{dyn} \ OAT \ H_x \ H_y \ H_z]^T$
VelW	[3x1]	The 3x1 vector of aircraft velocity in wind axes $[V_a \ \beta \ \alpha]^T$ in $[m/s \ rad \ rad]$
Mach	[1x1]	The current aircraft Mach number Angular
Acc	[3x1]	The vector of body angular accelerations $[\dot{p} \ \dot{q} \ \dot{r}]^T$
Euler	[3x1]	Vector of the attitude of the aircraft given in Euler angles $[\phi \ \theta \ \psi]^T$, in radians
AeroCoeff	[6x1]	Vector of aerodynamic coefficients $[C_D \ C_Y \ C_L \ C_l \ C_m \ C_n]^T$, in rad^{-1}
PropCoeff	[3x1]	Vector of propeller coefficients $[J \ C_T \ C_P]^T$
EngCoeff	[5x1]	Vector of engine coefficients $[MAP \ \dot{m}_{air} \ \dot{m}_{fuel} \ BSFC \ P]^T$ given in $[kPa \ \frac{kg}{s} \ \frac{kg}{s} \ \frac{g}{(W*hr)} \ W]$
Mass	[1x1]	The current aircraft mass, in kg
ECEF	[3x1]	Vector of aircraft position in the Earth-centred, Earth-fixed frame $[X \ Y \ Z]^T$
MSL	[1x1]	The aircraft altitude above mean-sea-level, in m
AGL	[1x1]	The aircraft altitude above ground, in m

Continued on next page

Output	Element Size	Element Description
REarth	[1x1]	The Earth equivalent radius, at current aircraft location, in m
AConGnd	[1x1]	The aircraft-on-the-ground flag (0 if aircraft above ground, 1 if aircraft is on the ground)

2.2.4.2 Planning Simulink[®] Model

The first mode of operation for this software is the planning mode. In this mode the operator is able to plan and test the feasibility of different aircraft routes. Performing this flight feasibility analysis is accomplished by simulating all aspects of the proposed flight. Table 2.3 shows the different aspects and their models.

Table 2.3: Components of the Route Planning Model

Aspect	Method of Modelling
Aircraft Dynamics	Aerosim 6 DOF Model [1]
Aircraft Control	Simulink [®] Model, developed by RAVEN
Weather Conditions	Environment Canada Regional Model [16]
Aircraft Route	User-defined by navigation waypoints

Route Definition and Tracking

The aircraft control system was based on specifying a target latitude, longitude, altitude which are fed into the system; the UAV is programmed to proceed to that location. Figure 2.15 purposely exaggerates the tracking method the UAV uses when generating and travelling through its fine waypoints; this exaggeration more clearly shows this approach to trajectory planning. In this figure each black dot represents a user-defined rough waypoint, and the black line is the spline that fits through all these rough waypoints. A set of fine waypoints is generated at nearly equidistant intervals and each fine waypoint has a boundary region which is represented by the

grey circle; typically there will be substantially more fine waypoints than rough ones. The red line shows the UAV flight heading though the fine waypoints. In this method the UAV flies towards the n^{th} fine waypoint; once the UAV reaches the boundary line (the distance between the UAV and target location is less than the radius of the circle) the UAV will target the $(n + 1)^{st}$ fine waypoint. The radius of the boundary circle and the distance between the waypoints are used as tuning parameters for the tightness of the route tracking. This method was chosen primarily to diminish fuel waste during high winds. Fuel wasted is reduced because the UAV can navigate through the waypoints with an acceptable margin of error, without having to double back on itself to find a missed waypoint.

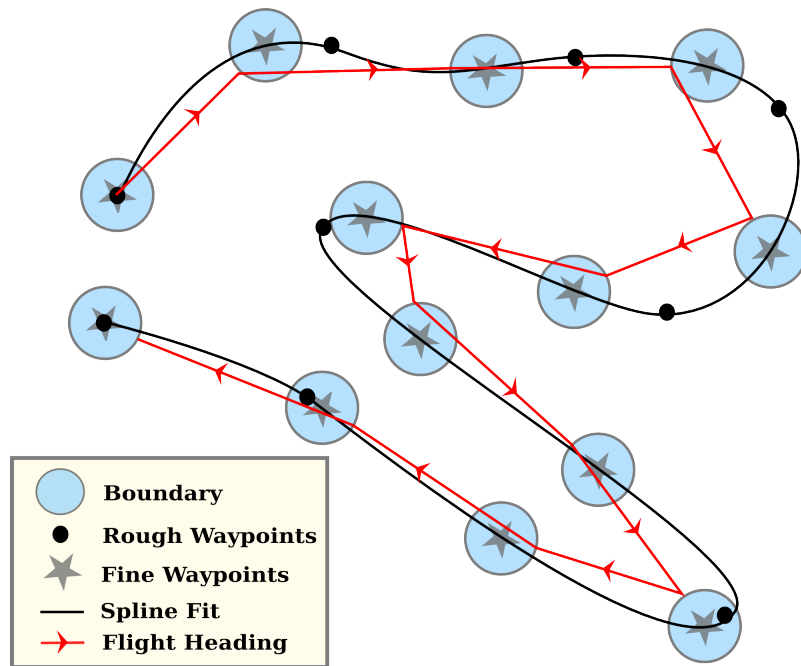


Figure 2.15: Aircraft Navigation from Fine Waypoint to Waypoint

The precision of route tracking is traded off by the fuel efficiency of the UAV trip. Long distance general operation is the goal of this project. In general, operational fuel efficiency is more important than exact location precision; therefore, the route tracking method illustrated in figure 2.15 was used.

Logging Data

By interfacing all of these components together a high fidelity simulation can be performed and the resulting data can be captured and stored. The model updates the UAV's position every 0.008 seconds. Capturing all model data would result in an excessively large data set, filled with mostly superfluous data. It was determined that relevant model data would be stored every 8 seconds of model runtime. The model variables that are stored are listed and described in table 2.4.

Table 2.4: Variables that are Logged During Simulation

Variable	Description
Time	The time of captured data
Aircraft Location	Actual Latitude, Longitude, and altitude
Aircraft Destination	Target latitude, longitude, and altitude
Weather Conditions	Wind and temperatures
Aircraft Fuel	Remaining Fuel

Model Status

The ability for the operator to see the status of the simulation is fundamental for the operator to understand the feasibility of the proposed flight. Hence, several important model variables are updated in the GUI while the simulation runs. These variables are listed and described below in table 2.5.

2.2.4.3 Execution Simulink[®] Model

The second mode of operation in this software package is called the execution mode. In this mode the operator is able to execute the previously planned mission. Since access to a real UAV is not possible during the development of this software package,

Table 2.5: Variables of the Route Planning Model

Variable	Description
Latitude	Latitude of UAV.
Longitude	Longitude of UAV.
Altitude	Altitude of UAV.
Remaining Distance	Remaining mission distance until simulation is complete.
Remaining Fuel	Mass of fuel remaining in the UAV.

the UAV aircraft model was used in its place. This model has been developed in a manner such that the actual UAV radio-frequency/satellite up link can be easily replaced by the model. There are some fundamental differences between the execution model and the feasibility/planning model. The weather forecast data is not used, since this will eventually be included by the real world situation; secondly, a number of rudimentary real-time course adjustment options are implemented. These may be refined and extended as real test experience is gained; that is beyond the scope of this project. The different route control options are described in table 2.6.

Table 2.6: Real-time UAV Route Control Options

Control Command	Method of Implementation
Go Home	The aircraft immediately goes back to the first waypoint
Hold	The UAV circles and waits
Change Course	A new course is entered and the UAV follows this route

In execution mode relevant mission data are also automatically stored as the mission unfolds. This data are the same as described in figure 2.4 except weather data is

not stored. Similar to the planning mode, all model status data is delivered to the GUI from the execution model, see figure 2.5.

2.2.4.4 GUI Software Flow

The flow of software execution is key when developing a new product. In this instance flow refers to the sequential paths followed by the software. The flow should be logical and reasonably intuitive. Since this project involves inventing original software, it has been possible to make logical development of this software a primary goal. Figure 2.16 details how the software execution proceeds. The software flow chart describes a typical software run. It is assumed that weather data is downloaded as needed and the flight trajectory is defined by the operator. In this flow diagram the operator plans the mission, determines mission feasibility and then executes the mission. It should be noted that this diagram details typical use of the software and is not intended to contain more advanced scenarios.

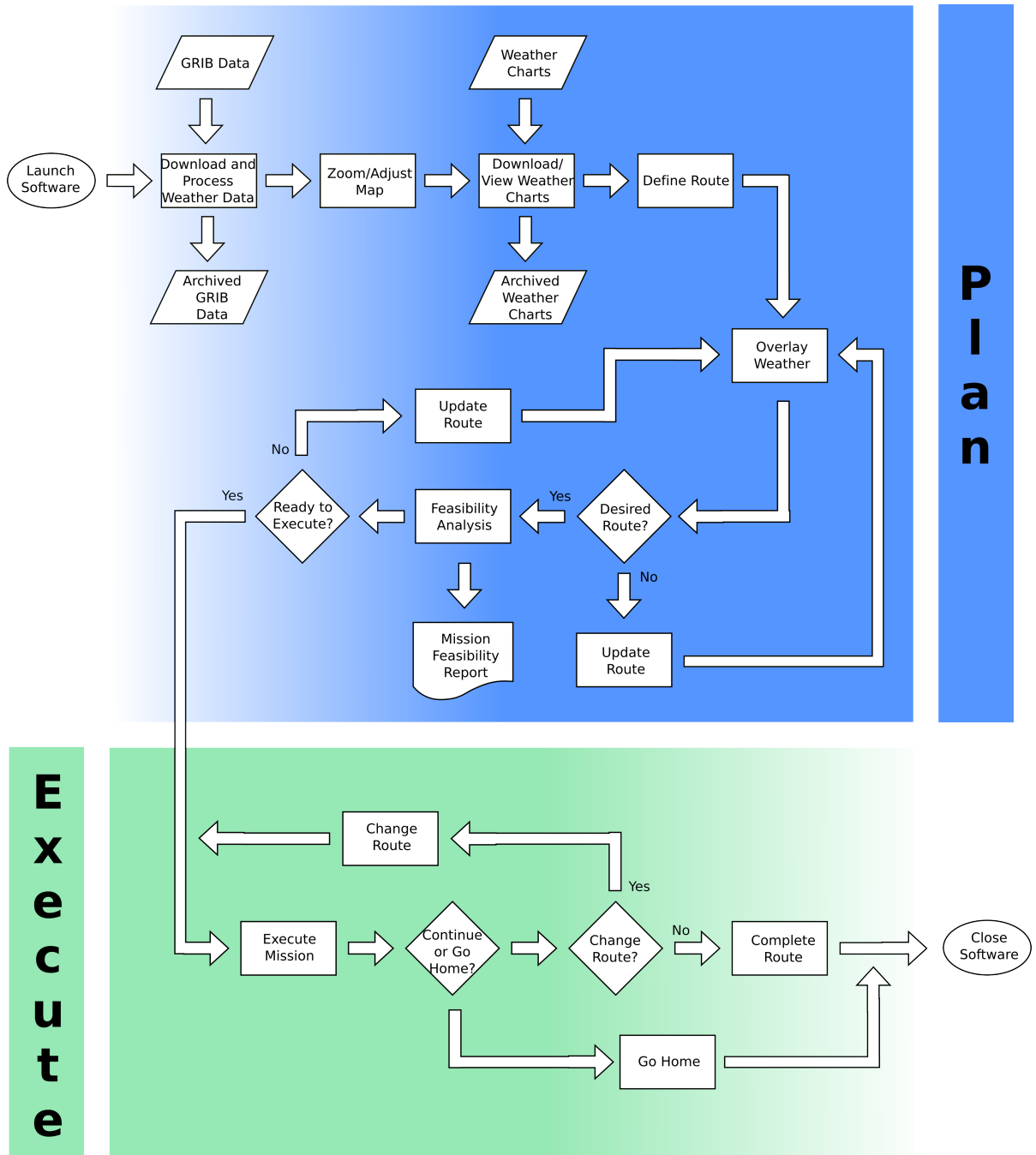


Figure 2.16: Software Flow Chart

Chapter 3

Implementation

This chapter will describe how the theory, data, and systems described in chapter two are integrated into the final functional product. Due to the size of this project it is not possible to document every line of code. As a result, time will be allotted to explaining how functions are used, how algorithms are implemented, and how data are acquired and processed. Towards the end of this chapter, the flow of EFOP operations will be described.

