Integration of Weather Data into Airspace and Traffic Operations Simulation (ATOS) for Trajectory-Based Operations Research

Contract No.: NNL08AA17B
Task Order No. NNL08AB91T

Document No.: 09-12458-03
30 September 2009

Prepared for:

NASA Langley Research Center
Mail Stop 126
Hampton, VA 23681-2199

Prepared by:

Mark Peters
Ben Boisvert
Diego Escala

1700 Dell Avenue
Campbell, CA 95008
(408) 364-8200
FAX: (408) 364-8270
CAGE Code: 42UY9

Detect the Difference

Acknowledgements

This project was supported by a contract with the NASA Langley Research Center. The author wishes to thank all individuals who made this work possible and thank those people whose insight and technical expertise contributed to the success of the project. Specifically, the author wishes to thank Jim Chamberlain, David Wing, and Michael Palmer, for assisting with numerous questions and helping to make the overall effort a success.

Table of Contents

List of Figures

Acronyms

AATT ……. Advanced Air Transportation Technologies
AOP ……Autonomous Operations Planner
APDLC ……Airport Pilot Datalink Communications
ARINC ……...Aeronautical Radio, Incorporated
ASTOR ……..Aircraft Simulation for Traffic Operations Research
ATC ……. Air Traffic Control
ATM ……..Air Traffic Management
ATOS ……..…… Air Traffic Operations Simulation
BTS ……...Bureau of Transportation Statistics
CD&R ……...Conflict Detection and Resolution
ConOps …… Concept of Operations
CNS . ……. Communication Navigation Surveillance
DAG-TM ……....Distributed Air Ground – Traffic Management
D-ATIS ……..……Digital – ATIS (Automated Terminal Information Service)
DOT ...……...…… Department of Transportation
DOF ……..…… Degree of Freedom
FAA …… Federal Aviation Administration
FAF ……..… Final Approach Fix
FMS ……..Flight Management System
GRIB ……..Gridded Binary
HLA ……..… High Level Architecture
IAF ……Initial Approach Fix
IFR …… Instrument Flight Rules
LaRC ……...NASA Langley Research Center
LaSRS ……NASA Langley Standard Real-time Simulation
NAS ……National Aviation System
NASA ……..…National Aeronautics and Space Administration
NLR ……National Aerospace Laboratory of the Netherlands
PDS ……...Pair Dependent Speed
RPFMS ……Research Prototype Flight Management System
RTI ……..……..…… Real Time Interface
TMX ……..Traffic Generation Tool

1 Introduction

This document details the work performed for the NASA Langley Research Center to integrate realistic weather information into the Airspace and Traffic Operations Simulation (ATOS) under contract number NNL08AA17B. Four main tasks were performed under this effort. These are:

- Data Conversion (polygonalization of the radar data)
- WSI Pilot Brief Pro Emulator
- Convective Weather Simulator
- High Impact Scenarios

This document focuses on the theory of operation for the various tools and algorithms developed under this contract to support the first three tasks. The High Impact Scenarios task, due to its scope, is covered in its own separate document. For the tools created under this contract, separate users guides are provided that detail operation of the software.

1.1 Objectives and Significance

The purpose of this effort is to develop capabilities that enable the integration of weather forecast information and weather phenomena into, ATOS, a large, distributed air traffic management simulation at the Langley Research Center. The integration of weather information into trajectory management decision-making is critical to research on trajectory-based air traffic management (ATM) and performance-based operations that strive to increase capacity and safety.

Weather is the largest contributing factor to air traffic delay in the National Airspace System[1]. Among weather hazards, convective weather is one of the most serious hazards in aviation and contributes greatly to delays in air travel. Convective weather hazards may inhibit traffic flow across a few square miles or across thousands of square miles, depending on the weather configuration. Similarly, gaps in convective activity can be large enough to allow significant traffic to flow through, or so small that only a single aircraft can pass. The degree of hazard thus becomes a function of weather intensity. The nesting of multiple intensity levels, in turn, raises still more complex issues regarding acceptable proximity versus other ATM objectives, such as efficiency or traffic separation. The dynamic characteristics of convective weather (e.g., growth, translation, and decay) must also be considered in trajectory management decision-making. Therefore, the rerouting problem is dynamic and time-dependent, and as weather conditions change, routing strategies have to be modified. To study these many issues in simulation, recorded and simulated convective weather data must be incorporated into the ATOS.

Other more benign phenomena influence the system in often subtle ways. Consider the impact of final approach headwinds on airport capacity. Headwinds cause the aircraft

groundspeed to decrease. Therefore, for aircraft flying at given airspeed profile, with given in-trail spacing on final approach, the presence of a headwind increases the inter-arrival time spacing, and thus reduces the runway throughput[2].

1.2 ATOS Background

ATOS is a large, distributed, medium-fidelity, human-in-the-loop, multi-aircraft simulation originally developed to explore the Distributed Air/Ground Traffic Management (DAG-TM) concept, under NASA's Advanced Air Transportation Technologies (AATT) project. DAG-TM describes a set of future traffic operations that redefines the roles of flight crews, air traffic service providers, and aeronautical operational control organizations. It is based on the premise that the sharing of information between these system participants and the delegation of decision authority to the most appropriate decision maker will result in large improvements to airspace system[3].

Through several years of research and development, the NASA Langley Research Center (LaRC) has made progress in the development of the ATOS system and has used it to study DAG-TM concept elements in the en-route and terminal airspace. The ATOS system consists of multiple workstation-based components networked together through High Level Architecture (HLA). ATOS components usually operate on hardware dedicated for that process and are referred to as federates in the HLA architecture. A special federate, the simulation manager, manages the simulation. The simulation manager controls time, events and simulation system modes. The only non-federate is the Run Time Infrastructure (RTI) exec, an HLA process which manages HLA communications and other HLA specific tasks. Figure 1 illustrates the components of the Federation[4]

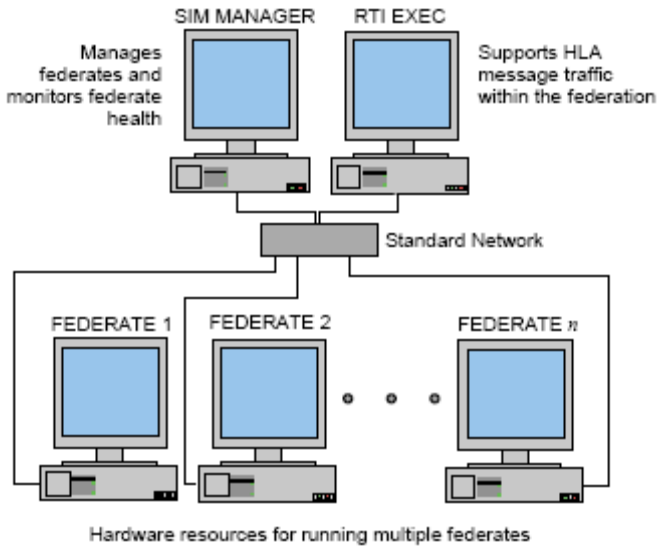

Figure 1: HLA Federation[4].

Each piloted aircraft within ATOS is a separate workstation-federate running a specially-developed aircraft model referred to as an ASTOR (Aircraft Simulation for Traffic Operations Research).[5] Also on this network as a separate federate is a traffic generation tool (TMX) developed by the National Aerospace Laboratory of the Netherlands (NLR).[6]

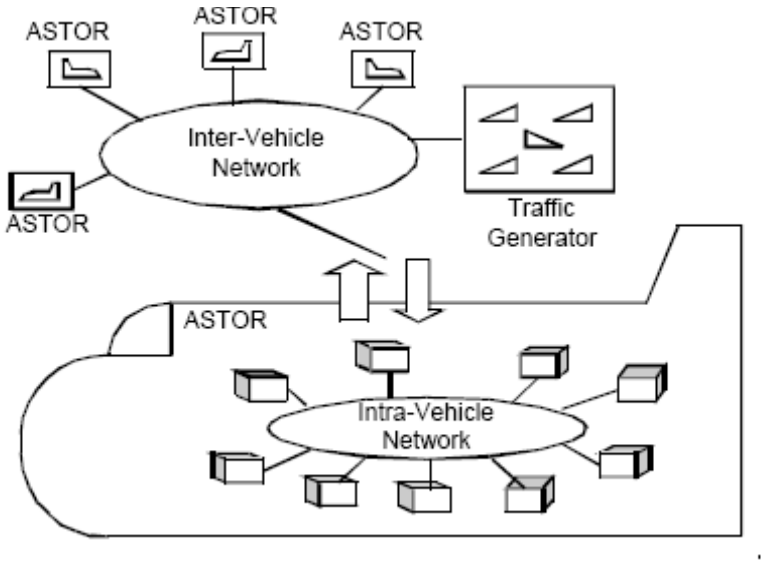

Figure 2. Illustration of network with multiple ASTORs and a traffic generator[5].

