As of 10 June 2016

This GOES-R Level 2+ space weather Algorithm Theoretical Basis Document (ATBD) is preliminary and subject to change as the GOES-R Program prepares for a launch in late 2016.

The NOAA National Centers for Environmental Information (NCEI) is currently developing a demonstration version of the Satellite Product Analysis and Distribution Enterprise System (SPADES) which will host the L2+ algorithms. The operational SPADES will be instantiated by the National Weather Service in FY2017 to serve the needs of the Space Weather Prediction Center (SWPC).

Updated versions of the L2+ ATBDs will periodically be made available and posted to the SPADES website.

Bill Denig
NCEI

http://www.ngdc.noaa.gov/stp/space-weather/online-publications/stp_sii/spades/

NOAA NESDIS NATIONAL GEOPHYSICAL DATA CENTER

GOES-R SUVI
CORONAL HOLE BOUNDARIES
ALGORITHM THEORETICAL BASIS DOCUMENT

Version 0.2

TITLE: SUVI CORONAL HOLE BOUNDARIES ALGORITHM THEORETICAL BASIS DOCUMENT VERSION 0.2

AUTHORS:
Jonathan Darnel – CIRES/NGDC
Steven Hill – NOAA/SWPC
William Denig – NOAA/NGDC

APPROVAL SIGNATURES:

____<Actual Signature Date>
<Name of Technical Approver> Date
Technical Approver

____<Actual Signature Date>
<Name of Agency Approver> Date
Agency

SUVI CORONAL HOLE BOUNDARIES ALGORITHM THEORETICAL BASIS DOCUMENT
VERSION HISTORY SUMMARY

Version	Description	Revised Sections	Date
0.1	Created by Jonathan Darnel CIRES/NGDC	New Document	
0.2	Created by Jonathan Darnel CIRES/NGDC	All	Aug. 9, 2013
0.2′	Revised	All	12/11/13

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AIA	Atmospheric Imaging Array
AIT	Algorithm Integration Team
AT	Application Team
ATBD	Algorithm Theoretical Basis Document
AU	Astronomical Unit (~150,000,000 km)
AWG	Algorithm Working Group
CCD	Charge Coupled Device
CDR	Critical Design Review
CIRES	Cooperative Institute for Research in Environmental Sciences
CME	Coronal Mass Ejection
EIT	EUV Imaging Telescope
EUV	Extreme UltraViolet
GOES-R	Geostationary Orbiting Environmental Satellite – R series
HDR	High Dynamic Range
ICM	Iterated Conditional Modes
L1B	Level 1B (data)
L2	Level 2 (data)
MAP	Maximum *A priori* Probability
ML	Maximum Likelihood
MRD	Mission Requirements Document
NCEP	National Center for Environmental Prediction
NOAA	National Oceanic and Atmospheric Administration
NWS	National Weather Service
POV	Point Of View
SDO	Solar Dynamics Observatory
SEP	Solar Energetic Particle
SOHO	Solar and Heliospheric Observatory
STAR	Center for Satellite Apps and Research
SUVI	Solar Ultra-Violet Imager
SWPC	Space Weather Prediction Center
SWx	Space Weather
SXI	Solar X-ray Imager
TBD	To Be Determined

TBR	To Be Reviewed
XRS	X-ray Sensor

ABSTRACT

The GOES-R Solar Ultraviolet Imager (SUVI) will provide high-cadence full-disk images of the solar atmosphere in six narrow-band extreme ultraviolet (EUV) spectral channels that will support Space Weather forecast activities at NOAA's Space Weather Prediction Center (NOAA/NWS/NCEP/SWPC). This document provides a comprehensive description of the SUVI Coronal Hole Boundaries algorithm. The product of the SUVI Coronal Hole Boundaries algorithm is the results of several complex computations. The complexity is necessary due to the inherent difficulty of the underlying problem: asking a computer to perform image analysis to determine the properties of a human-defined feature.

To compute the SUVI Coronal Hole Boundaries product, the algorithm must ingest the most current SUVI Thematic Map (SUVI.19), identify individual coronal hole features within the thematic map, determine the boundary pixels for all coronal hole features identified, convert pixel coordinates to heliographic coordinates, and determine other useful quantities such as extremities and overall area.

Information necessary for developers and reviewers to verify that the algorithm meets operational requirements is presented. Also presented, when applicable, is traceability to heritage, and the design details necessary for development and implementation of the algorithm into operational use. Test and validation procedures are also provided along with assumptions and known limitations of the algorithm.

1 INTRODUCTION

The GOES-R Space Weather Team (SWxT) at the National Geophysical Data Center (NOAA/NESDIS/NGDC) is responsible for producing algorithms that generate products using space environment data from the GOES-R series satellites. These algorithms shall meet the operational needs of the NOAA/NWS/NCEP Space Weather Prediction Center (SWPC) to observe and forecast space weather conditions impacting near-earth systems such as satellites, communications, electrical power grids, manned space missions and many others. They shall also meet the requirements of outside operational and research agencies as agreed.