3.1 Numerical Model Weather Data

The software must be able to acquire the weather forecast data from remote servers. MATLAB[®] does not include an infrastructure for downloading data files. Fortunately MATLAB[®] does allow for system commands to be run from the MATLAB[®] command prompt. There are various open source projects that facilitate remote data acquisition. One such project is called curl. Curl is capable of acquiring files

via various file transfer protocols. It is also capable of downloading a series of files, providing the filenames have a definitive structure. Curl commands can be executed by having the curl executable file in MATLAB[®]'s working directory. An example of how curl is called can be seen in figure 3.1. Lines 8 to 10 show how MATLAB[®] calls the `curl.exe` routine.

```
1 cd curl;
2 for i = 1:size(names,2)
3     mkdir([temp_dir '\' names{i}]);
4     for ii = 1:size(times,2)
5         ulr_filename = ['CMC_reg_' names{i} '_ISBL_'...
6             var_pres '_ps60km_' dates '_' times{ii} '.grib'];
7         computer_filename = ['CMC_' names{i} '_#1.grib'];
8         [status,result] = system(['curl.exe ' ulr ulr_filename...
9             ' -o ' temp_dir '\' names{i} '\' times{ii} '\'...
10            computer_filename]);
11     end
12 end
13 cd ..;
```

Figure 3.1: Code used to Download the GRIB Data

As a precaution, all data that are downloaded are backed up in an archive directory; data are time/date stamped and stored in the `/archive/` directory.

Reading the numerical weather data into MATLAB[®] proved to be a complex task. GRIB data is highly structured and requires exact processing. Fortunately there was a project called `read_grib` [9] that provided a method to input GRIB data into MATLAB[®] memory. This function was not initially compatible with Environment Canada's GRIB 1 data, as this function was not designed to be compatible with all GRIB 1 data. After some investigation into the GRIB 1 structure [10] this problem was corrected. In figure 3.2 an example of a `read_grib` statement can be seen. The `-1`, following the string concatenation of the filename, means that all data objects in the GRIB file should be read into memory.

```

1 rawdata.time{iii}.temp{i} = read_grib([directory...
2 '\grib_data\TMP\P' index_time{iii} '\CMC_TMP_'...
3 index_temp{i} '.grib'],-1);

```

Figure 3.2: Code used to Read GRIB Data into Memory

3.1.1 Numerical Model Weather Data Processing

Once the weather data has been loaded into memory, it is rearranged into a more organized data structure. This structure is documented in section 2.2.1.1. The code that accomplishes this task is a combination of `for` loops and variable declarations. A sample of this code can be seen in figure 3.3.

```

1 for iii = 1:length(index_time)
2     time{iii}.grid{ninjtolatlong(ii,2),ninjtolatlong(ii,1)}.lat = ...
3     ninjtolatlong(ii,3);

```

Figure 3.3: Code used to Process the Weather Data in a Data Structure

3.1.2 Weather Data Interpolation

Implementation

Interpolation of the weather data is one of the most complex parts of this project. Several nested functions have been developed to accomplish this task. For the purpose of documenting the implementation of this methodology, the code will be approached as if it were one large function.

The weather interpolation method is a function that has both input and output. The inputs and outputs of this function are described in table 3.1.

Table 3.1: Input/Output Table

I/O	Name	Description
Input	latitude	Latitude of the interpolation point
Input	longitude	Longitude of the interpolation point
Input	altitude	Altitude of the interpolation point
Input	time	Time of interpolation
Input	m-struct	MATLAB [®] mapping structure of the grid
Input	grid	Weather data structure
Input	map-info	Data structure defining Environment Canada's polar stereographic grid
Output	temperature	Temperature at the grid point
Output	x-wind	Wind in the x or longitudinal direction
Output	y-wind	Wind in the y or latitudinal direction

The first step in this function is to translate the latitude and longitude into the grid's coordinate system. This is accomplished by using MATLAB[®]'s Mapping Toolbox[™]. The following `mfdtran` command allows the user to translate between different coordinate systems: `[x,y] = mfdtran(m_struct,lat,lon,[],'line');`

After some simple calculations the output of the file is the exact x and y coordinates on the grid. The value of x and y are not integers on the grid, they are precise points (e.g., 23.433, 54.123).

Once these values have been calculated, the next step is to get the precise time index. Since there is only one grid per six hours it is necessary to translate the hour to the hour index in the data structure. This can be accomplished by the command:

```
index_time = hour./6+1
```

This value is a double and will most likely contain a remainder after the decimal point. If this value is rounded both up and down, the two time indices that bracket

this point is determined. The next sections are calculated using the grid data from both the earlier and later time indices. The step to perform the interpolation between these two times is the last step in calculating the interpolated weather.

The vertical index corresponding to the particular altitude needs to be determined for both time indices. The coordinates of the grid points to perform the height search are determined by using the round function: `x=round(x.real)`; `y=round(y.real)`;

Once the closest grid coordinates are determined, the altitude values can be extracted from the grid data structure. A binary search algorithm[17] is applied to the data to determine the upper and lower indices that the desired altitude lays between. In figure 3.4 the binary search algorithm can be seen.

```
1 function [l,h] = search_sorted(height,h_vector)
2 low = 1;
3 high = size(h_vector,1);
4 mid = round((low + high)/2);
5 while(mid ~= high)
6     midValue = h_vector(mid,1);
7     if height < midValue
8         high = mid;
9     else
10        low = mid;
11    end
12    mid = round((low + high)/2);
13 end
```

Figure 3.4: Binary Search Algorithm used to Determine the Height Index

Once the upper and lower height indices are found, a linear interpolation is performed to determine exact location the desired altitude lies between these two isobaric levels. At this point the exact indices for the desired latitude, longitude, altitude, and time have been determined. Using this information, the weather data at each time can be calculated. The first step in calculating these values is to extract

the weather on each corner of the surface containing the point of interest, i.e., the weather parameters are interpolated at each corner based on the height index that was previously calculated. This code can be seen in figure 3.5

```

1  wb.tmp.ul = tmp.ul(h.low) + (h.real-h.low).*...
2      (tmp.ul(h.high)-tmp.ul(h.low));
3  wb.xwind.ul=xwind.ul(h.low)+(h.real-h.low).*...
4      (xwind.ul(h.high)-xwind.ul(h.low));
5  wb.ywind.ul=ywind.ul(h.low)+(h.real-h.low).*...
6      (ywind.ul(h.high)-ywind.ul(h.low));

```

Figure 3.5: Code used to Interpolate each Corner of Weather Block Based on Height Index

A weather value has now been determined at each corner and for bracketing times. The next step is to combine the corner information at each time into the interpolated weather parameters at the desired point for each of the time values. This interpolation is performed using the weather values at the corners and the x and y indices. The code to perform this interpolation is found in figure 3.6.

```

1  output.tmp = (1-(x.real-x.left)).*(wb.tmp.ul+(y.real-y.lower).* ...
2      (wb.tmp.ll-wb.tmp.ul)) + (x.real-x.left).*...
3      (wb.tmp.ur+(y.real-y.lower).*(wb.tmp.lr-wb.tmp.ur));
4  output.xwind = (1-(x.real-x.left)).*(wb.xwind.ul+(y.real-y.lower).*...
5      (wb.xwind.ll-wb.xwind.ul)) + (x.real-x.left).*...
6      (wb.xwind.ur+(y.real-y.lower).*(wb.xwind.lr-wb.xwind.ur));
7  output.ywind = (1-(x.real-x.left)).*(wb.ywind.ul+(y.real-y.lower).*...
8      (wb.ywind.ll-wb.ywind.ul)) + (x.real-x.left).*...
9      (wb.ywind.ur+(y.real-y.lower).*(wb.ywind.lr-wb.ywind.ur));

```

Figure 3.6: Code used to Combine Each Corner into the Weighted Point Value

Next the weather parameters are interpolated with respect to time. Using the weather parameters calculated for each time and the time index, a final linear interpolation can be performed. The code for this time interpolation can be seen in figure 3.7.

```

1 output.tmp = w1.tmp + (hour_index.real - hour_index.index1)...
2     .*(w2.tmp - w1.tmp);
3
4 output.xwind = w1.xwind + (hour_index.real - hour_index.index1)...
5     .*(w2.xwind - w1.xwind);
6
7 output.ywind = w1.ywind + (hour_index.real - hour_index.index1)...
8     .*(w2.ywind - w1.ywind);

```

Figure 3.7: Code used to Combine each Time Value into the Weighted Value

The last step in this weather interpolation algorithm is to rotate any vectors between the polar stereographic grid and the Lat/Lon grid. This is accomplished using small perturbation linearization. Some of the code to achieve the rotation can be seen in figure 3.8.

```

1 lat1 = lat; lon1 = lon; % Values of lat1 and lon1
2 lat2 = lat; lon2 = lon + 0.05; % Values of lat2 and lon2
3 x_grid1 = x_grid; y_grid1 = y_grid; % Grid Values of interest
4 [x_grid2,y_grid2] = get_indexes(lat2,lon2,m_struct,map_info);%Calc L2\l2
5 ang = atan2(y_grid2-y_grid1,x_grid2-x_grid1);
6 wx = w.xwind; wy = w.ywind;
7 lon_wind = wx*cos(ang)+wy*sin(ang);
8 lat_wind = -wx*sin(ang)+wy*cos(ang);

```

Figure 3.8: Code used to Rotate between Two Frames of Reference using Small Perturbation Linearization

Data Files

Due to the complexity of this interpolation process, the code spans multiple functions. Table 3.2 lists all the functions and a brief description of what they do.

Table 3.2: Weather Interpolation Files

Name	Description
interpolate-weather.m	Main function that calls all the other functions
initialize.m	Sets initial parameters
get-height-index.m	Returns the index of the height; <code>search-sorted</code> is called
get-indexes.m	Returns index of desired x and y. Coordinate system conversion is performed, <code>mwdtran</code> and <code>map-index</code> functions are called
get-time-indexes.m	Returns index of time
get-weather.m	Returns the interpolated weather. Functions <code>get-weather-boundary</code> and <code>interpolate-weather-boundary</code> are called
get-weather-boundary.m	Returns the weather
interpolate-time.m	Interpolates environmental variables with respect to time
interpolate-weather-boundary.m	Function that performs the 2D planar interpolation based on the four corners
linear-interpo.m	A linear interpolation function
map-index.m	Extracts relative information from the map
search-sorted.m	Binary search algorithm
rotatevector.m	Translates between the PS Grid vector components and the Lat/Lon vector components

3.1.3 Graphical Overlay Product

The proceeding weather interpolation function is utilized through the user interface. It allows the user to display both wind barbs and temperatures over the map for any time and altitude. In figure 3.9 the input dialogue box can be seen.

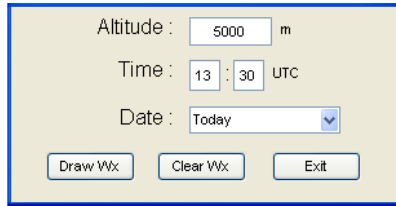


Figure 3.9: Graphical Weather Overlay Initialization Screenshot

Once the desired altitude and time have been input, the software gauges whether the longitude or latitude on the displayed map has a greater range. Whichever is greater has a linear space of 12 elements spanned over it. The distance between each element is calculated and this value is divided into the range of the shorter dimension. After rounding off to an integer, the resulting value gives the correct number of elements to produce an evenly-spaced grid. Such a grid can be generated automatically for any map. Once the grid points have been calculated the weather interpolation algorithm is used to extract the appropriate weather parameters from the weather data structure.

Once these values have been calculated it is possible to pass this information to a wind barb drawing function. This function uses basic geometry to draw the wind barb using line segments. The handle of every line is stored for easy removal. In figure 3.10 a screen-shot of the temperature and wind barb overlay can be seen.

3.2 Graphical Weather Charts

Having all available relevant information easily assessable for the operator is a key goal of this project. Retrieving pertinent weather charts is instrumental in the user-friendly operation of this software. Although MATLAB[®]'s GUIDE is robust, it did not present a simple way to display this information. Approximately 35 maps

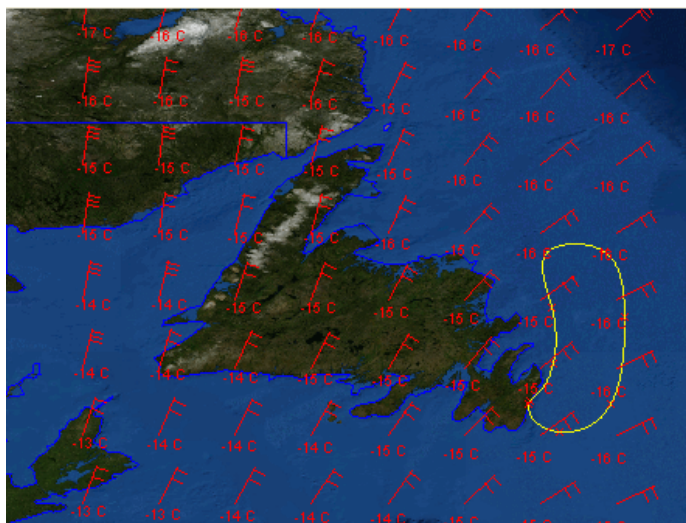


Figure 3.10: Graphical Weather Overlay Over Newfoundland

were considered significant, but not all operators would want to view all the maps. For this reason it is important for the user to be able to dynamically select the weather charts they want to view. The first step in achieving this is to allow the operator to select their desired charts by enabling check boxes. All the possible charts are organized into logical groups, i.e., upper air analysis, graphical forecast, etc. For example, graphical forecasting charts show forecasting information that is not available as vector data. Forecast turbulence, icing, freezing levels, cloud cover, and weather can be quickly viewed. Some of these charts are generated by an experienced forecaster. Although numerical models are convenient, sometimes the analysis by an experienced forecaster can be invaluable. Some charts are forecasted for different times, i.e., time1, time2, but exact chart update times cannot be determined from the downloaded files; hence, specific time values are not given. A screen-shot of this graphical form can be seen in figure 3.11.