An ASTOR aircraft is comprised of a series of processes that all reside on an individual workstation which then communicate through an inter-vehicle network. Separate components (e.g. FMS, displays etc) are individual processes. To help achieve the maintainability and reusability the ASTOR avionics architecture is that of a simulated ARINC 429 data bus (see Figure 3) with various channels defined for communications into and out of specific modules of the simulation. Each module roughly corresponds to a current-generation avionics component or to a new research capability that may eventually be incorporated into new or existing flight decks.[3]

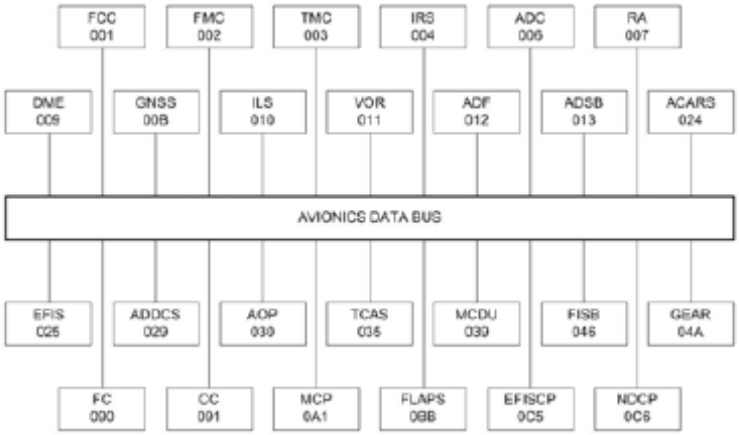

Figure 3. ASTOR ARINC429 data bus [3]

The airframe simulation model is based on the NASA Langley Standard Real-time Simulation framework in C++ (LaSRS++). LaSRS++ is a full 6 degree-of-freedom dynamic simulation that models all flight forces based on a combination of linear and non-linear stability and control derivatives and tabular drag polar and engine thrust data. The atmospheric data (e.g. air density, winds) used to model applied forces can either be supplied internally using Standard Day Atmosphere tables or it can be supplied from an external database based on GRIB (Gridded Binary) files. LaSRS++ provides aircraft state data to the avionics bus are via sensor and instrument models, auto-flight, and flight control actuation systems.

The ASTOR flight deck system has been developed in compliance with existing and advanced avionic system specifications to verify research concept benefit in a realistic onboard Communication, Navigation, and Surveillance (CNS) system environment. The ATOS system supports advanced ATM research by hosting prototypes of crew decision support tools such as the Research Prototype Flight Management System (RPFMS), Autonomous Operations Planner (AOP) to provide flight crew Conflict Detection and Resolution (CD&R) during all en route phases of flight, and a Pair Dependent Speed (PDS) guidance mode to support airborne merging and spacing upon entering terminal arrival areas.[6]

Each ASTOR federate needs configuration data that allows it to operate in the simulation. This configuration data is called scenario data and can consist of one or several scenario files. To simplify the creation of scenario data to the researcher, the ATOS system includes an offline Scenario Generator tool. The Scenario Generation tool consists of a database, a user interface, input data parser and automates the creation and distribution of scenario files for each federate in the system. The Scenario Generation tool also guarantees the consistency of the numerous scenario data files needed to run a scenario.

2 Data Conversion (Polygon Tool)

The objective of the data conversion task is to process weather data into a usable format for the ATOS simulation. The AOP function of the ATOS simulation is already designed to create routes around hazards with lateral bounds defined as polygons, where the vertices are latitude / longitude positions. Therefore, to take advantage of existing AOP functionality, it makes sense to represent the hazardous weather using polygons.

This section details an algorithm to create weather polygons from radar reflectivity weather data. The weather data is in bitmap format. The weather information is read into the computer where it can be visually displayed and used to create weather polygons. The weather polygons can then be overlaid onto the original bitmap for comparison with the actual weather data or displayed on a separate map. The weather polygons will make

it possible to integrate weather data into the Airspace and traffic Operations Simulation (ATOS).

2.1 Image Scaling and Projections

The polygon algorithm assumes that the bitmap image uses a cylindrical equidistant projection with the reference map coordinate lat/long (μ_o, l_o) defined in the center of the image. The image is then 250nmi x250 nmi square. Equations (2.1) and (2.2) define the transformation.

$$\mu = \mu_o + \frac{(y - y_o)}{R_{Earth}} \qquad (2.1)$$

$$l = l_o + \frac{(x - x_o)}{R_{Earth} \cos \mu_o} \qquad (2.2)$$

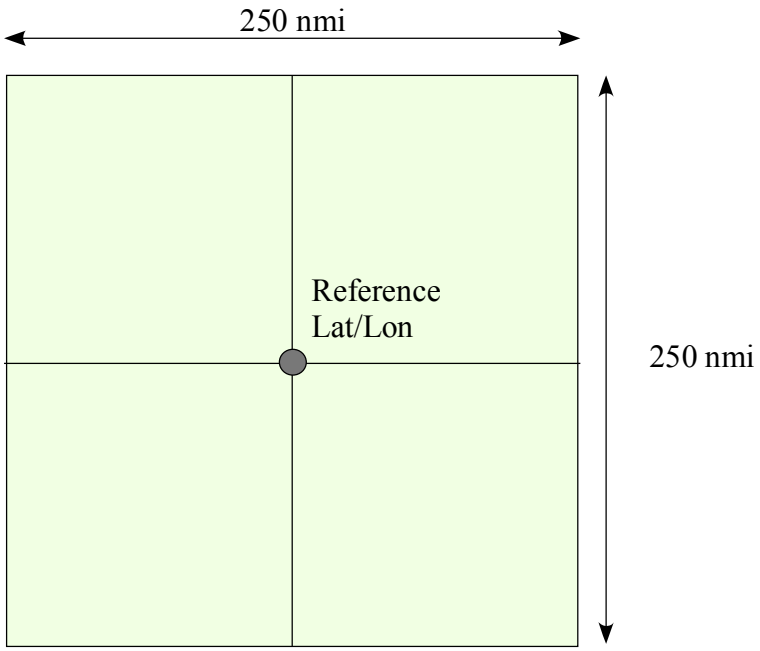

Figure 4. Drawing of Bitmap reference

2.2 Wildfire Enlarging

When avoiding special use or hazard polygons, the trajectory generator in AOP assumes all space not enclosed within the polygon is safe. Therefore, AOP may generate trajectories right up to the boundaries of the polygons. This is undesirable if the radar

intensities are high along polygon boundaries, so the polygons must include a factor of safety around high intensity weather.

An easy way to incorporate a factor of safety is to use a wildfire algorithm. A "Wildfire" algorithm, named for how fire spreads to surrounding areas, grows a specified region laterally in all original directions (depending on engineering design preferences).[9]

The wildfire algorithm examines the individual pixels within the bitmap. If a given pixel contains a reflectivity value, the surrounding pixels are raised to that value. If the surrounding pixels hold a higher value, the pixels are not affected. Figure 5 shows the effect of the wildfire algorithm.

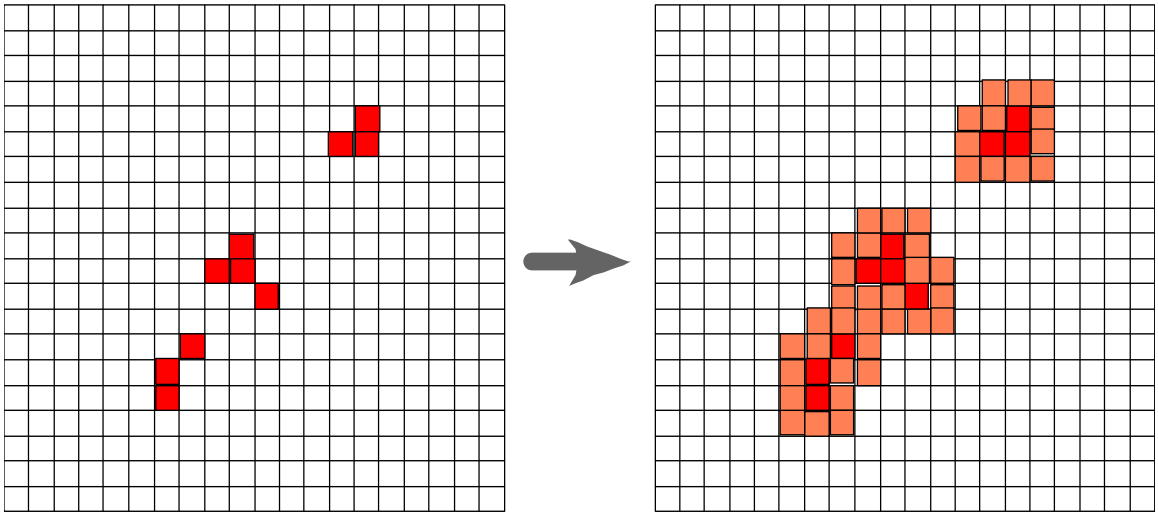

Figure 5. Example of Wildfire Algorithm

Inevitably, the factor of safety is larger around higher intensity weather. Therefore, the various weather intensities are handled independently. High intensity weather (e.g. greater than 50 db) may require 20 nmi of buffer, whereas low intensity weather may not require wildfire enlarging and may even be ignored.

To affect arbitrary factors of safety, successive wildfire operations are performed. The assumption is that each wildfire operation will add about 1 pixel length to the radius of the storm cell. The number of operations is then determined by the required factor of safety expressed in pixels. To convert from a nautical mile factor of safety to pixels, the scale factor of the bitmap (pixels/nm) is used. Equation (2.3) expresses the relationship

$$N_{wildfireOperations} = F_s(nmi) \cdot S_f\left(\frac{pixel}{nmi}\right) \qquad (2.3)$$

where F_s is the factor of safety expressed in nmi, and S_f is the scale factor of the bitmap expressed in terms of (pixel/nmi).

When the body of hazardous pixels has been established, the original weather intensity value associated with each pixel can be discarded. The bitmap array is normalized so that each pixel holds either a 'hazard /non-hazard' status.

2.3 Clustering Weather Data

Areas of adjacent pixels that contain hazardous weather ('hot' pixels) need to be identified so that the hazardous areas are separated from the non-hazard areas. Pixels that contain hazardous weather are grouped into weather clusters. These weather clusters form the basis of the polygons.

The algorithm starts by methodically searching through the bitmap from top to bottom and left to right. As soon as a "hot" pixel is found, the algorithm checks to see if it is already associated with a cluster. If it is not, a new weather cluster is initiated. From the initial pixel, adjacent 'hot' pixels are identified until no more adjacent pixels are found. This group of pixels then serves as a single cluster. The initial step of identifying a new cluster is shown in

Figure 6.

Once the initializing pixel for a new cluster has been identified, the search for adjacent pixels is started. Figure 7 shows the logic for searching through adjacent pixels. The algorithm creates three lists of pixels for creating a cluster. These are the Cluster List, Search List 1, and Search List 2. The Cluster List contains all the pixels that have been determined to be in the cluster. No further analysis is required for these pixels. The Search List 1 contains pixels that are in the cluster, but have not had their lateral bounds searched for adjacent pixels. Search List 2 is a list for keeping track of newly found pixels. The algorithm centers on Search List 1. The pixels in Search List 1 are sequentially examined for adjacent pixels that also belong in the cluster. Any pixels that are found are put in Search List 2. At the end of the sequence, all the pixels in Search List 1 can be placed in the Cluster List, and the newly found pixels are placed in Search List 1. The algorithm is initialized by placing the initial pixel in Search List 1. The process then repeats until a full sweep of Search List 1 produces an empty set for Search List 2. At this point the cluster is complete.

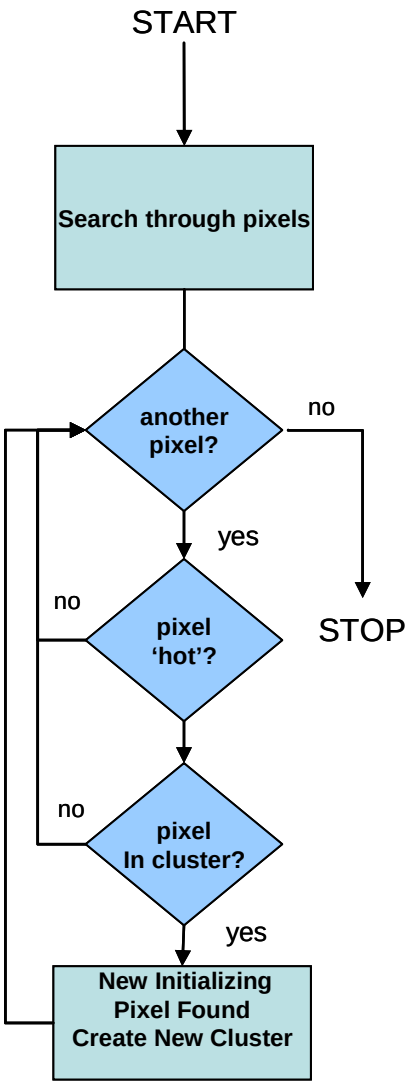

Figure 6. Searching for new Clusters

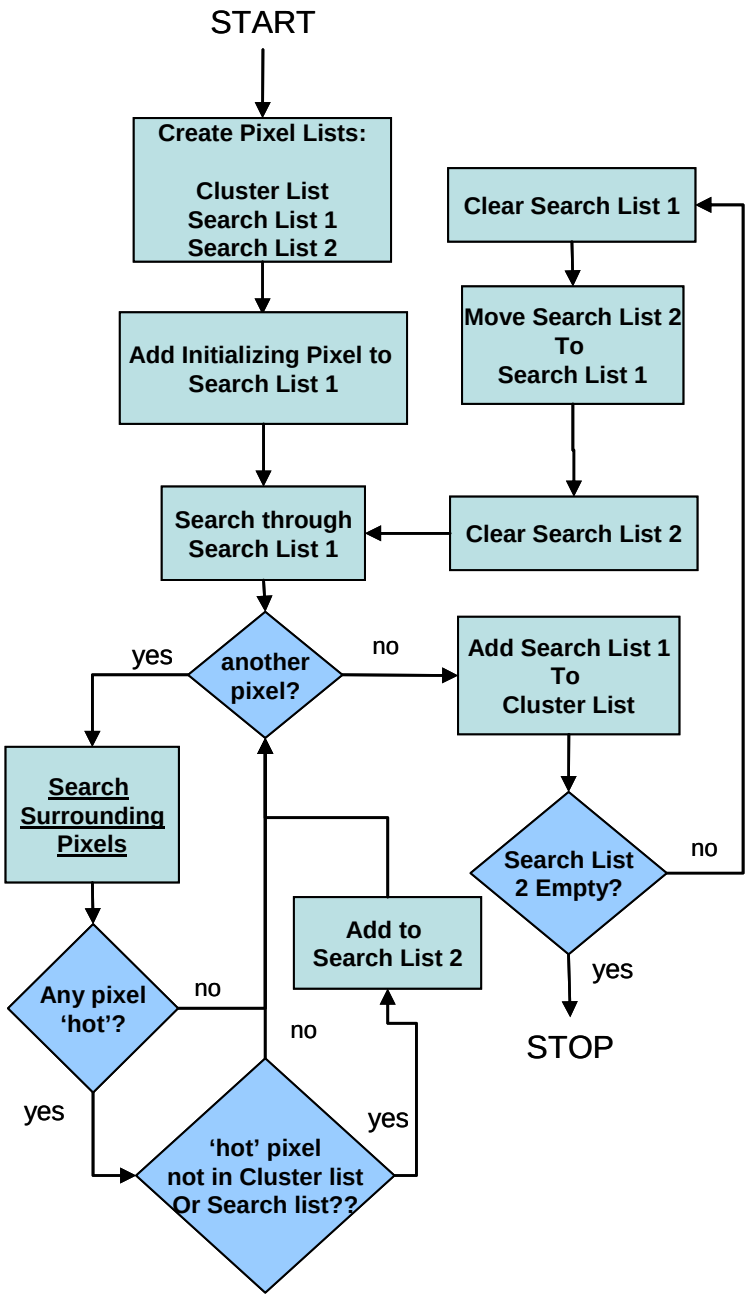

Figure 7. Creating Clusters

2.4 Determining Cluster Perimeters

The next step in creating the weather polygon is to find the perimeter of the weather cluster. The perimeter algorithm starts by finding the 'lowest' pixel in the cluster, which corresponds to the pixel with the greatest y value. If more than one pixel shares that

value, the pixel with the greatest x value is used. This minimum pixel is the first pixel in the set of perimeter pixels which will loop counterclockwise around the cluster.

To find the next perimeter pixel the algorithm starts a counter-clockwise search of the eight neighboring pixels surrounding the first perimeter pixel. The search starts with the pixel directly below the perimeter pixel and continues around the neighboring pixels until the first cluster pixel is identified. This becomes next perimeter pixel. Likewise a counter clockwise search is done around the second perimeter pixel starting at the last empty pixel (see Figure 4). The first "hot" pixel is once again identified and added to the set of perimeter pixels. This process continues until the perimeter is closed (see Figure 5).

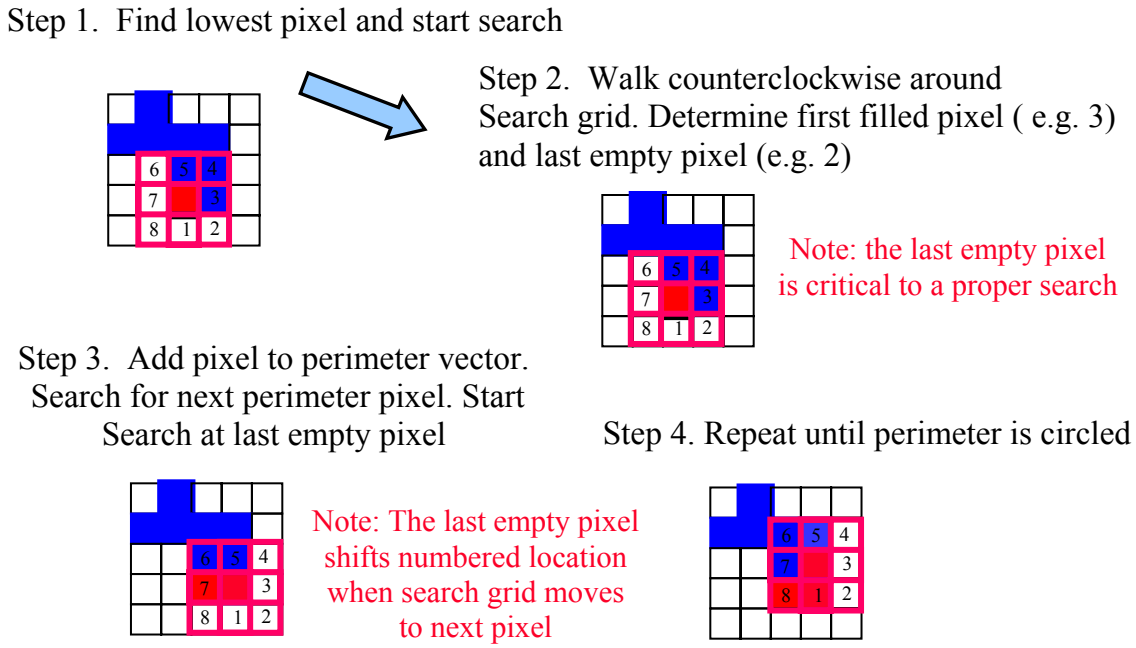

Figure 8. Perimeter Algorithm Diagram

2.5 Convex Hull Determination

The convex hull algorithm finds the polygon that encloses all of the perimeter points with a limited number of vertices. Figure 9 introduces an illustrative example of a grossly simplified 23 point perimeter that is used throughout this section to detail the methodology.