Once the relevant charts have been selected, charts are retrieved on-line using curl. A backup of the charts is timestamped and stored in an archive directory for validation purposes. A screen shot of the downloading process can be seen in figure 3.12.

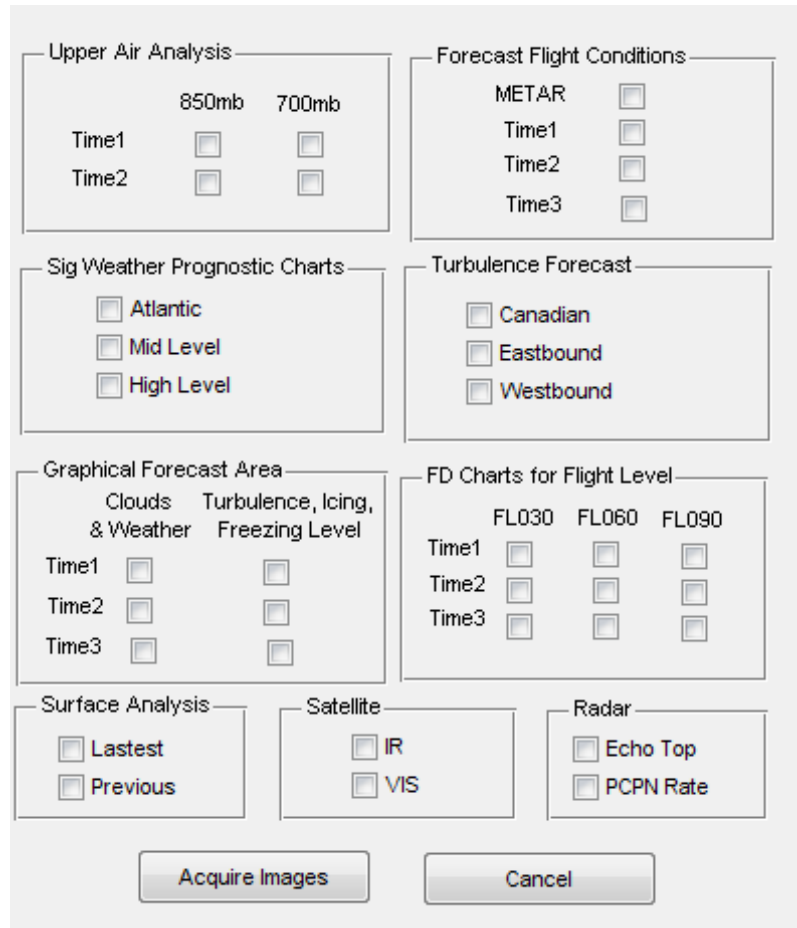


Figure 3.11: Method of Selecting Relevant Weather Charts

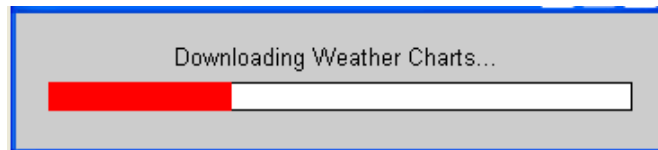


Figure 3.12: Downloading Weather Charts

Once this download is complete, all selected charts are loaded into memory and displayed in a tabbed figure window. The goal was to have each selected chart displayed in a different tab. The tabbed interface was difficult to achieve and required over 1000 lines of code. Some difficulties in accomplishing this resulted from not all charts being displayable using the same commands, since many charts had different file formats and/or color spaces. In screen-shot 3.13 the tabbed weather chart

display can be seen.

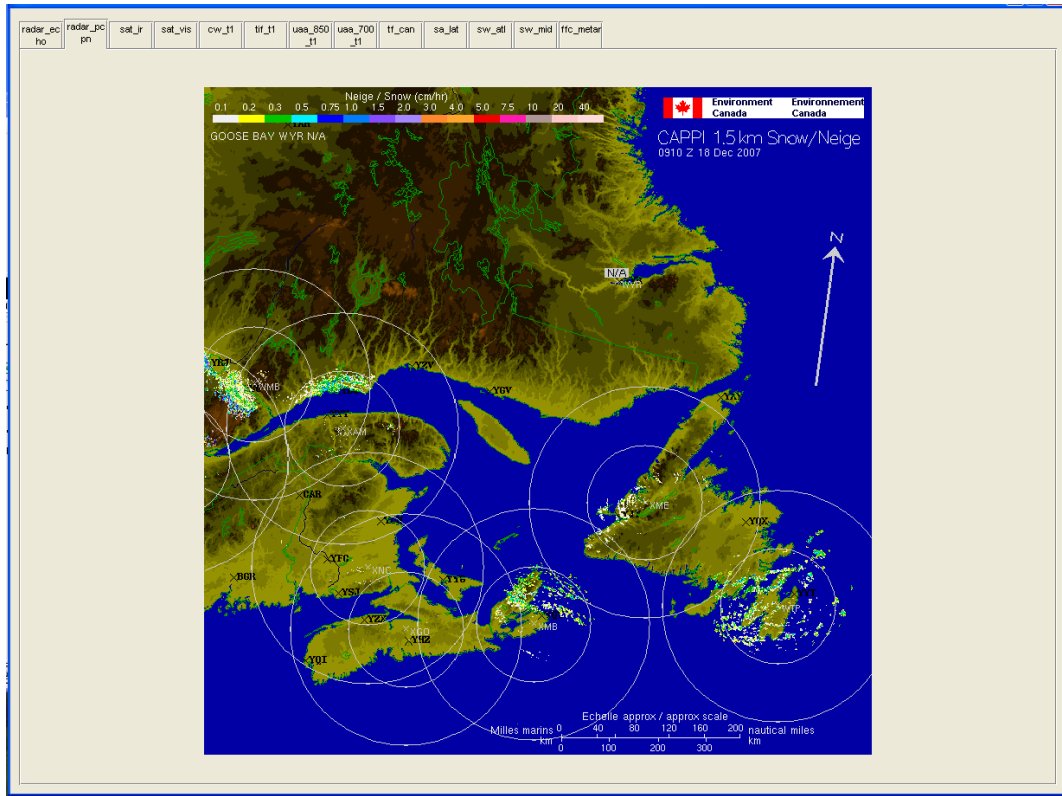


Figure 3.13: Method of Viewing Relevant Weather Charts

Another difficulty worth mentioning was encountered in displaying all the charts with adequate resolution. After some experimentation it was determined that there was no way to make the tab GUI continue to function if the window was re-sized. This problem was solved by having the software detect the screen resolution. A tabbed GUI figure is then drawn nearly the size of the screen and is centred. Since operator stations should always have high quality large display panels, this should not be a problem.

3.3 Map Processing

A critical component of this project is the ability to display maps and mapping information. Fortunately most maps can be displayed in MATLAB[®] through the use of the Mapping Toolbox[™].

3.3.1 Displaying a Map

Raster Data

Displaying a map is a two step process. First the raster data is drawn and then the vector data is drawn over it. The code to plot the raster data can be seen in figure 3.14.

```
1 [image,BBox,R] = image-latlon(lat-upleft,lat-bottomright...  
2     ,lon-upleft,lon-bottomright);  
3 ax = geoshow(image,R);
```

Figure 3.14: Code used to Display the Raster Map Data

This custom function can be further broken down. The `image_latlon` function, although custom, was designed to work seamlessly with the Mapping Toolbox[™]. The BBox (Bounding Box) and R (Reference Matrix) outputs are variables that the Mapping Toolbox[™] needs to function properly.

The first step in this process is to verify that inputs to the function are within the correct region of the globe. The latitude and longitude are translated into the map labelling scheme by the following command:

```
Xind = (((Xreal-Xreal0)*(Xind1-Xind0))/(Xreal1-Xreal0))+Xind0;
```

The shrinking factor of the image is calculated based on the ratio of the desired width by the total width. All map panels within the desired region are loaded into memory using function `geotiffread` (see Fig. 2.14). These files are quite large and very slow to process. Accordingly, each image is re-sized using the `imresize` command. Next the images are converted from an indexed color space to an RGB color space using the `ind2rgb` command. Even though this takes up more memory, this color space conversion was required since some of Image Processing Toolbox[™] functions are not capable of handling indexed images. The smaller images were stitched together by cascading the data accordingly. The last step in preparing this image was to crop the image. This was accomplished using the `imcrop` function which can be seen below in figure 3.15.

```

1 XMIN = round(shrinking_factor.*1441.*...
2           (lon_left-floor(lon_left)));
3 YMIN = round(shrinking_factor.*1441.*...
4           (lat_up-floor(lat_up)));
5 WIDTH = desired_width;
6
7 HEIGHT = round((desired_width).*...
8           ((lat_down-lat_up)./(lon_right-lon_left)));
9 rect = [XMIN YMIN WIDTH HEIGHT];
10 I2 = imcrop(image_all,rect);

```

Figure 3.15: Code used to Crop the Image

Although the image data has been properly prepared, the mapping data also needs to be calculated. The first variable `BBOX` stands for bounding box and is trivial to construct using the following code:

```
BBOX = [lon_upleft,lat_bottomright;lon_bottomright,lat_upleft];
```

The `R` variable was not as trivial to calculate, but fortunately MATLAB[®] contains a function to streamline this process called `makerepmat`. The code to create the reference matrix (`R`) can be seen in figure 3.16.

```

1 num_pix_lat = size(image,1);
2 num_pix_lon = size(image,2);
3 range_lat = lat_upleft - lat_bottomright;
4 range_lon = lon_bottomright - lon_upleft;
5 lat_rez = range_lat/num_pix_lat;
6 lon_rez = range_lon/num_pix_lon;
7 lat11 = lat_bottomright + lat_rez/2 ;
8 % Cell-center latitude corresponding to geoid(1,1)
9 lon11 = lon_upleft + lon_rez/2;
10 % Cell-center longitude corresponding to geoid(1,1)
11 dLat = lat_rez;
12 % From row to row moving north by one degree
13 dLon = lon_rez;
14 % From column to column moving east by one degree
15 R = makerefmat(lon11, lat11, dLon, dLat);

```

Figure 3.16: Code used to Create Reference Matrix R

Vector Data

The vector data is plotted using the `shaperead` function and the `geoshow` or `mapshow` commands. A bounding box is used so only shape data objects within the maps region will be loaded into memory. In figure 3.17 the code to plot a simple vector map is shown.

```

1 dire = pwd;
2 dire = [dire '\data\vector\'];
3 provinces = shaperead([directory 'prov_ab_p_geo83_e.shp'],...
4     'UseGeoCoords', true,...
5     'BoundingBox', [lon_ul lat_br; lon_br lat_ul]);
6 for i = 1:size(provinces,1)
7     x = [provinces(i,1).Lon];
8     y = [provinces(i,1).Lat];
9     mapshow(x,y)
10 end

```

Figure 3.17: Code used to Read and Plot Shape Data

Once both the raster and vector data has been loaded and plotted, the combined

image is displayed. A screen-shot of a representative map can be seen in figure 3.18. Take note of both the data sources; boundaries are drawn with vector data while satellite imagery is rendered from raster data.

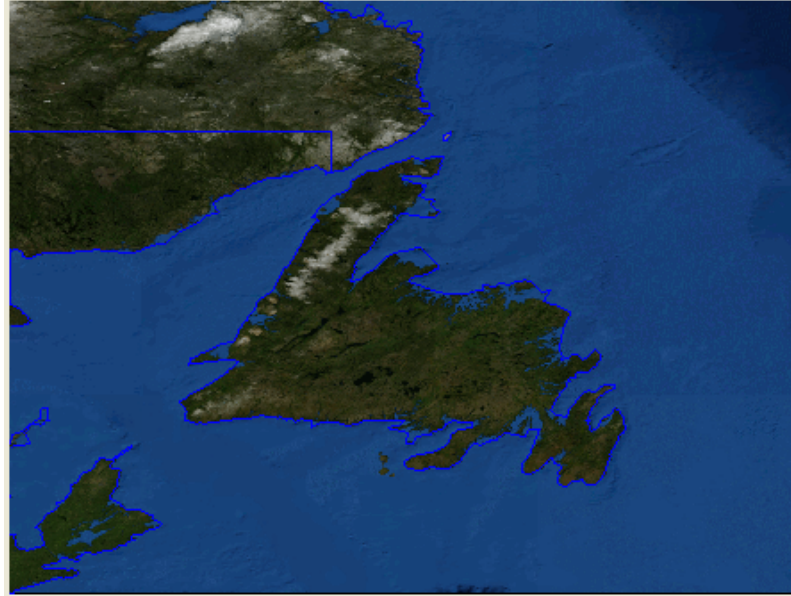


Figure 3.18: Screen-shot of a Map

3.3.1.1 Quick Zoom Tool

Processing and preparing map information can take a fair amount of time, depending on the computer's speed and the size of the region. Since guessing at the boundary latitudes and longitudes for the desired view can be trial and error, a quick zoom tool was developed. The quick zoom tool uses only vector data at first, allowing the operator to view the image before fully processing it. The quick zoom tool incorporates all of the functions mentioned in the previous section, 'Displaying a Map', as well as some for GUI development. A screen-shot of the quick zoom tool can be seen in figure 3.19.

The four text fields surrounding the plot in the quick zoom tool contain the latitude, longitude boundaries of the plot. If one of these values is changed the plot will

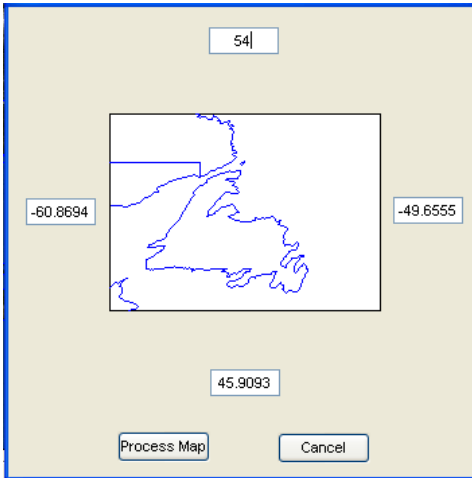


Figure 3.19: Screen-shot of the Quick Zoom Tool

automatically update. If the **Process Map** button is clicked then a map with these boundary's will be processed and loaded into the software. If the **Cancel** button is clicked, no changes will be made.

3.3.2 Simulink[®] Models

Since the Simulink[®] model was developed by another member of the RAVEN group, the main goal was not to refine it but to interface it with this software package. Interfacing a GUI with a Simulink[®] model is made possible through the use of a level 2 m-code s-function, a user definable Simulink[®] block. Different functionality is required in each of the software's two modes of operation; hence, two distinct s-functions and models were developed. The s-function is used to update aircraft status text displayed in the GUI, update UAV location marks on the GUI's maps, log data, update the targeted fine waypoint, and update weather information. The model seen in figure 3.20 illustrates how a level 2 s-function can be interfaced with a UAV model. When comparing the models for feasibility and execution mode, the Simulink models look very similar. The differences typically lay in coding of the s-function.

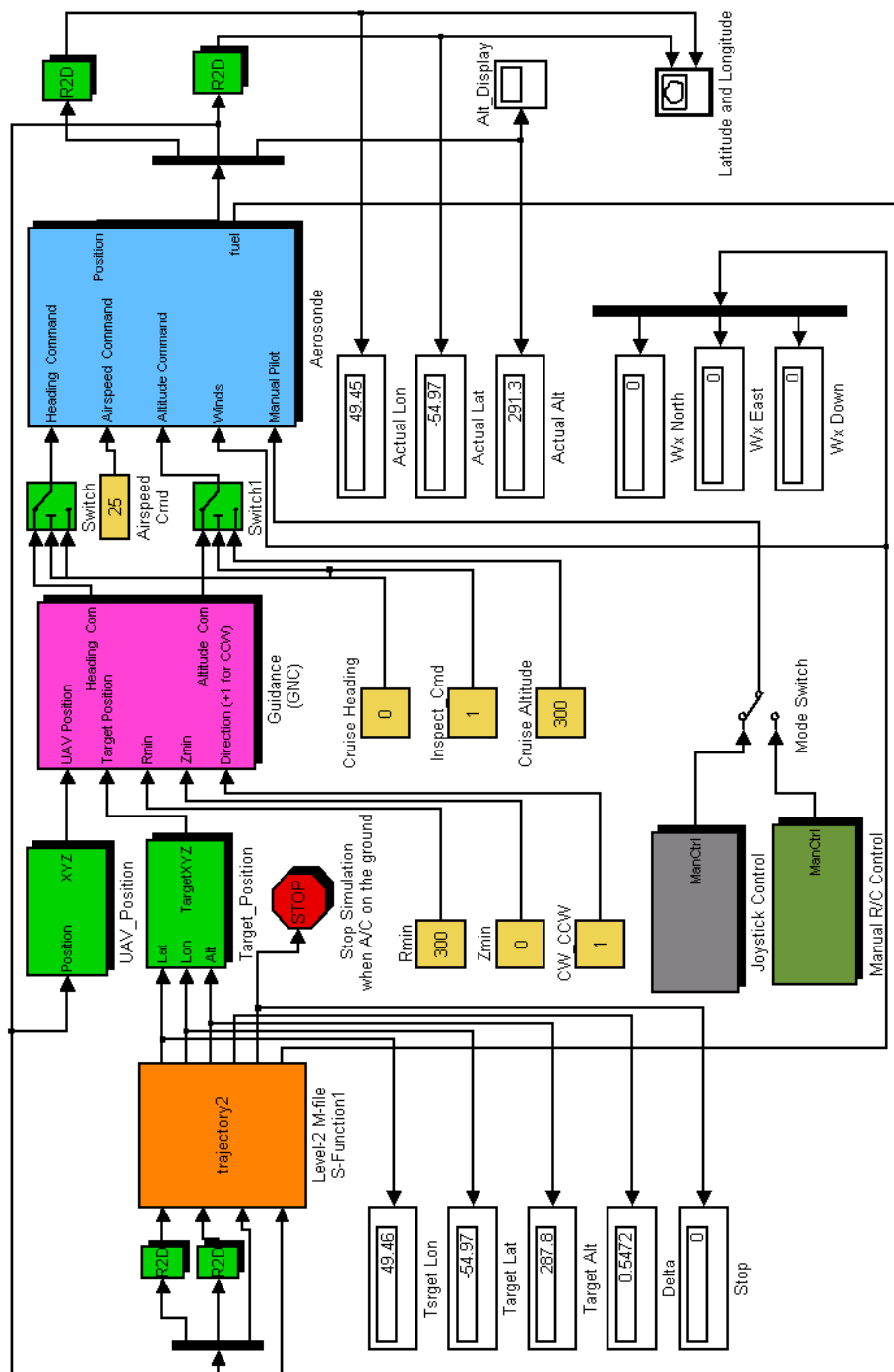


Figure 3.20: Planning or Executing Simulink® Model

3.3.2.1 Planning Simulink[®] Model

In the planning mode of operation there are four primary activities. These include retrieving and processing weather data, performing route tracking, logging relevant data, and updating model status.

Weather

The weather interpolation algorithm described previously can be called, since Simulink[®] s-functions are capable of running functions written in m-code. This function can be called by executing the following command:

```
[temp, x_wind, y_wind] = interpolate_weather(lat,lon,height,hours,...  
m_struct,time,map_info);
```

Executing this code is very computer resource intensive and is not necessary on every model integration step, since the weather model variables change at a very slow rate in comparison to the aircraft motion. Accordingly, executing the weather retrieval algorithms is only done at set intervals. Using this methodology, weather data can be recalculated every few seconds of model time, as opposed to every eight milliseconds as required for model dynamic update. This method of holding values in the s-functions memory until the next scheduled update can be used for both logging data and updating the GUI status.

Route Tracking

The route tracking software functions by determining the distance between the simulated UAV's current location and the next fine waypoint. Waypoints are updated

once a minimum distance is met (see figure 2.15), i.e., the UAV changes from aiming at waypoint k to waypoint $k + 1$ when it reaches the k^{th} boundary. The code in figure 3.21 shows how this is accomplished. Similar to the weather retrieval algorithms, checking for the minimum distance is not performed every model iteration. The minimum distance is checked every 1 second.

```
1 % Ok zone
2 distance_limit = 0.2;
3
4 % Create Legs
5 x=[lat,data1(step,1)];
6 y=[lon,data1(step,2)];
7
8 % Get Distance
9 [courses,distances] = legs(x,y);
10 distance = nm2km(distances);
11
12 % Determine if Distance is More
13 if distance < distance_limit %Next set
14     step = step +1;
15 end
```

Figure 3.21: Code used to Track a Route

Logging Data

It is desirable to only log data every 8 seconds, for similar reasons. The code listed in figure 3.22 shows how data is logged and stored in a public data structure. The data structure is made public, since this attribute allows the GUI to also access this data. Upon completion of the mission, this data is properly archived.

Model Status

Model status is updated every 4 seconds of model run time. This includes all relevant text fields in the user display, as well as updating the graphical mark of the UAV's

```

1 isgood = count8.*0.008./8;
2 if floor(isgood) - isgood == 0;
3     results.sample(isgood+1,1) = count8;
4     results.time(isgood+1,1) = (count8.*0.008)./8;
5     results.lat(isgood+1,1) = lat;
6     results.lon(isgood+1,1) = lon;
7     results.alt(isgood+1,1) = alt;
8     results.targetlat(isgood+1,1) = data1(step,2);
9     results.targetlon(isgood+1,1) = data1(step,1);
10    results.targetalt(isgood+1,1) = data1(step,3);
11    results.fuel(isgood+1,1) = fuel;
12    results.windx(isgood+1,1) = x_wind;
13    results.windy(isgood+1,1) = y_wind;
14 end

```

Figure 3.22: Code used to Log Data from the Simulink[®] Model

position on the map. The code in figure 3.23 illustrates how this update is performed using the `set` function to change properties of specific GUI objects .

3.3.3 Execution of the Simulink[®] Model

All activities that are used in the planning model are also used in the execution model except weather retrieval. The main difference between the planning mode and the execution mode is the execution mode is capable of handling on-the-fly course changes/corrections. These changes were implemented in MATLAB[®] using a switch/case logic statement. Due to the length of this code, pseudo-code describing this function is shown in figure 3.24.

3.4 Graphical User Interface

The graphical user interface was developed and setup using MATLAB[®]'s GUIDE. GUIDE uses both an M-file containing callbacks and an M-fig containing the GUI's

```

1  updatesec = 30;
2  isgood4 = count8.*0.008./updatesec;
3  if floor(isgood4) - isgood4 == 0;
4      sumtotaldistance = my_handles.sumtotaldistance;
5      distancetraveled = sumtotaldistance(step);
6      distanceleft = sumtotaldistance(length(sumtotaldistance))...
7          - distancetraveled;
8
9      % Update Strings
10     set(my_gui.textlat,'String',num2str(lat));
11     set(my_gui.textlon,'String',num2str(lon));
12     set(my_gui.textalt,'String',num2str(alt));
13     set(my_gui.textfuel,'String',num2str(fuel));
14     set(my_gui.textkm,'String',num2str(distanceleft));
15     % Update PLane and Draw Line in Axes 1
16     axes(my_handles.axes1);
17     hold on;
18     try
19         delete(my_handles.plane_handle);
20     end
21     my_handles.plane_handle = plotm(lat,lon,'rs',...
22         'LineWidth',2,...
23         'MarkerEdgeColor','k',...
24         'MarkerFaceColor','g',...
25         'MarkerSize',10);
26     drawnow;
27     % Update Altitude Plot
28     axes(my_handles.axes2);
29     hold on;
30     try
31         delete(my_handles.alt_handle);
32     end
33     my_handles.alt_handle = plot(distancetraveled,alt,'rs','LineWidth',2,...
34         'MarkerEdgeColor','k',...
35         'MarkerFaceColor','r',...
36         'MarkerSize',10);
37     drawnow;
38 end

```

Figure 3.23: Code used to Read and Plot Shape Data

```
1 global my_gui;  
2 global my_handles;  
3 my_gui.home = [my_handles.data.y(1) my_handles.data.x(1)...  
4               my_handles.data.z(1)];  
5 my_gui.case_exe = 3;
```

Figure 3.24: Code used to Change Flight Mode of Aircraft

objects. The GUIDE development environment can be seen in figure 3.25. By using the GUIDE development environment and strategically filling in callback routines, flow, and content, functionality can be implemented using this software. Additionally the basic framework for the EFOP GUI can be seen in figure 3.25. The top axes are for the latitude and longitude map while the lower axes are for the altitude plot. Surround the axes there are buttons that have various functions. A menu bar was also constructed using GUIDE.