The first step in the convex hull algorithm is to create an initiating polygon using the 4 min/max points from the perimeter points. Figure 9 shows the initiating polygon.

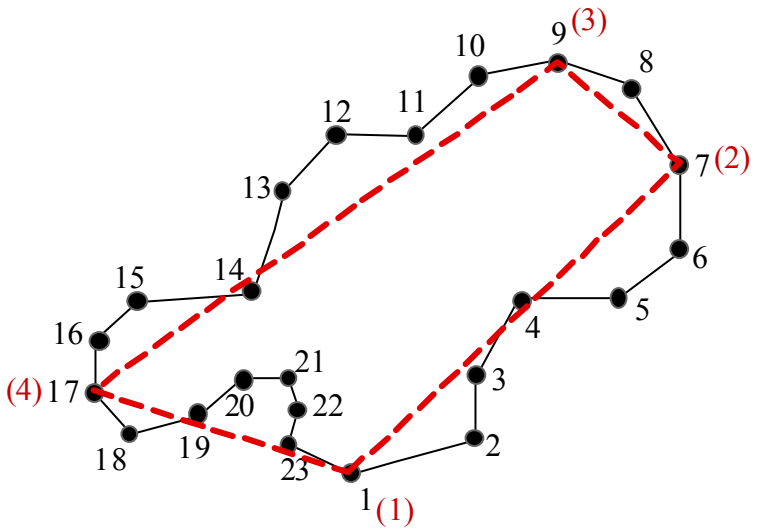

Figure 9. Initial polygon created from the 4 (min/max) points

The initiating polygon does not completely enclose the perimeter points. More points need to be added from the perimeter to enclose the cluster. To determine what points to add, careful bookkeeping of indices and vector algebra are employed.

Consider the polygon from Figure 9 in more detail. Of the 23 original perimeter points, four are used as vertices for the polygon. The perimeter points are all individually indexed (1-23), and the polygon vertices are indexed as well. To maintain a distinction between the two sets of indices, the polygon vertices are denoted by ( )). Therefore polygon vertices have an index corresponding to their order in the polygon but also retain their original index corresponding to their order in the perimeter. For example, vertex (2) of the polygon corresponds to point 7 of the perimeter. This mapping between the indices is used within the algorithm. Using this nomenclature, it is easy to keep track of what perimeter points lay between the vertices of the polygon. For instance, in Figure 9, the first two vertices of the polygon are indices 1 and 7 of the original perimeter. Enclosed between them are points 2,3,4,5, and 6.

From visual inspection, it is clear that the only point that lies within polygon along the interval between 1-7, is point 4. All the other points lie outside the polygon and must be enclosed. To determine how to expand the polygon to enclose the outlying points, vector algebra is used. First a vector, $\vec{V}_{(j),(j+1)}$, is defined which describes edge of the polygon as shown in Figure 10. A more detailed view of the edge in question appears in Figure 11. A unit vector normal to the edge vector is defined, $\hat{n}_r$, such that the unit vector always points in a right handed fashion from original vector. Since the edge vectors are oriented in a counterclockwise fashion, the unit vector always points outward. A series of vectors are then defined from the *jth* polygon vertex to all the perimeter points lying between the *jth and j+1* vertex. Each vector is dotted with $\hat{n}_r$ such that a scalar distance, *d,* from the polygon edge is calculated for each point. Figure 11 illustrates this process for the 6th perimeter point which also corresponds maximum distance in this case. The point with the maximum distance is then added as a vertex to the polygon as shown in Figure 12.

$$\vec{V}_{(j)(j+1)} = \left(x_{(j+1)} - x_{(j)},\ y_{(j+1)} - y_{(j)} \right) \tag{2.4}$$

$$\hat{n}_r = \frac{\left(y_{(j+1)} - y_{(j)},\ x_{(j+1)} - x_{(j)} \right)}{\sqrt{\left(x_{j+1} - x_j \right)^2 + \left(y_{j+1} - y_j \right)^2}} \tag{2.5}$$

$$\vec{v}_{(j),p} = \left(x_p - x_{(j)},\ y_p - y_{(j)} \right) \tag{2.6}$$

$$d_p = \vec{v}_{(j),p} \cdot \hat{n}_r \tag{2.7}$$

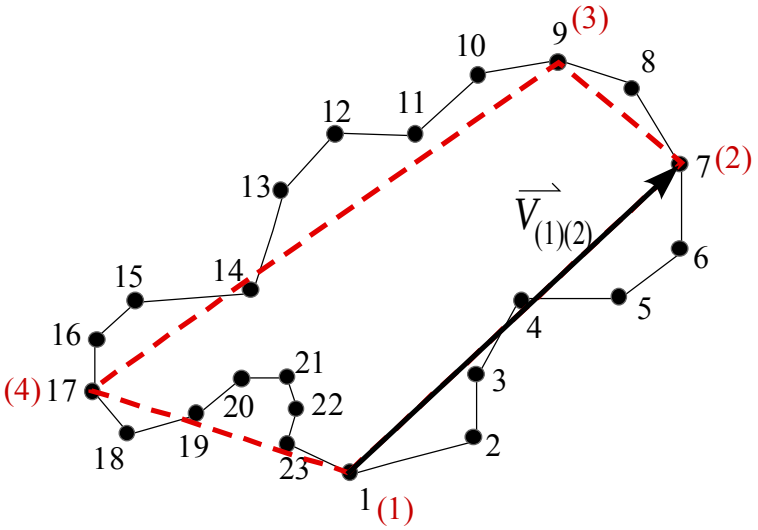

Figure 10. Vector defining the first edge of the polygon

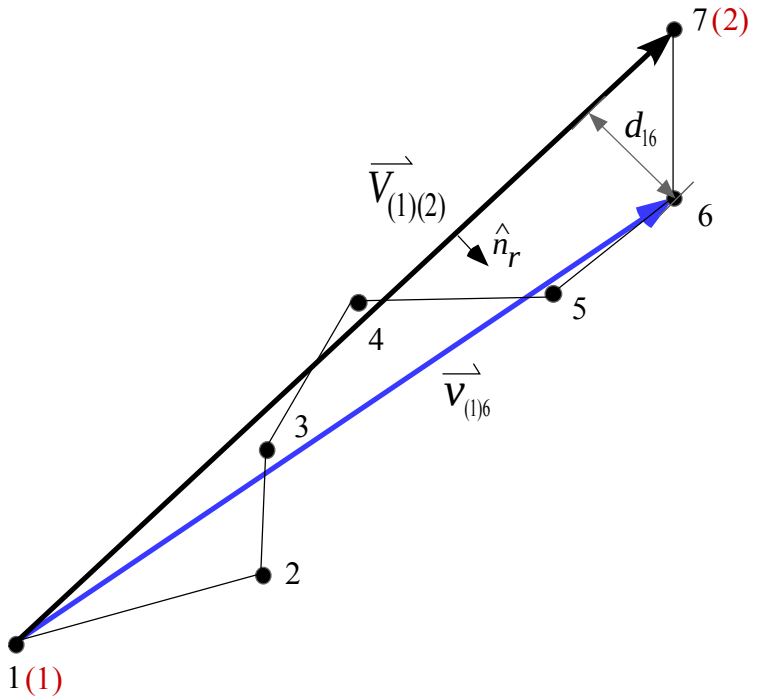

Figure 11. Expanded view of first polygon edge

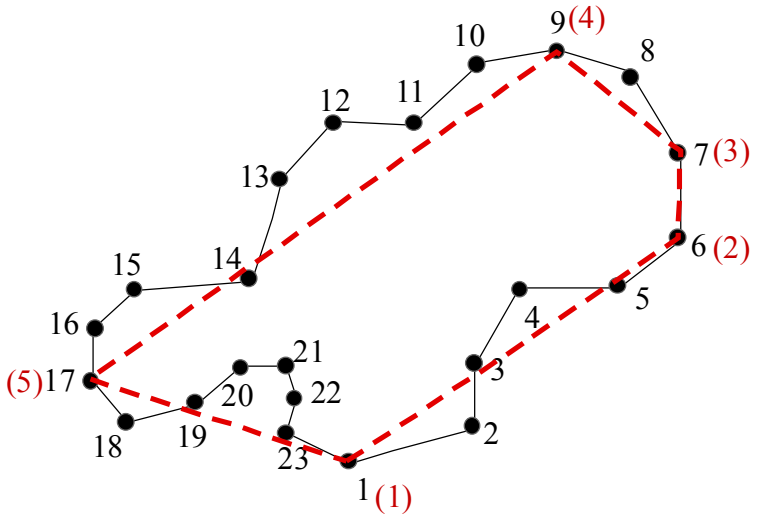

Figure 12. View of Polygon with perimeter point 6 added as a vertex

With the additional polygon vertex, perimeter points #2 and #5 are still outside the bounds of the edge of the polygon so the process is repeated. The #2 perimeter point is included as a polygon vertex on the second run as shown in Figure 13.