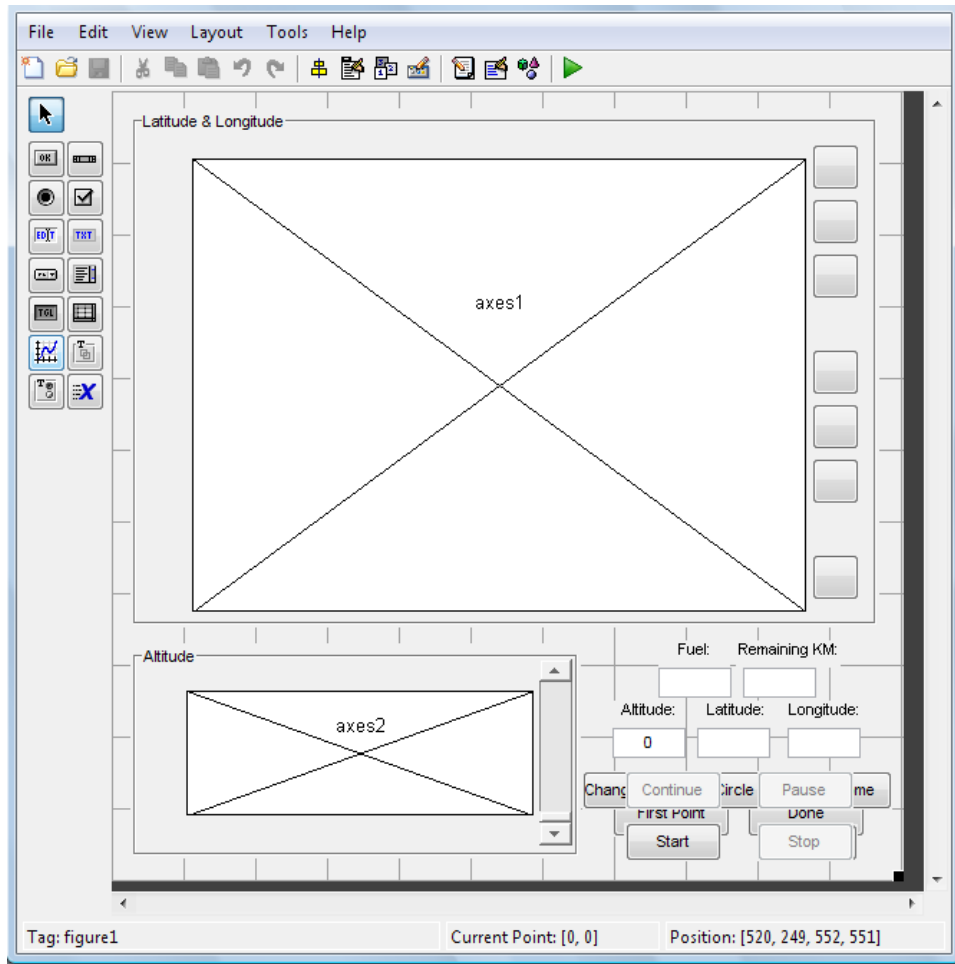


Figure 3.25: Screen-shot of EFOP software GUI in MATLAB's GUIDE Development Environment (Some GUI elements overlap, since their visibility is toggled on and off as desired)

Chapter 4

Presentation of Using the System for Planning

This chapter illustrates the planning and feasibility functions of the EFOP software package. The operator can easily process weather data, define the UAV's trajectory, and run a simulated flight. These steps are further detailed in the following sections.

4.1 Initialize

The weather model may require updating once the software package has been launched. The update is accomplished by clicking on the **Initialize** menu and then the **Get Weather Data** item. Downloading and processing of the weather data is required daily and can be quite time consuming. In order to prevent multiple data downloads, an **alter** window will indicate if the weather data is current. The software package can acquire current weather data, load previously acquired weather

data, or proceed without weather data. In the latter case, the simulation is executed with all weather parameters set to 0.

4.2 Planning

The next step requires the operator to plan the route for the mission. By selecting the `Define Trajectory` item in the `Initialize` drop-down menu, a map is displayed in the GUI. Before planning the trajectory, various factors must be taken into account before selecting the final set of waypoints. Weather charts should be reviewed by clicking on the `Display Wx Chart(s)` item located in the `Initialize` drop down menu. Environment Canada's numerical model weather data can be overlaid on the map by selecting the `Wx` button.

Planning the UAV's trajectory is accomplished by placing rough waypoints. Placing a waypoint requires three steps. The desired altitude is set on the altitude slider, the `First Point` or `Next Point` button is pressed, and the location of the waypoint is selected via a cross hair on the `Latitude & Longitude` map. The steps are repeated until all waypoints are chosen and finally the `Done` button is selected. Once the initial trajectory has been selected, it is possible to modify the trajectory. The trajectory is modified by using the `add waypoint`, `remove waypoint` and `change altitude` buttons. Additional details regarding trajectory planning are provided in Appendix A. To prevent repetitive daily trajectory planning, trajectories can be saved and loaded from the file menu.

A short but fully featured trajectory has been planned to illustrate the operation of this software package. Table 4.1 lists the latitudes, longitudes, and altitudes that define this trajectory. Figures 4.1 and 4.2 display maps of this trajectory.

Table 4.1: Rough Waypoints for an Example Mission

Latitude	Longitude	Altitude (m)
49.4880	-55.1216	75
49.3762	-55.1288	275
49.3700	-55.0331	325
49.4218	-54.9399	250
49.4860	-55.0068	325
49.5750	-54.8992	100

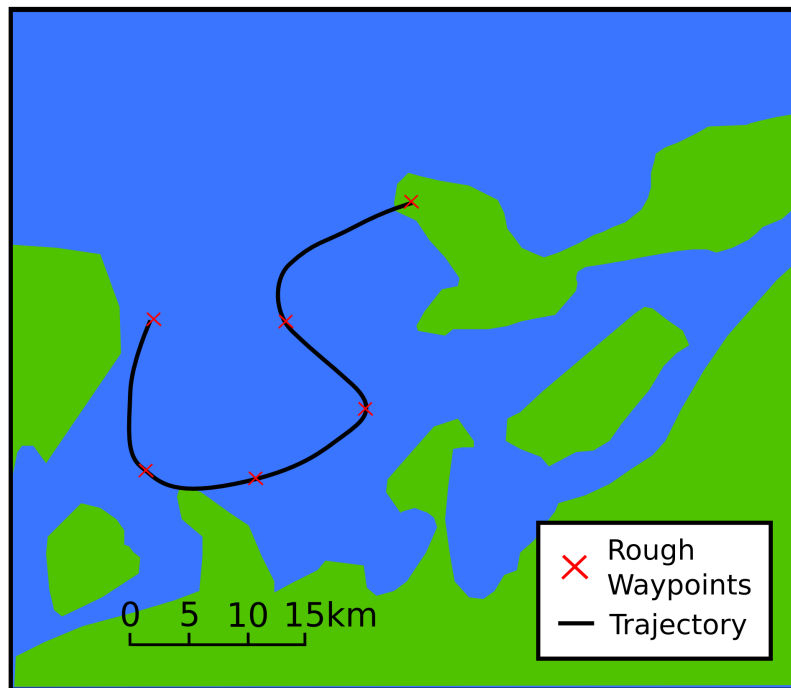


Figure 4.1: Plot of Planned Latitude/Longitude while in Planning Mode

4.3 Simulate UAV Flight

With a flight trajectory defined and optional weather data present, a feasibility simulation can be initiated. By selecting the `Run Simulation` item under the

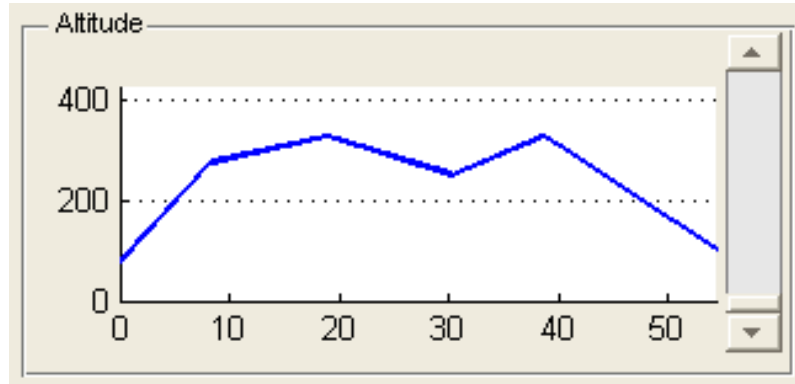


Figure 4.2: Plot of Planned Altitude while in Planning Mode

Feasibility menu, the software enters feasibility mode. Clicking the **Start** button launches the simulation, which will run until completion if left untouched. The simulation can be cancelled or paused by clicking the **Stop** and **Pause** buttons respectively. Numerical data are displayed in the GUI and graphical location indicators are shown in both the **Latitude & Longitude** and **Altitude** axes. Upon completion of the simulation, results are displayed. These results can be saved and loaded via the file menu. A screen-capture of the EFOP system, while determining the feasibility of a mission, can be seen in figure 4.3. In this figure the red and green UAV markers indicate the UAV altitude and latitude & longitude position respectively, at one time in the simulation. In the altitude plot the horizontal axis is distance traveled, measured in kilometers, and the vertical axis is the altitude measured in meters.

4.4 UAV Simulation Results

The simulation results are extracted from the time series data generated by the simulation. In table 4.2 various attributes of the mission simulation are displayed. In this table it is possible to see some of the differences between a simulation with weather

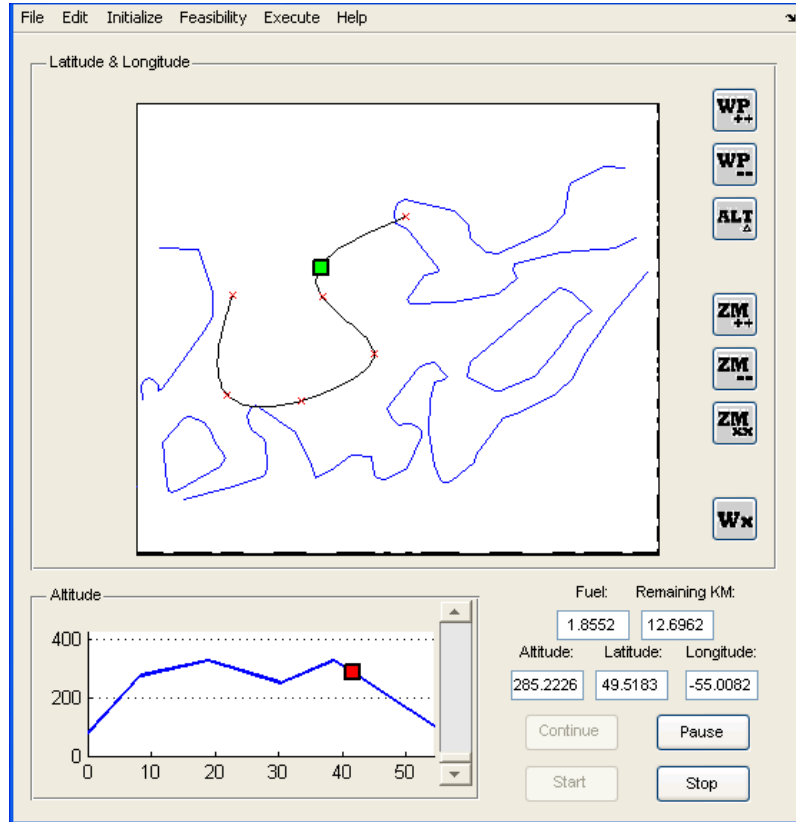


Figure 4.3: Screen-capture of EFOP while in Planning Mode

(with WX) and a simulation without weather (w/o WX). By utilizing weather data in the model, the fuel consumption was increased by 18% and elapsed mission time was increased by 25%. After the table, in figure 4.4 and 4.5, the latitude/longitude plot and the altitude plots are respectively shown.

Table 4.2: Attributes of the Simulations

	Without Weather	With Weather
Fuel Consumed	0.2038 kg	0.2414 kg
Elapsed Time	6251 s	7852 s
Distance Travelled	63.68 km	65.60 km

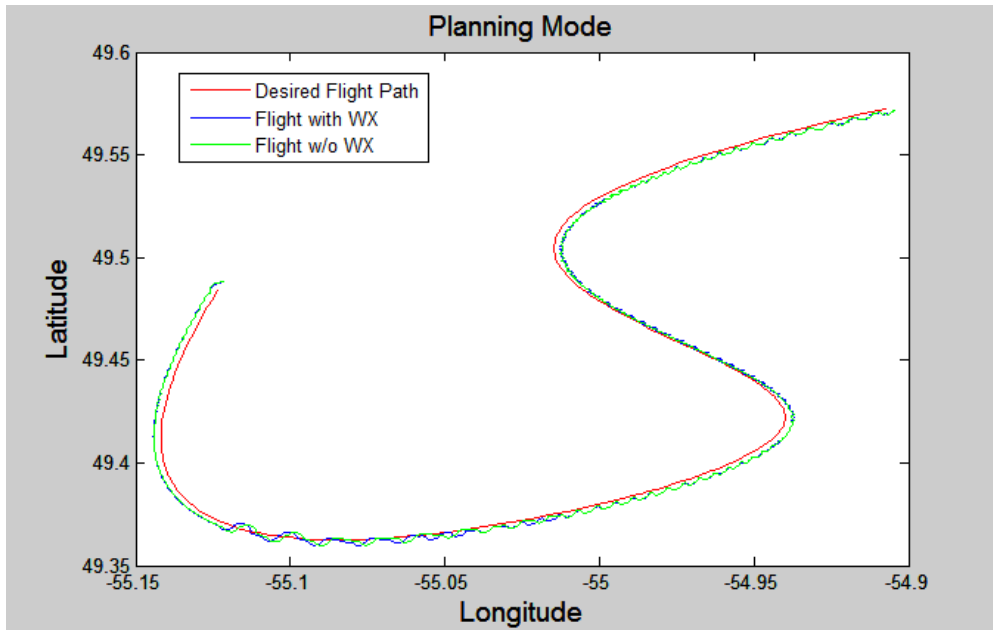


Figure 4.4: Plot of Logged Lat/Lon while in Planning Mode

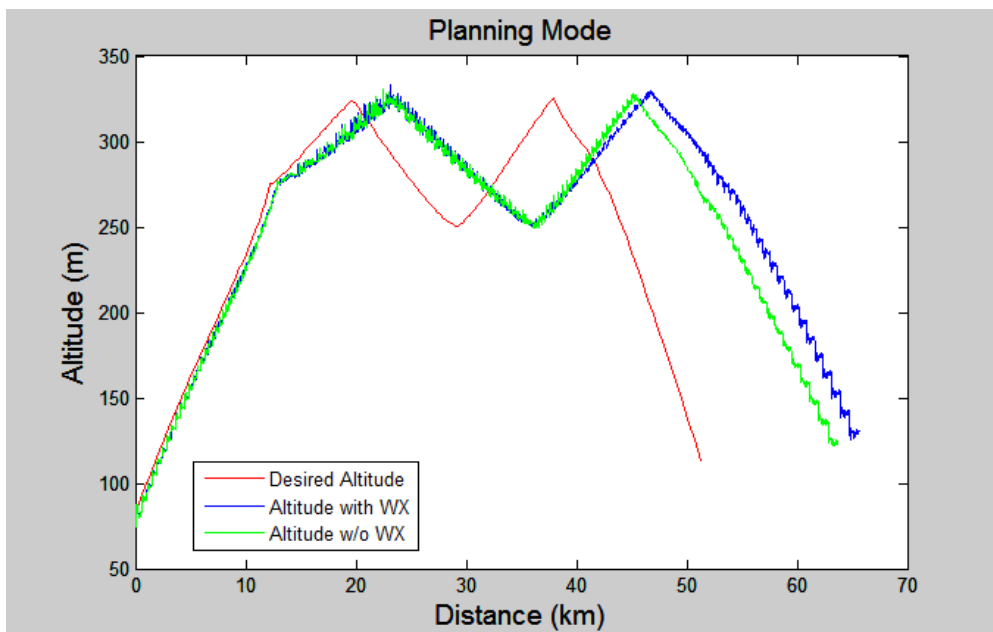


Figure 4.5: Plot of Logged Altitude while in Planning Mode

In figures 4.4 and 4.5 the boundary radius, see figure 2.15, of the waypoint tracking algorithm is 0.2 km. The small scale flight anomalies produced in figures 4.4 and 4.5 are a product of the control system design. The control system should be refined to produce a smoother flight.

Chapter 5

Presentation of Using the System for Execution

When executing a UAV mission, the operator is able to control the UAV in real time. Having direct control over the UAV allows the operator to revise the flight trajectory mid-flight, to order the UAV to hold near a significant event, or to immediately return the UAV to base. In testing this software package, the UAV being controlled is, unfortunately, only a model.

Executing a mission can be started by selecting the **Execute Mission** item in the **Execute** menu. The **Start** button can be used to start the simulation. Real time changes are accomplished by clicking either the **Change Route**, **Hold**, or **Go Home** buttons. This section of the project was completed as a proof of concept and will require significant future development for use with an actual UAV. This development will entail a coupling of the UAV's hardware with the EFOP software package via a wireless link.

Figure 5.1 graphically displays the data logged from a sample mission. For the purpose of illustrating how mission execution functions, a **Go Home** command was called before the UAV finished its mission. The flight trajectory used is the same flight trajectory described in table 4.1, figure 4.1 and figure 4.2.

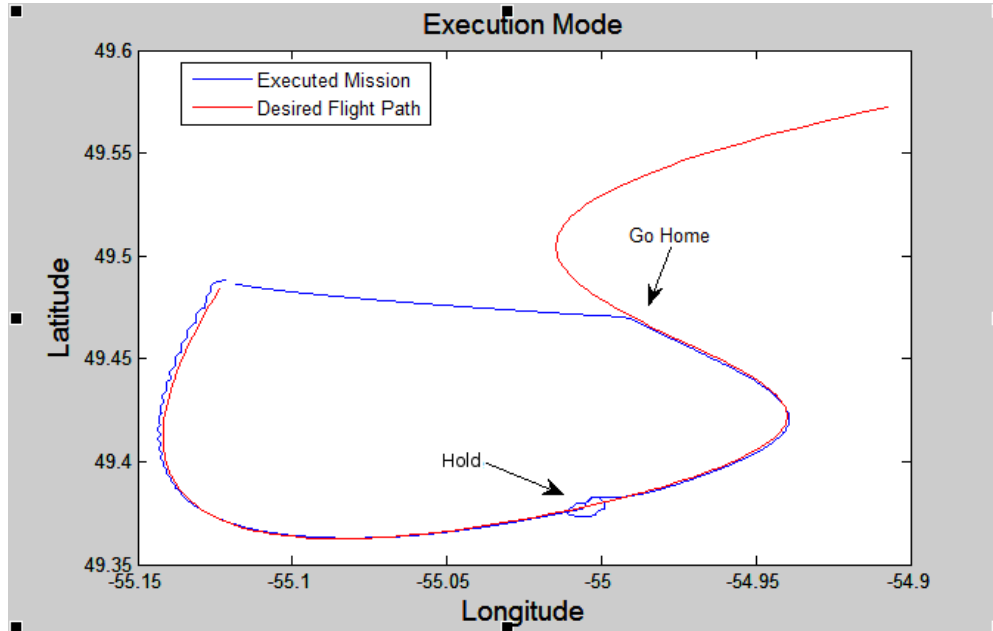


Figure 5.1: Plot of a Logged Data while in Execution Mode

In figure 5.1 the boundary radius, see figure 2.15, of the waypoint tracking algorithm is 0.3 km. When compared to figures 4.4 and 4.5 small scale anomalies were reduced by slightly increasing the boundary radius.

Chapter 6

Contributions, Future Work and Conclusion

6.1 Contributions

The projects' main contribution to current technology is not in the form of a revolutionary new methodology, but instead a compilation of ideas, software, and information to help solve a very specific and important problem. A copious amount of information has been gathered, sorted, and assimilated from various technologies. All of these components are integrated into the EFOP software package designed for operators to extract relevant information efficiently. Operators are able to easily run complex simulations providing mission feasibility projections and later execute these missions. The functionality of the overall software package, as well as some of its key functions, are this project's main contributions.

6.2 Future Work

Although this project software is fully functional, it is important to note that it is in a relatively early stage of developmental. As the software is used by operators, beneficial modifications and additional functions will be identified and incorporated. Both usability and ease-of-use will improve as the software is further developed. Several suggestions of possible improvements are listed below. Although few, some of these improvements relate to known software bugs.

- **Improve control system** - Small scale flight anomalies are a product of the control system. Refinements to produce a more robust control system would result in a smoother flight.
- **Decrease runtime duration** - Although EFOP was developed with speed in mind, unfortunately MATLAB[®] can be quite slow. This software runs fast enough to be marginally functional but improvements should be made. These include directly compiling the UAV simulation model to C code or re-coding the S-function/weather interpolation code in C. Depending on the application, compiling MATLAB[®] code can sometimes yield increased performance [18].
- **Fast Jpeg image stitching and processing through the use of C code** - Unfortunately there is a bug when dealing with raster maps in MATLAB[®], so Matlab would intermittently run out of memory while processing the raster image. After some analysis, it is suspected this bug is in MATLAB[®] routines. This should be corrected to ensure robust operability.
- **Take off and landing modes of operation** - This additional feature is required if the software is going to be used in real-time execution with UAVs. This code would allow for easy take off and landing. Ideally landing/take off routines would be assisted by an Instrument landing system (ILS).
- **Camera control, radar control, and remote sensor integration into the software package** - This new feature would allow the operator to control the aircrafts' sensors, make course correction based on radar, and inspect and record any event of interest using a camera.

- **HLA Interface** - This is an architecture that allows different distributed computer simulation systems to easily communicate with each other. There are HLA MATLAB[®] compatibility barriers.

6.3 Conclusion

In this project a unique synthesis of technologies, information and software was used to construct a user friendly software package for mission planning and execution. The EFOP software package aspires to function as a complete solution for deploying unmanned aerial vehicles. Most relevant data is archivable for easy reproduction of mission circumstances for interesting case studies. A majority of the mission feasibility features are enabled by using integrated nationwide numerical weather models. A complete weather monitoring solution is provided by making overlay and graphical charts easily accessible to the operator. Rich customizable mapping allows the operator to display additional relevant graphical information such as hydrology, transportation routes, and populated areas. Mission feasibility planning can be evaluated and, when finalized, seamlessly rolled into an operational flight. Control during real-time flights is easily facilitated through the user interface, as operators have the ability to modify flight routes on the fly. By comparing missions both with and without weather information, it was determined that meteorological information can have a very significant impact on the mission planning duration and fuel consumption. Although this software is fully functional, it is still in a developmental stage. As the RAVEN project grows and matures, this software package will too, eventually offering the operator a complete UAV mission planning and execution package.

Bibliography

- [1] Unmanned Dynamics. AeroSim, aeronautical simulation blockset, version 1.2, User's Guide. http://www.u-dynamics.com/aerosim/aerosim_ug.pdf, August 2009.
- [2] Environment Canada. Technical Grid Specifications. http://www.weatheroffice.gc.ca/grib/CMC_GRIB_Technical_Grid_Specifications_e.html#low_res_reg, August 2009.
- [3] Siu O'Young P. Hubbard. RAVEN: A maritime surveillance project using small UAV. *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, 2007.
- [4] The MathWorks. MATLAB - The Language of Technical Computing. <http://www.mathworks.com/products/matlab/>, August 2009.
- [5] The MathWorks. Simulink - Simulation and Model-Based Design. <http://www.mathworks.com/products/simulink/>, August 2009.
- [6] Environment Canada. Environment Canada. <http://www.ec.gc.ca/>, August 2009.
- [7] Environment Canada. Low-resolution CMC GRIB Database. http://www.weatheroffice.gc.ca/grib/Low-resolution_GRIB_e.html, August 2009.

- [8] World Meteorological Organization. WMO Website. <http://www.wmo.int>, August 2009.
- [9] Brian Blanton. read grib V1.4.0, a WMO GRiB file reader for MATLAB. http://www.opnml.unc.edu/OPNML_Matlab/read_grib/read_grib.html, August 2009.
- [10] World Meteorological Organization. A GUIDE TO THE CODE FORM FM 92-IX Ext. GRIB Edition 1. <http://www.wmo.ch/pages/prog/www/WDM/Guides/Guide-binary-2.html>, August 2009.
- [11] Wilhelm Burger & Mark J. Burge. *Digital Image Processing: An Algorithmic Introduction Using Java*. Springer, 2008.
- [12] Nav Canada. Aviation Weather Website, Forecasts and Observations. http://www.flightplanning.navcanada.ca/cgi-bin/CreePage.pl?Langue=anglais&NoSession=NS_Inconnu&Page=forecast-observation&TypeDoc=html, August 2009.
- [13] National Resources Canada. Geo-gratis, Download Directory, Vector Data. <http://geogratis.cgdi.gc.ca/geogratis/en/download/vector.html>, August 2009.
- [14] ESRI. ESRI Shapefile Technical Description: An ESRI White Paper. Technical report, Environmental Systems Research Institute, Inc., 1998.
- [15] NASA. Blue Marble Next Generation. <http://earthobservatory.nasa.gov/Features/BlueMarble/>, August 2009.
- [16] Environment Canada. Operational Model Forecasts. http://www.weatheroffice.gc.ca/model_forecast/index_e.html, August 2009.