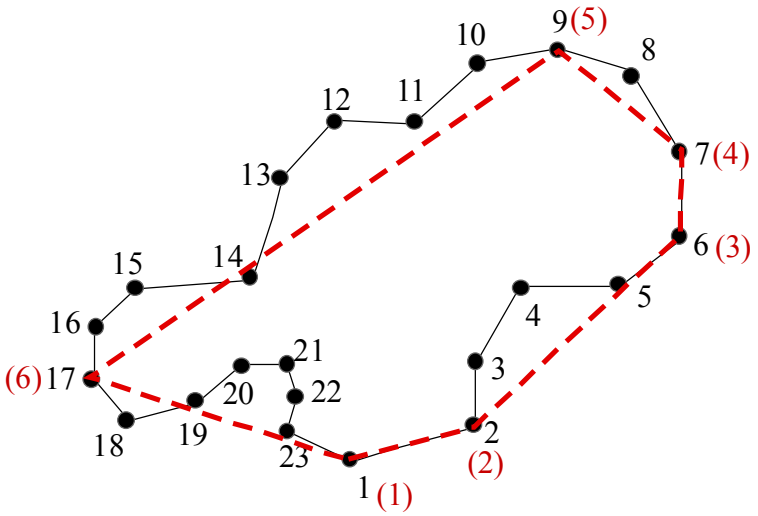

Figure 13. View of Polygon with perimeter point 2 added as a vertex

This process is repeated on all the edges of the polygon until all perimeter points are enclosed as shown in Figure 14.

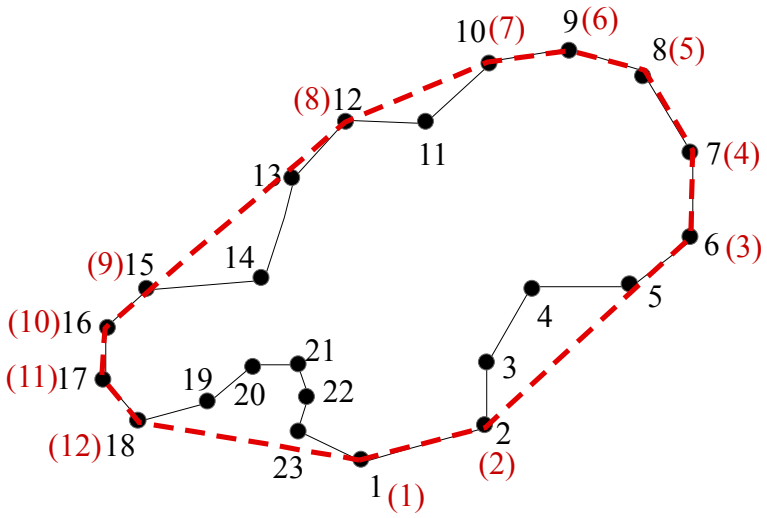

Figure 14. View of Polygon fully enclosing perimeter

The process is formalized in the flow diagram shown in Figure 15. Once the initiating polygon is determined, the algorithm loops through all the edges of the polygon. If needed to enclose points, the algorithm adds a single vertex to the polygon. The additional vertex should enclose some, but probably not all of the outlying points. Additionally, the new vertex effectively splits the original edge into two new edges. At this point the algorithm moves on to the next edge of the original set (not the newly created edge). The algorithm also makes a note of the fact that a vertex was added using a single Boolean operator 'RepeatVal.' Once the original set of edges has been operated on, the algorithm determines whether to run again. This decision is based on the value of 'RepeatVal'. A true value indicates that at least one vertex was added on the last run. If this is the case, the algorithm must run again. Once the algorithm makes a complete pass through without adding a vertex, it finishes.

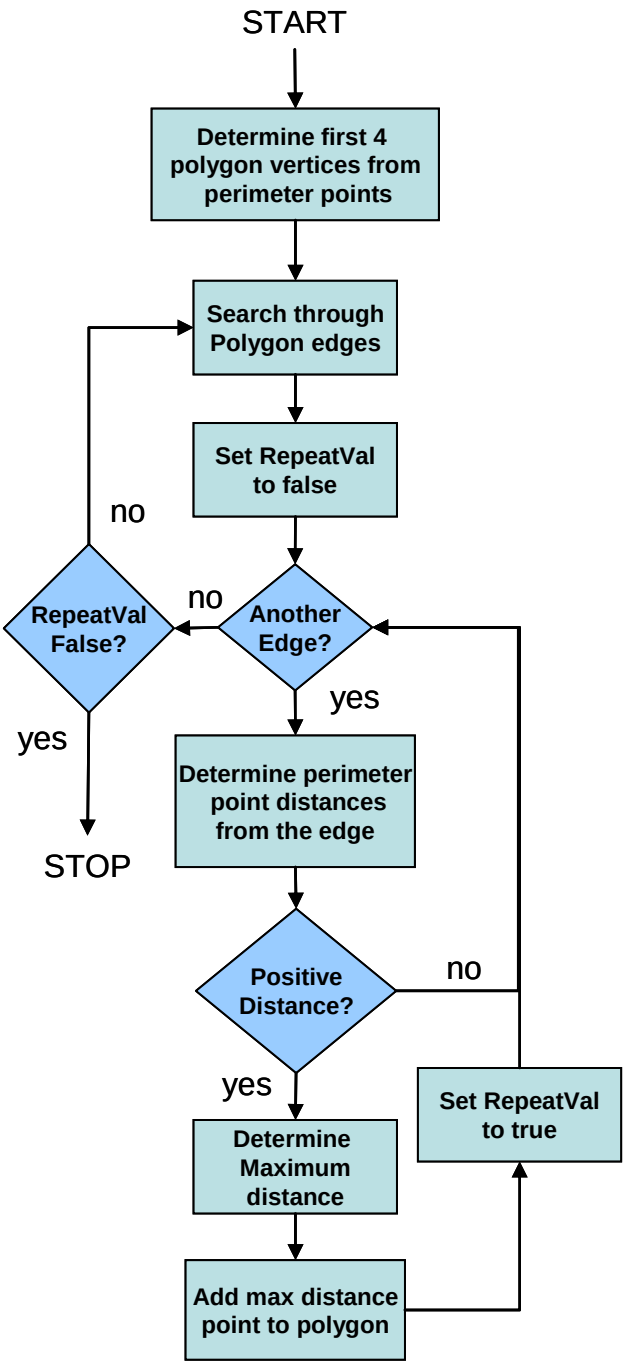

Figure 15. Flow diagram for determining convex hull

2.6 Concave Hull Determination

The convex hull algorithm will completely enclose the perimeter points but it may enclose an excessive area depending on the concavity of the storm. Often it is desirable to have a 'tighter fit' to the polygon to allow maximum use of the airspace. So, a concave hull algorithm is used to refine the weather polygon. Figure 16 illustrates a convex hull that encloses a large non-hazard area due to the concavity of the original perimeter.

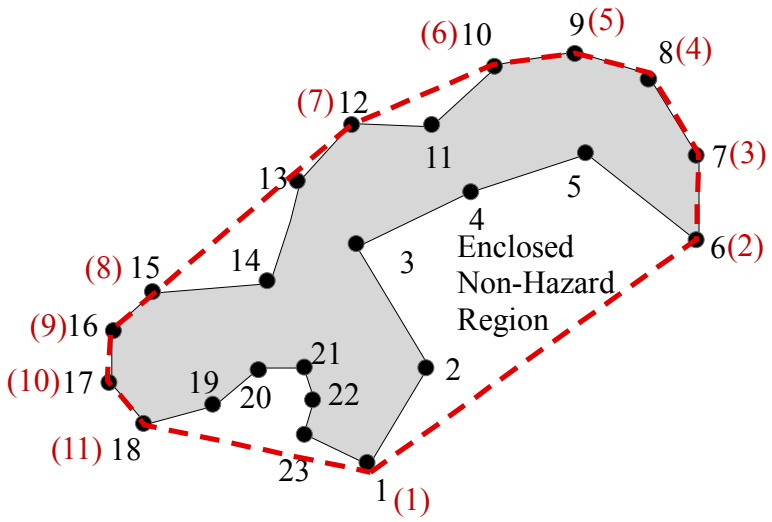

Figure 16. Convex hull that encloses an excessive amount of non-hazard area

To recapture some of this area, a modified convex algorithm from the previous section is run through one cycle looking for the maximum 'negative' perimeter point, along each edge as shown in Figure 17. If the distance associated with the perimeter point is large enough (according to some user specification), the point is added as a vertex to the polygon (see Figure 18). The problem however, is that the added vertex does not guarantee that all the hazard area is still contained. Therefore the convex hull algorithm is then re-run, using this modified polygon as the initializing polygon.