- [17] Duane A. Bailey. *Java Structures*. Number 174. WCB/McGraw-Hill, 1999.
- [18] Luiz Antonio de Rose. *Compiler Techniques for Matlab Programs*. PhD thesis, University of Illinois, 1996.

Appendix A

Software Manual

Introduction

This manual describes the installation, layout, and function of the Environmental Feasibility Operations Package or EFOP. EFOP is a software package that helps facilitate responsible decision making when deploying unmanned aerial vehicles. EFOP software enables the aircraft operator to easily plan missions, assess the mission feasibility and execute the mission.

System Requirements

- High Performance Workstation *
- Microsoft® Windows®
- MATLAB® 2007a or Newer **
- Image Processing Toolbox™

- Mapping Toolbox™
- Unmanned Flight Dynamics Aerosim Blockset

** Although this software will run on slower machines, the amount of time required to run the model is directly related to the performance of the machine. Higher performance machines (i.e. i7 or Core 2 Duo with 4GB of Ram) can effectively decrease runtime.*

*** The software was designed to work with MATLAB® 2007a but unfortunately may exhibit some memory management issues. This issues may be remedied through the use of newer a version of MATLAB® (MATLAB® 2009a performs better).*

Installation

Presently EFOP does not include an automated installation process. Installing EFOP involves installing several components. This procedure is detailed below. It is assumed that you have a computer with Microsoft® Windows® already installed.

1. Install MATLAB® r2007a or greater (preferably r2009a) with both the Image Processing Toolbox™ and the Mapping Toolbox™. MATLAB® installation documentation can be followed. If MATLAB® and the appropriate toolboxes are already installed, skip to setup 2.
2. Download the Aerosim blockset from the Unmanned Dynamics website (<http://www.u-dynamics.com/aerosim/>). Install the blockset using instructions from the Aerosim User Guide. License information for the Aerosim blockset is available on the Unmanned Dynamics website.
3. Copy the contents of the EFOP DVD to a suitable location (EFOP location) on the workstation.

The software can be launched by first setting the current directory in MATLAB® to the location where EFOP was copied to. The command `run('launch')` typed in the MATLAB® command window will launch the software.

Software Layout

The following screen-shot illustrate the basic layout of the software while in planning mode:

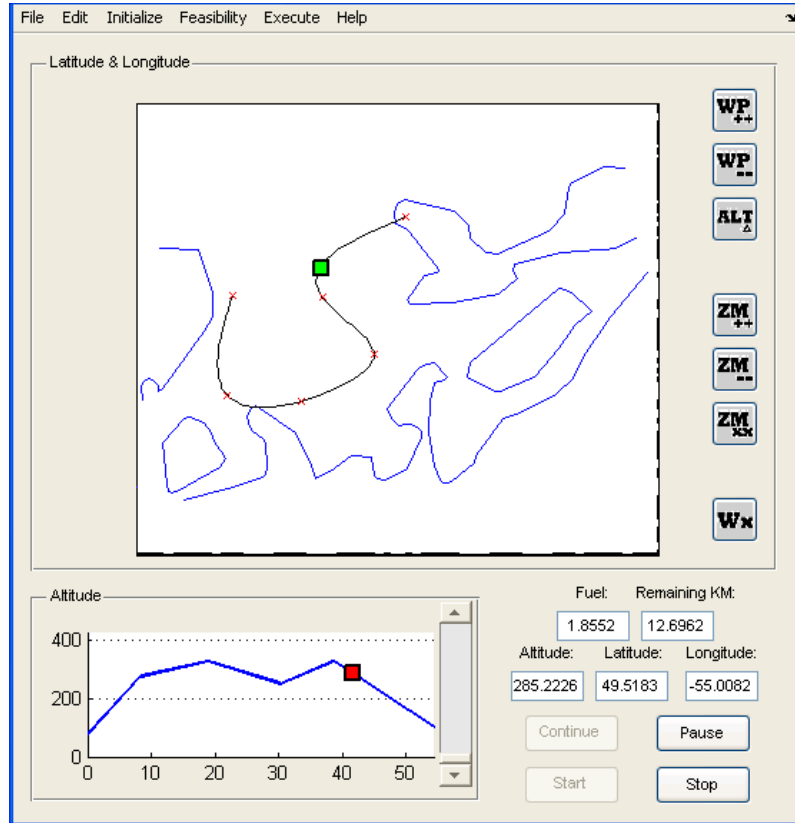


Figure A.1: Screen-capture of EFOP while in Planning Mode

Obtain Weather Data

Download Latest Weather Data:

1. Click the Initialize drop-down menu.
2. Select Get Weather Data

Load Saved Weather Data:

1. Click the **File** drop-down menu.
2. Select **Open**
3. Select **Open Weather Data**
4. Select the weather data file to load.

Environmental Tools

There are several environmental tools built into this software to help the operator make crucial operational decisions. These tools are listed below, as well as a brief description of how they can be significant and the data/network requirements for them to function.

Tool 1 • Display Weather Charts

Description: This tool allow the operator to download and display meteorologically significant weather charts from various online sources and display them seamlessly in the tabbed pop-up window.

Location: In the Initialize Menu under Display Wx Charts

Requirements: Live internet connection to acquire weather data.

Significance: These charts display various information that cannot be layered on top of the EFOP map.

Tool 2 • Overlay Show Arrows

Description: This tool overlays a graphical representation of wind speed and direction on top of any displayed map. The altitude and time of overlay is entered.

Location: The Wx button is in the lower right corner of the `Latitude & Longitude` map area.

Requirements: Live internet connection to acquire weather data.

Significance: These overlay allows the operator to explore weather information at a specific location, time, and altitude.

Tool 3 • Integrate Weather Models into Simulation

Description: This tool allows the operator to determine the feasibility of a mission by simulating the proposed mission using detailed weather models.

Location: Feasibility menu items.

Requirements: Live internet connection to acquire weather data.

Significance: Being able to perform feasibility analysis increases confidence in missions and decreases the likelihood of losing aircrafts due to severe weather.

Mission Planning

Defining a Series of Waypoints

Mission planning mode can be entered by clicking on the `Define Trajectory` item in the `Initialize` menu.

- Once in the mission planning mode the **First Point** button appears at the bottom of the window. Before the button is pressed the starting altitude needs to be selected via the altitude slider next to the altitude plot.
- The **First Point** button is pressed and the mouse cursor becomes a crosshair when placed over the Latitude & Longitude map. When the desired location on the map is clicked, the altitude, latitude, and longitude are stored as a rough waypoint and displayed on the map.
- These steps must be repeated; set altitude, click **Next Point** button, and select location on map, until a desired trajectory to rough waypoints is formed. All trajectories must consist of at least 3 waypoints.
- The **Done** button is selected to stop the rough waypoint selection. The fine waypoints (aircraft trajectory) will be drawn on the map.

Previous waypoints cannot be adjusted while the series of waypointed is being recorded; therefore, previous errors can not be corrected while defining a series of rough waypoints. To allow for modifications and refinements to a trajectory, buttons to change the altitude of a waypoint, add a waypoint, and a remove waypoint, become visible when the **Done** button was clicked.

Modify Trajectory

A trajectory can be modified by selecting either the change altitude button, the add waypoint button or the remove waypoint button. These are located to the right of the **Latitude & Longitude** map.

Change Altitude:

1. Select the new desired altitude on the altitude slide-bar.
2. Click the change altitude button.

3. The cursor will change to a cross-hair. Click closest to the waypoint that needs an altitude change. The closest waypoint will be circled in green.
4. If this is the correct waypoint and you want to accept the altitude change, click the left mouse button, and if you want to cancel the altitude change, click the the right mouse button.

Add Waypoint

1. Select the altitude of the new point on the altitude slide-bar.
2. Click the **Add Waypoint** button.
3. The cursor will change to a cross-hair. The location to insert the waypoint must be selected. Click in between the two waypoints you want to insert the new waypoint. When clicking between the two waypoints, it is important that your click is co-linear and in between the two points. The two waypoints will be circled in green. (In the case you click near the end waypoint or starting waypoint, only one waypoint will be selected)
4. If this is not the correct two waypoints click the the right mouse to cancel this operation, and try gain.
5. The cursor will change to a cross-hair. Click the location of the new waypoint on the **Latitude & Longitude** map. The new way point will be added to the trajectory.

Remove Waypoint

1. Click the **Remove Waypoint** button.
2. The cursor will change to a cross-hair. Click closest to the waypoint that needs to be removed. The closest waypoint will be circled in green.
3. If this is the correct waypoint to remove, click the left mouse button, and if you want to cancel, click the the right mouse button.

Load Flight Trajectory

Once a flight trajectory has been created it is possible to save this trajectory for later use: click **File** menu → click **Save As** item → select **Save Flight Trajectory**. Previously saved files can be loaded by clicking on **Open** in the **File** menu and selecting **Open Flight Trajectory**.

Feasibility Measurement

Performing a feasibility analysis on a mission requires clicking the **Feasibility** drop-down menu and selecting **Run Simulation**. A window with the mission start time and weather data usage must be filled out. The **Start** button in the bottom right of the window will start the simulation.

A simulation summary will be displayed when the simulation definition is complete. The **Start** button in the bottom right of the window will start the simulation.

Mission Execution

Select the **Execute** drop-down menu and then select **Execute Mission**. The **Start** button in the bottom right of the window will start the mission *. Mission changes can be preform by using the various buttons in the lower right hand corner.

** This feature is presently not connected to a real aircraft. This feature runs the Aerosim simulation model to demonstrate functionality.*

Appendix B

Grib Data Set Parameters

There are many parameters in the grib data sets. These parameters can be seen in table B.1.

Table B.1: GRIB Data Set Parameters

Parameter Number	Parameter description	Abbreviation	Levels	Units
001	Pressure	PRES	SFC	Pa
002	Pressure Reduced to Mean Sea Level	PRMSL	MSL	Pa
007	Geopotential Height	HGT	ISBL (28 isobaric levels) ISBY (layer between two isobaric levels) SFC	gpm
011	Temperature	TMP	ISBL (28 isobaric levels) TGL (2m above ground)	K
017	Dew Point Temperature	DPT	TGL (2m above ground)	K
018	Dew Point Depression	DEPR	ISBL (28 isobaric levels) TGL (2m above ground)	K
032	Wind Speed (module)	WIND	ISBL (4 isobaric levels) TGL (2m above ground)	$\frac{m}{s}$

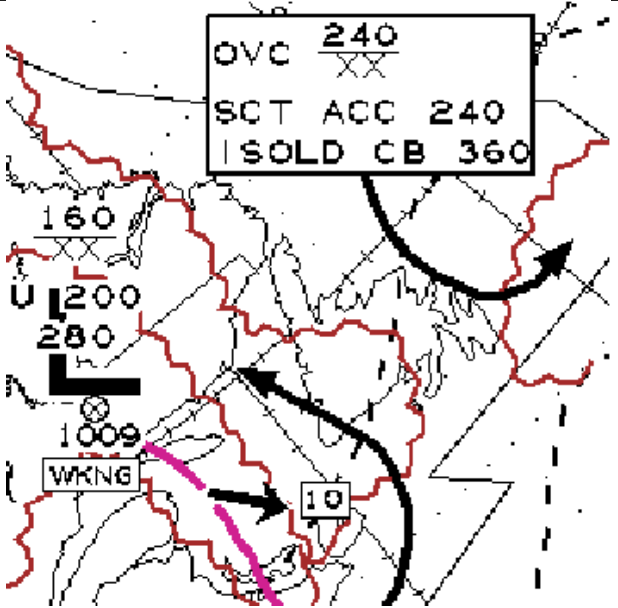
Continued on next page

GRIB Pa- parameter Number	Parameter descrip- tion	Abbrevi- ation	Levels	Units
033	U-Component of Wind	UGRD	ISBL (28 isobaric levels) TGL (2m above ground)	$\frac{m}{s}$
034	V-Component of Wind	VGRD	ISBL (28 isobaric levels) TGL (2m above ground)	$\frac{m}{s}$
039	Vertical Velocity	VVEL	ISBL (4 isobaric levels)	$\frac{Pa}{s}$
041	Absolute Vorticity	ABSV	ISBL (4 isobaric levels) SFC	$\frac{1}{s}$
059	Precipitation Rate	PRATE	SFC	$\frac{kg}{m^2 \cdot s}$
061	Total Precipitation	APCP	kg/m ²	$\frac{kg}{m^2}$
071	Total Cloud Cover	TCDC	SFC	<i>percentage</i>
204	Downward Short Wave Radiation Flux	DSWRF	SFC	$\frac{W}{m^2}$