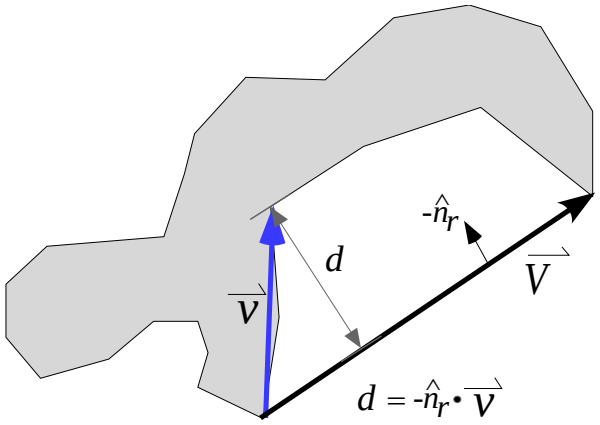

Figure 17. Vector algebra involved in determining concave points

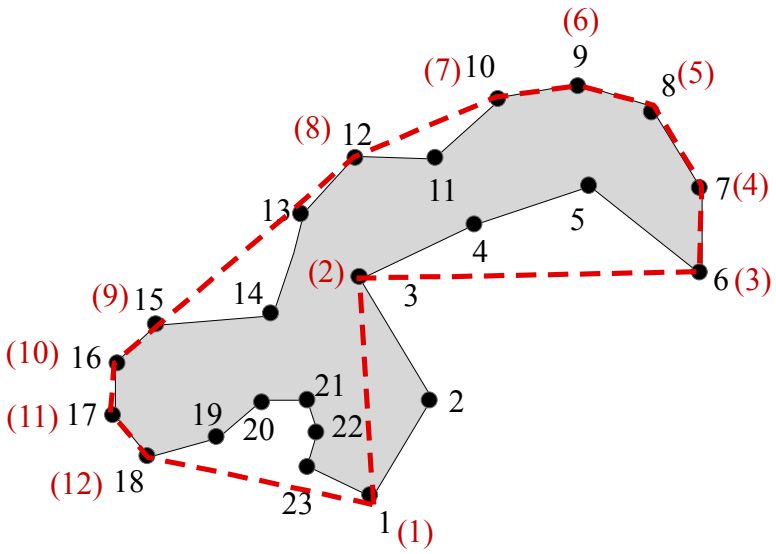

Figure 18. Concave point added to eliminate some non-hazard area captured by the convex hull

The convex hull algorithm recaptures all of the hazard area as shown in Figure 19.

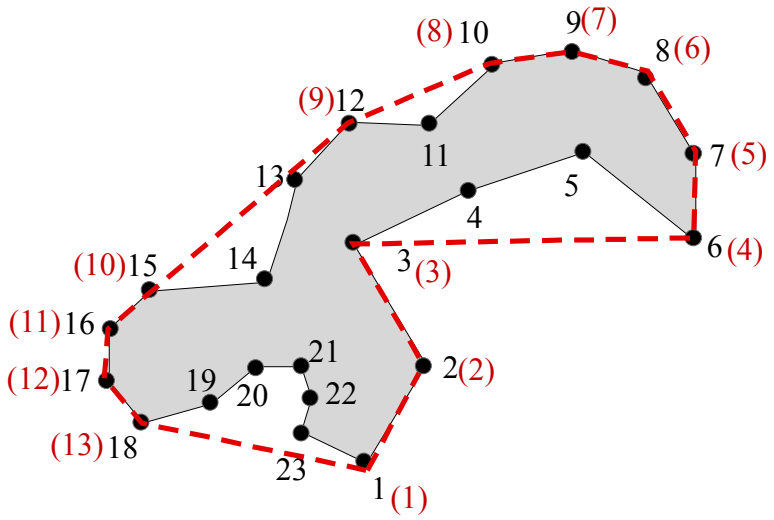

Figure 19. Convex hull algorithm recaptures all the hazard area

It is easy to exceed the maximum allowable number of vertices, if the tolerance is set too low, so the operator must specify how much area is worth recapturing. If at any time the number of allowable vertices is exceeded, the algorithm ceases operation and goes back to the last acceptable polygon state.

2.6.1 Concave smoothing

There are a few special cases that can occur that require special fixes. In some cases, the algorithm will recover a point that is a large distance from the edge, but isn't particularly useful. The situation is shown in Figure 20.

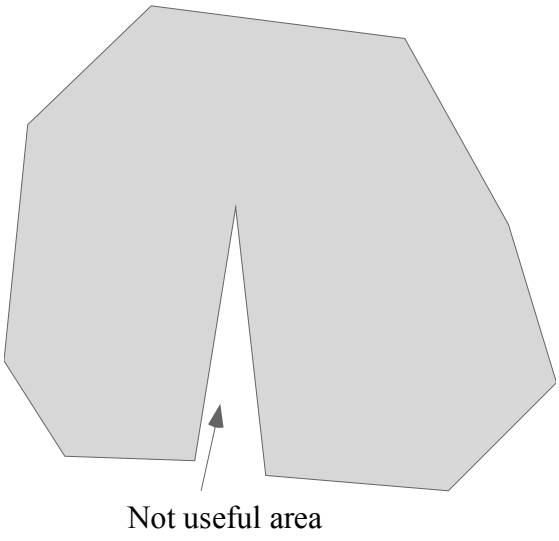

Figure 20. Convex hull algorithm recaptures all the hazard area

To correct the problem, unit vectors along each edge are calculated. If at any point along the polygon, the dot product of adjacent edges is smaller than a specified value (nominally -0.707 for 45 degrees), the point is removed, and a new edge replaces the two previous edges (see Equation (2.8)).

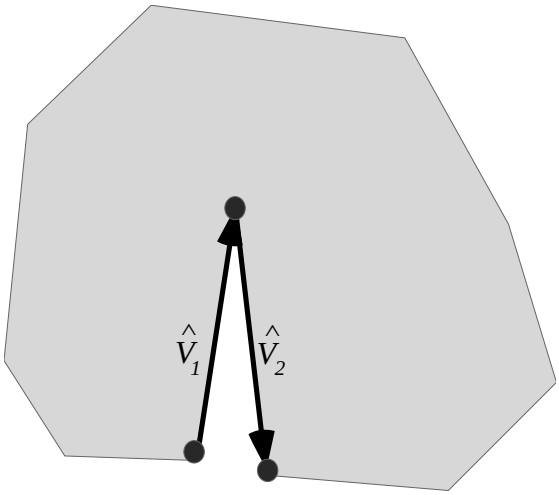

Figure 21. Unit vectors along each edge

$$\hat{V}_1 \cdot \hat{V}_2 < \tau_{tol} \tag{2.8}$$

3 WSI Pilot Brief Pro Emulator

To complete the suite of tools necessary to run human in the loop experiments, the government determined that a briefing tool that emulates typical tools available to pilots for pre-flight preparation should be developed. The emulator, using historic data, would provide a weather briefing appropriate for the historical weather scenario being used in the experiment. The platform chosen to emulate was the WSI Pilotbrief tool, which is a common tool available at airports for pre-flight planning.

Figure 22 shows the WSI Pilotbrief product. The WSI Pilotbrief product is a data link based product which collects a steady stream of weather data using a satellite antenna and a small receiver. The small receiver is actually a digital computer with a self contained hard drive designed to archive the data linked data.

Figure 22. WSI Pilot Brief advertising graphic (graphic removed)

The receiver is then networked with a desktop computer that contains the WSI Pilotbrief (display) software. WSI Pilotbrief is client-side application written in JavaScript and the WSI receiver is the server.

The data collected is mostly in the form of images in .jpg format with some textual data for the METAR, TAF, and area forecasts (FA). The display software receives the data and displays it in the proper window when requested by the operator. The display software is updated periodically as new data arrives. Since the data is in the format of images, rather than raw data, the data collected has little value for data analysis.

The WSI tool offers numerous weather products consisting of the following:

- WSI NOWrad® national and regional radar mosaic with 5 and 15 minute update options
- National and Regional Radar Summary charts indicating mesocyclones, tornadic vortex signatures, hail, and severe weather watch boxes
- 5-Minute Single Site Base Reflectivity data from more than 150 sites across the U.S
- High-resolution Infrared and Visible Satellite Imagery
- WSI AVcharts, Domestic and International
- Up-to-the-minute, Auto Plotted SIGMET and AIRMET charts depicting boundaries of in-flight advisories
- Automated Route Briefings
- Complete worldwide text weather data
- History loops and zooming for radar and satellite imagery with a single mouse click
- Dial-out access to free DUAT services (requires a telephone connection)

The NASA emulator emulates all but two of these features. The tool does no flight planning, so the Automated Route Briefings feature is not implemented. Neither is the DUAT services tool, which in the real system enables pilots to file flight plans with the FAA electronically. A screen shot of the NASA emulator is shown in Figure 23.

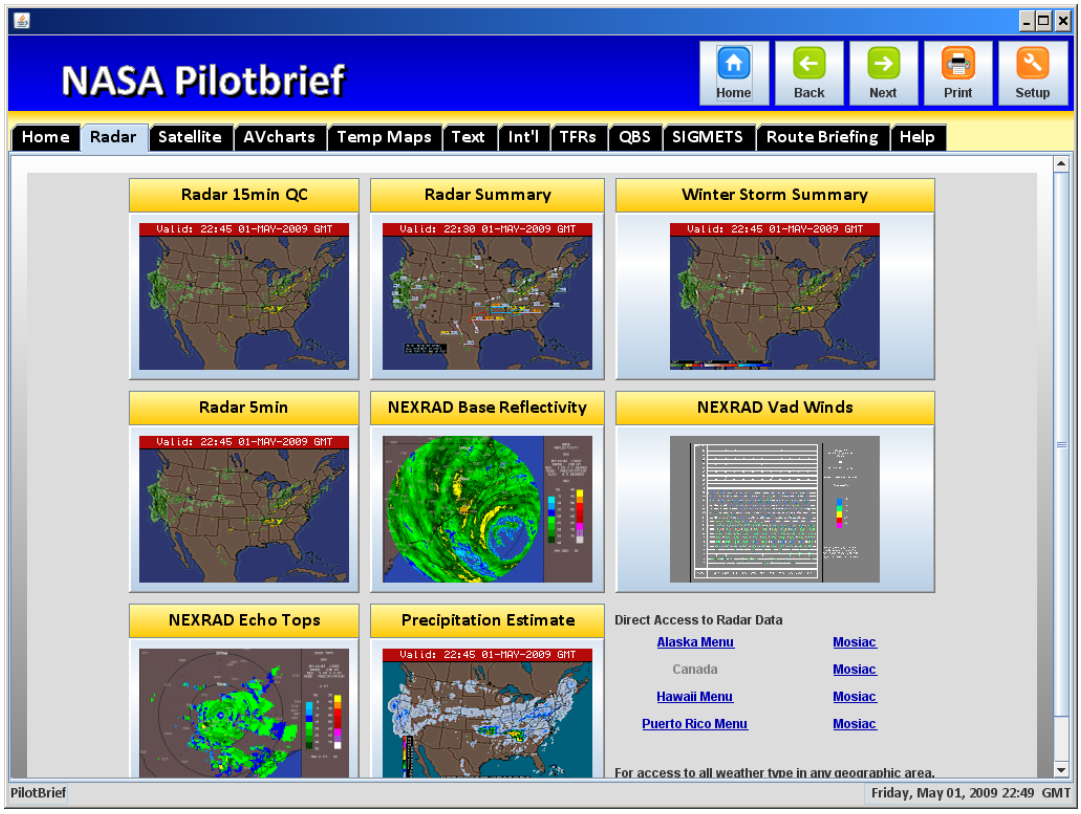

Figure 23. Screen shot of NASA Pilotbrief Emulator

To collect the data to drive the emulator, a special data collection interface was written which emulates the original Pilotbrief code, however, instead of displaying the data, the data is archived on a large hard drive. This software, the WSI Gateway recording software, creates time stamped content files. The amount of data collected each day amounts to roughly 3.5 GB compressed so a 6 month supply of data is roughly 641GB.

Table 1. Data collection rates

• ~3.5GB compressed daily
• 260,000 files (2200 folders) daily
• 6 months (641GB)
• 2 x 1TB External Disk Drive (Primary / Secondary)

A full user's guide for the software is contained in a separate document.

4 Convective Weather Simulation Tool

The convective weather simulation tool is designed to provide the researcher with a quick and easy means to create arbitrary weather scenarios, without having to rely on historical data. Simulated data permits researchers to run large scale Monte-Carlo simulations and investigate macroscopic effects that cannot be easily discerned from the limited number individual, manually-constructed scenarios.

True weather simulations based on the first principles of atmospheric physics are staggeringly complex and often require the use of super computers to model even the slightest aspects of a storm.[8] Therefore; the weather simulator was viewed as a high risk task in the overall the effort.

The method chosen to simulate convective activity was to qualitatively imitate the mechanisms that produce actual convective weather, though not in a rigorous manner. This extremely simplified model cannot be considered to be an actual physics-based model of convective weather. However, it produces results for two-dimensional radar reflectivity that are qualitatively similar in appearance to those from actual convective weather systems.

4.1 Simplified Qualitative Convection Model

The qualitative convection model represents the atmosphere as a two-dimensional array of air parcels, where the entire vertical column of air is modeled as a single parcel that has an associated altitude, vertical speed, temperature, water vapor content and liquid water content.

Each parcel of air has certain characteristics, and the prevailing environment in which the parcel exists has certain characteristics. In particular, the environment has a prevailing pressure versus altitude profile (generally set as the standard day profile.) The environment and the parcel each have a temperature versus altitude profile as well, and these temperatures (together with environmental pressure) determine the density of the environment and the parcel at the parcel's instantaneous altitude. The density difference between the environment and the parcel in turn determines an upward or downward buoyancy acceleration term for the parcel.

Each parcel also has a certain amount of water vapor present within it. When the parcel's temperature drops below the saturation temperature, water vapor proceeds to condense to form liquid water. Liquid water has two effects: it is the value upon which radar reflectivity is based; and, the presence of liquid water adds downward acceleration to the parcel. Liquid water also gradually dissipates (as precipitation); and any water vapor initially in the parcel, once converted to liquid water, is not replenished.

Downward or upward velocity of a parcel also influences that parcel's neighbors' vertical acceleration through a friction coefficient. This provides for spatial correlation of growth and diminishment of convective activity through the two-dimensional array of air parcels.

To produce disturbances that can lead to the growth of convection within the array of parcels, the environment also imparts semi-random vertical accelerations to the parcels in the simulation.

4.2 Equations for Simplified Convection Model

The following set of equations is used to model the vertical motion and evolution of characteristics of the parcels.

$$\dot{h} = v \tag{4.1}$$

$$\dot{v} = a_{env} - g\frac{\rho - \rho_{env}}{\rho} - k_w w + \sum_{i=1}^{8} k_f (v_i - v) \tag{4.2}$$

$$\dot{x} = -k_c (T - T_{sat}(x))(x - x_{sat}(T)) \tag{4.3}$$

$$\dot{w} = -\dot{x} - k_r w \tag{4.4}$$

where the state variables of the model are:

h = altitude of parcel (m)
v = vertical velocity of parcel (m/s)
x = specific humidity of parcel (nondimensional)
w = liquid water fraction of parcel (nondimensional)

auxiliary variables are

a_{env} = vertical acceleration imparted to parcel by environment (m/s^2)
ρ = parcel density (kg/m^3)
ρ_{env} = environmental density at parcel altitude (kg/m^3)
T = parcel temperature (C)
$T_{sat}(x)$ = parcel saturation temperature (as a function of specific humidity) (C)
$x_{sat}(T)$ = parcel saturation specific humidity (as a function of temperature) (ND)

and constants are

g = acceleration due to gravity (m/s^2)
k_w = downward acceleration due to parcel liquid water (m/s^2 per ND)
k_f = friction acceleration due to adjacent parcel velocity difference (m/s^2 per m/s)
k_c = condensation rate constant (ND/s per C-ND)
k_r = precipitation rate constant (ND/s per ND)

The above four equations will now be described in turn.

Equation (4.1) simply states that the rate of change of parcel altitude is the parcel's vertical velocity.

In Equation (4.2), the vertical acceleration of the parcel is taken as the sum of four terms:

1) Environmentally-induced vertical acceleration;
2) A buoyancy term due to the difference between parcel density and environmental density at the parcel's altitude and temperature;
3) A downward acceleration due to the presence of liquid water in the parcel; and
4) A friction acceleration taken as the sum of a constant times the difference in vertical velocity between the parcel and each of its nearest neighboring parcels (in general, each parcel has 8 neighbors; the exceptions being those parcels located on an edge or in a corner of the simulated array.)

Equation (4.3) provides a model for conversion of water vapor into liquid water.

Equation (4.4) states that liquid water accumulates due to condensation, and is removed from the parcel via precipitation.

These equations can be numerically integrated for any number of parcels, given any desired initial conditions.

The output of the model, radar reflectivity, is taken as proportional to the parcel liquid water content w. As the simulation proceeds, whenever the parcel temperature decreases below the saturation temperature, water vapor represented by a nonzero value for x will condense and thereby increase the value of w. Liquid water is thereby increased by condensation, but whenever nonzero, is also decreased by precipitation.

4.3 Lateral Boundaries of the Weather Cell

The simplified convection model tends to produce realistic looking behavior for the growth and decay of the storm. However, the model tends to produce weather all the way up against the boundaries of the two dimensional weather array. The result is a 'square' looking weather cell. To correct this anomaly, an activity factor is introduced. The activity factor is scalar value for each parcel that determines the geographic distribution of convective activity intensity. The value modulates the intensity of environmental disturbance, as well as the parcel's starting specific humidity. If it is zero, the algorithm skips integrating equations for that pixel. A value of 1.0 maximizes the activity. Setting the activity factor in a spatially-correlated manner, with appropriate randomization, results in realistic-looking spatial distribution of convective activity.