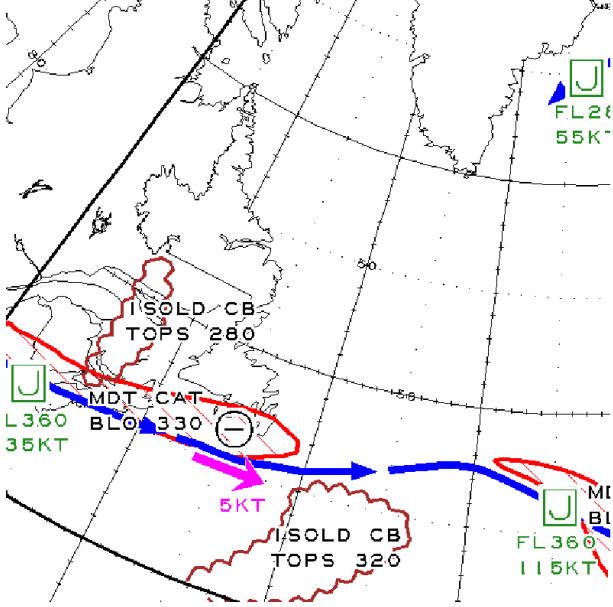
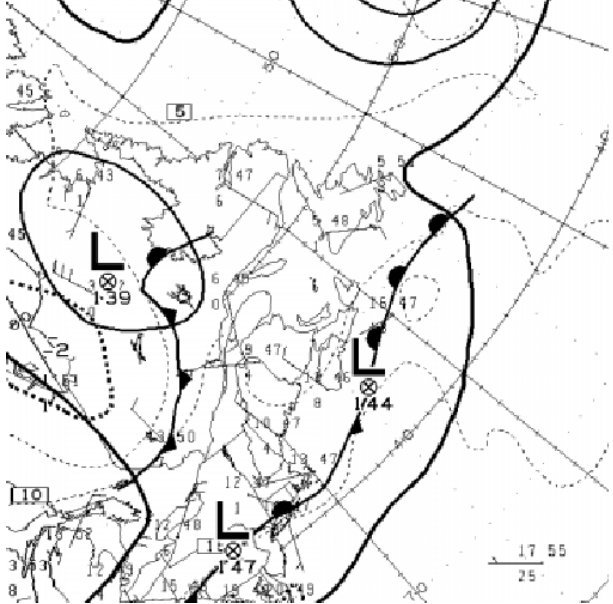
Appendix C

Various Weather Charts

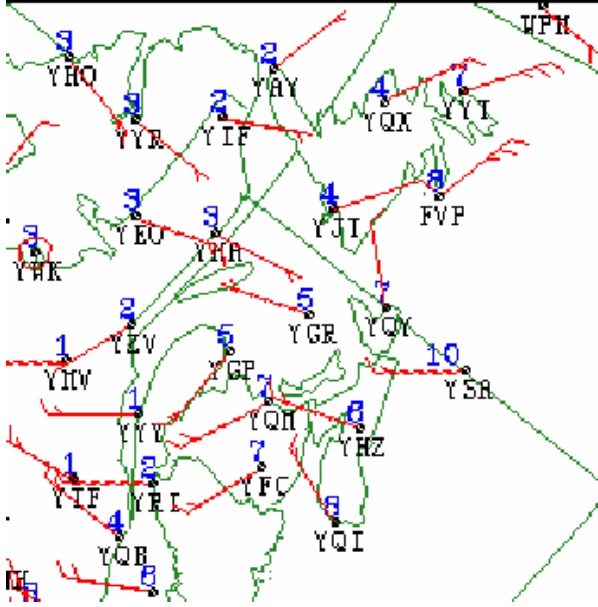
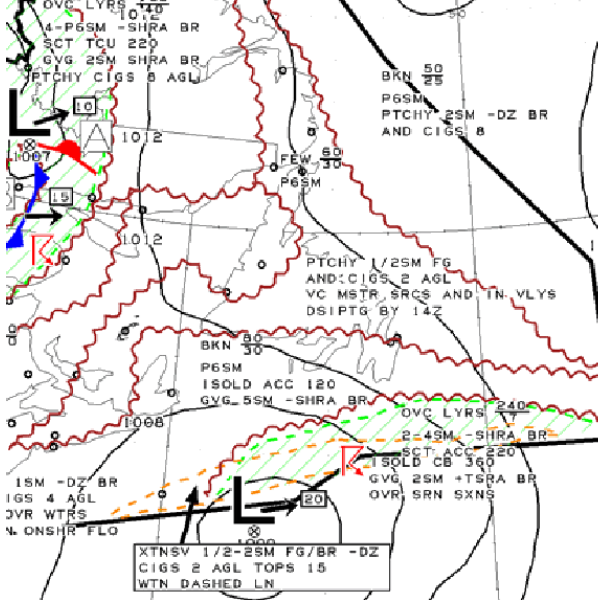
Table C.1: Various Weather Charts

Chart	Description	Sample Charts
Significant Weather Chart	Shows where the significant weather.	

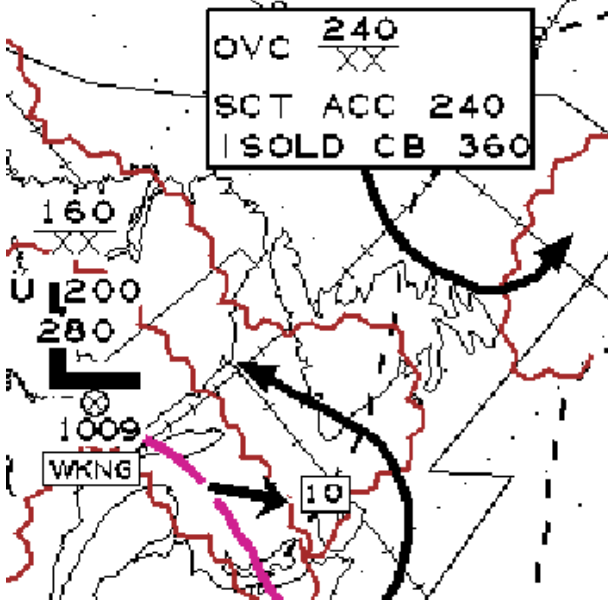
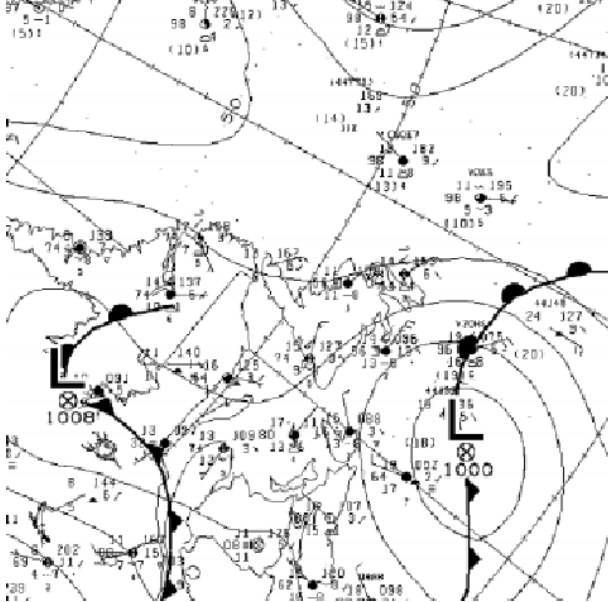
Continued on next page

Chart	Description	Sample Charts
<p>Turbulence Forecast for North Atlantic</p>	<p>Shows where and at what altitude turbulence may be experienced.</p>	 <p>The chart displays a map of the North Atlantic region with a grid of latitude and longitude. It features several key elements: <ul style="list-style-type: none"> Red outlines representing cloud tops, with labels such as 'SOLD CB TOPS 280' and 'SOLD CB TOPS 320'. Blue arrows indicating flight paths or wind directions. Green text labels for flight levels and altitudes: 'FL 280 55KT', 'L 360 BLO 330 35KT', and 'FL 360 115KT'. A pink arrow labeled '5KT' pointing towards the right. A central circle with a minus sign (-). </p>
<p>Upper Air Analysis Charts</p>	<p>Shows the wind and the temperature of the upper atmosphere</p>	 <p>The chart is a detailed upper air analysis map showing wind and temperature patterns. It includes: <ul style="list-style-type: none"> Isobars (lines of equal pressure) and isotherms (lines of equal temperature). Wind vectors represented by arrows of varying lengths and directions. Pressure levels marked with numbers such as 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100. A scale bar at the bottom right indicating 17 and 25 units. </p>

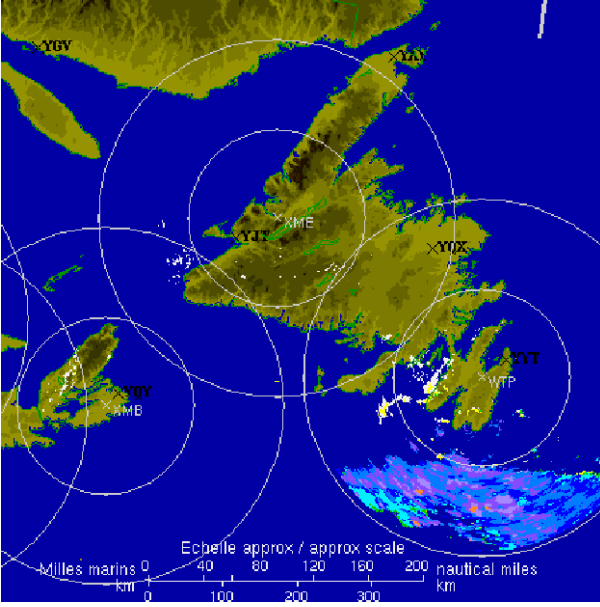
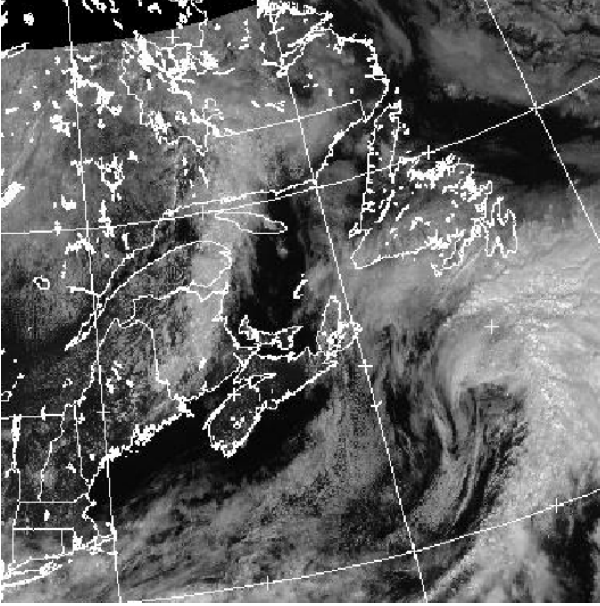
Continued on next page

Chart	Description	Sample Charts
Wind Direction and Speed Charts	Significant Weather Chart	
Graphical Area Forecast	Significant Weather Chart	

Continued on next page

Chart	Description	Sample Charts
Precipitation Charts	Significant Weather Chart	 <p>A Significant Weather Chart (SWC) for a region of the United States. It features a high-pressure system (H) at 1009 and a low-pressure system (L) at 1000. A cold front (solid line with triangles) and a warm front (dashed line with semicircles) are shown. A squall line (dashed line with long dashes) is also present. A box in the upper right contains the following data: OVC 240, SCT ACC 240, and ISOLD CB 360. Other cloud layers are labeled as 160, 200, and 280. A station identifier WKNG is shown near the low pressure system.</p>
Surface Condition Analysis	Significant Weather Chart	 <p>A Surface Condition Analysis chart for the same region. It displays isobars (lines of equal pressure) and isotherms (lines of equal temperature). A high-pressure system (H) is centered at 1008 and a low-pressure system (L) is centered at 1000. The chart includes various weather symbols such as clouds, rain, and wind vectors. Station identifiers like WKNG and WJAX are visible.</p>

Continued on next page

Chart	Description	Sample Charts
Radar Imagery	Significant Weather Chart	 <p>The radar weather chart displays precipitation intensity over a geographical area. The intensity is represented by colors ranging from blue (lightest) to red (heaviest). Several airports are marked with their IATA codes: XTV, XME, XTY, XTP, XVT, and XWB. The chart includes a scale at the bottom: 'Echelle approx / approx scale' with markings for 0, 40, 80, 120, 160, and 200 nautical miles, and 0, 100, 200, and 300 kilometers. The text 'Milles marins 0' and 'km' is also present.</p>
Satellite Imagery	Significant Weather Chart	 <p>The satellite weather imagery shows cloud cover and atmospheric features over a geographical area. The image is in grayscale, with white and light gray areas representing clouds and darker areas representing clear sky or water. A grid of latitude and longitude lines is overlaid on the image.</p>

Vita

Candidate's full name: Sean R. Perry

University attended: Bachelor of Science in Electrical Engineering, UNB, 2006