To automate the process of developing realistic-looking spatial distributions for the activity factor, a modified version of the clustering algorithm from the polygon code is

used. A single 'seed' pixel is chosen by the operator, and a value N_p is associated with the pixel. The larger the initial value of N_p, the larger the overall shape. The surrounding pixels are assigned their own respective values of N_p, based on the N_p value of their neighbors, in such a manner that N_p decreases with distance from the original seed pixel. Once N_p reaches zero, no more pixels are included in the cluster. Successive values of N_p are determined using Equation (4.5), where x_{ran} is a random number between 0.0 and 1.0, with a uniform distribution.

$$N_{p(k+1)} = N_{p(k)} x_{ran}^2 \tag{4.5}$$

Figure 24 contains a flow diagram that illustrates the process of creating the pixel distribution in more detail and Figure 25 shows an example of the pixel distribution created using the algorithm.

Once the pixel distribution has been determined, the N_p, values are discarded, and a uniform activity factor is assigned to all of the pixels. To achieve various levels of activity factor in the same weather cell, several activity factor distributions are superimposed onto the same weather array. A typical technique is to create a large wide spread area of lower intensity activity, and then include higher intensity regions within. This can be achieved using the weather cell editor, which enables the user to graphically manipulate the activity factor distribution. A simple example of this technique is shown in Figure 26.

For a complete discussion on the functionality and use of the weather simulation tool, a separate user's guide has been provided.

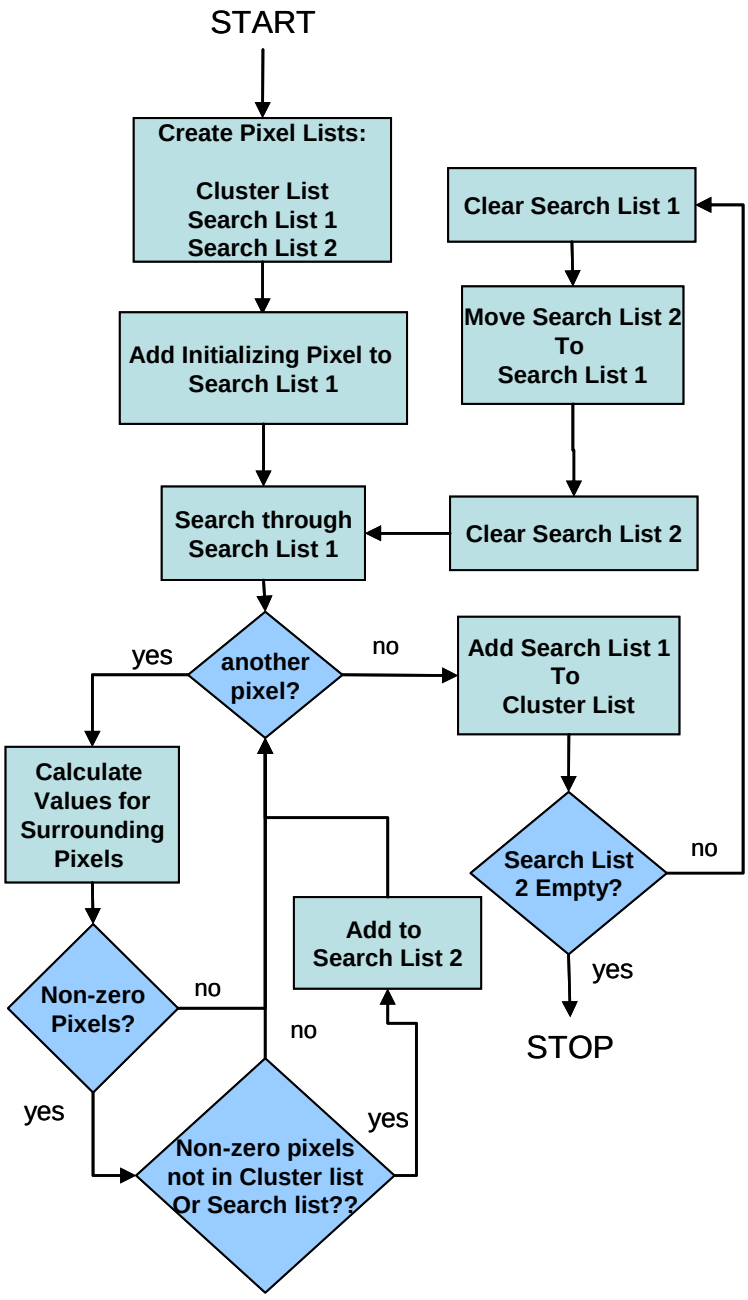

Figure 24. Process for generating a random spatial distribution of activity factor

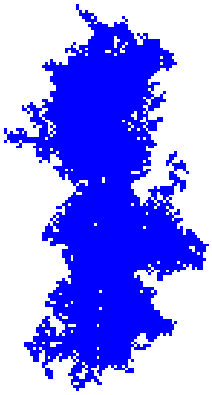

Figure 25. Example activity factor distribution

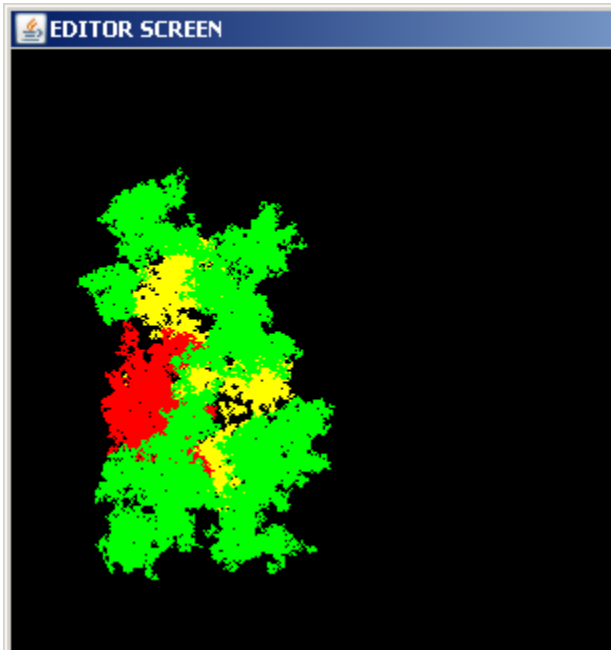

Figure 26. Example cell using multiple activity factor distributions

5 Concluding Remarks

Four main tasks were performed under this effort. These are:

- Data Conversion (polygonalization of the radar data)
- NASA Pilotbrief emulator
- Convective Weather Simulator
- High Impact Scenarios

As part of this effort, 3 software deliverables were made, which include the data conversion code for generating polygons, the NASA Pilotbrief Emulator, and the Convective Weather Simulator. The polygon code was written in C++ in the form of static libraries so that the government can include the algorithms in the AOP code. The other two products were written in JAVA as standalone tools. The Pilotbrief emulator was provided already installed and working as a standalone briefing station on a desktop computer.

6 References

1. National Research Council (NRC), 2003: Weather Forecasting Accuracy for FAA Traffic Flow Management: A Workshop Report. The National Academies Press, Washington, DC, 68 pp.

2. Hunter, G., Ramamoorthy, K., "Evaluation of the National Airspace System Aggregate Performance Sensitivity," 26th Digital Avionics Systems Conference, October, 2007.

3. Palmer, M. and Ballin, M., "A High-Performance Simulated On-Board Avionics Architecture to Support Traffic Operations Research," AIAA-2003-5452, Modeling and Simulation Technologies Conference Austin, TX, 2003.

4. Peters, M., Ballin, M., Sakosky, J., "A Multi-Operator Simulation for Investigation of Distributed Air Traffic Management Concepts,"AIAA 2002-4596, Modeling and Simulation Technologies Conference, Berkely CA, August, 2002.

5. Liu D. and Chung W., "ASTOR: An Avionics Concept Test Bed in a Distributed Networked Synthetic Airspace Environment," AIAA-2004-5259, Modeling and Simulation Technologies Conference, Providence, RI, August, 2004.

6. Finkelsztein, D., Lung, T., Vivona, R., Bunnell, J., Mielke, D., Chung, W., "Airspace And Traffic Operations Simulation for Distributed ATM Research and Development", AIAA 2005-6488, Modeling and Simulation Technologies Conference, San Francisco, CA, 2004.

7. Hunter, G., "The Engineering description of the Sensis STC Probabilistic Traffic Flow Management (TFM) Experimental Tool ," Internal Document, Sensis Corporation, 2006.

8. McGhee, E., "Scientists Use Powerful Cray Supercomputer to Develop Groundbreaking Strategies in Weather Prediction: Latest Computer Models Zoom Down to Level of Individual Storm Cells," Cray News Release, Marketwire Inc., August, 2007.

9. Pu, R., Gong, P., Li, Z., Scarborough, J., "A Dynamic Algorithm for Wildfire Mapping with NOAA/AVHRR Data," International Journal of Wildland Fire, 2004, Vol 13, pp 275-285.

10. Hoffman, J., **Numerical Methods for Engineers and Scientists**, McGraw-Hill, Inc., New York, 1992.

11. Bortins, R. and Hunter G. “Sensitivity of Air Traffic Control Automation System Performance to Storm Forecast Accuracy,” AIAA Guidance, Navigation and Control Conference, Boston, MA, August 1998.

12. Krozel, J., “Hazardous Weather Avoidance for Air Traffic Control Systems,” Tech. Report TR-98174-01, Seagull Technology, Los Gatos, CA, May 2000.

13. O’Rourke, J., **Computational Geometry in C, 2nd ed.** Cambridge, England: Cambridge University Press, 1998.