1.1 Purpose of This Document

This Algorithm Theoretical Basis Document (ATBD) details of the GOES-R Solar Ultraviolet Imager (SUVI) Coronal Hole Boundaries algorithm design and processing. The ATBD provides operational requirements for this product and defines how these requirements will be met using this algorithm. The algorithm inputs, processing, and outputs are described in enough detail to design, develop, test and implement the necessary processing software and storage mechanisms.

1.2 Who Should Use This Document

The members of the Space Weather Forecast Office shall use this ATBD to verify their operational requirements are being met by the proposed algorithm. They should also use it to understand the strengths and weaknesses of the algorithm as well as its applicability, accuracy, and robustness. It shall also be used by the prime development and implementation contractor to design, develop, test, validate and implement the algorithm into the final operational processing system.

1.3 Inside Each Section

Section 2.0 OBSERVING SYSTEM OVERVIEW:

- provides objectives of the SUVI Coronal Hole Boundaries algorithm including the output and how they may be used operationally;
- discusses SUVI instrument characteristics and the fundamental algorithm approach;

Section 3.0 ALGORITHM DESCRIPTION:

- contains a complete SUVI Coronal Hole Boundaries algorithm description, including an outline of the processing, input data, and a theoretical description;
- provides estimates of the algorithm performance and output quality;
- reviews the numerical computation, programming and procedural issues and a description of how the algorithm has been validated;

Section 4.0 TEST DATA SETS AND OUTPUTS:

- describes the test data sets used to characterize the performance of the algorithm and quality of the data product(s);
- discusses the results from algorithm processing on simulated input data;

Section 5.0 PRACTICAL CONSIDERATIONS:

- discusses issues involving numerical computation, programming and procedures, quality assessment and diagnostics and exception handling at a level of detail appropriate for the current algorithm maturity;

Section 6.0 ASSUMPTIONS AND LIMITATIONS:

- describes all assumptions concerning the SUVI Coronal Hole Boundaries algorithm theoretical basis and performance;
- discusses planned product improvements for future enhancements;

Section 7.0 REFERENCES:

- Provides references to all sources cited in the ATBD.

Appendix A ONE METHOD FOR CLUSTERING/LABELING

- Describes a simplistic method for aggregating contiguous pixel regions together

Appendix B NEXT NEAREST-NEIGHBOR PIXEL ORDERING

- Describes the next nearest-neighbor ordering method

Appendix C VERTEX REDUCTION

- Describes a method of reducing the number of vertices necessary to describe a 2d region

Appendix D DELAUNAY TRIANGULTION

- Describes the method of creating a set of triangles from a set of points

Appendix E CONVERSION TO HELIOGRAPHICAL COORDINATES

- Describes the method to convert on-disk pixel coordinates to heliographic latitude and longitude

1.4 Related Documents

GOESR SUVI Coronal Hole Boundaries Test Plan and Results
GOESR SUVI Coronal Hole Boundaries Implementation and User's Guide

1.5 Algorithm Requirements and Traceability

Table 1: SUVI.15 Algorithm Requirements

Product: SUVI.15 Coronal Hole Boundaries		
Reference: GOES-R_Level 2+ SWx Algorithm Requirements Phase 3 07262012.xlsx		
Revision: Revised to reflect changes recommended during and following the SUVI Coronal Hole Boundaries CDR, 06/19/13.		
1. General		
SUVI.15.01.01	The algorithm shall identify coronal holes in the solar EUV imagery and provide including boundary vertices, extents, and estimates of area.	
SUVI.15.01.02	The algorithm shall identify from the thematic map (SUVI.19) distinct coronal holes greater than 25 square arcseconds in size based upon pixel classification.	
SUVI.15.01.03	The algorithm shall meet a daily cadence.	
SUVI.15.01.04	The algorithm shall perform all operations in less than 4 minutes after all necessary inputs have been ingested.	
SUVI.15.01.05	The algorithm shall report all locations with an accuracy of no more than 5heliographic degrees from the true locations.	
2. Algorithm Inputs		
SUVI.15.02.01	The algorithm shall ingest all necessary inputs to detect coronal holes from	

	the designated SUVI.19 Thematic Map.	
SUVI.15.02.02	The algorithm shall ingest all necessary inputs to perform the pixel to heliographic coordinate conversion.	
3. Processing		
SUVI.15.03.01	The algorithm shall perform all processing as a callable method or object.	
SUVI.15.03.02	The algorithm shall identify all distinct coronal holes from the thematic map.	
SUVI.15.03.03	The algorithm shall determine a set of vertices to describe each distinct coronal hole.	
SUVI.15.03.04	The algorithm shall convert vertex coordinates from pixel to heliographic latitude and longitude.	
SUVI.15.03.05	The algorithm shall estimate the area of each coronal hole in heliographic square degrees.	
SUVI.15.03.06	The algorithm shall determine the extents of each coronal hole.	
4. Algorithm Outputs		
SUVI.15.04.01	The algorithm shall report tables of heliographic latitude and longitude vertices for each distinct coronal hole.	
SUVI.15.04.02	The algorithm shall report vertices in Stonyhurst and Carrrington heliographic coordinates.	
SUVI.15.04.03	The algorithm shall report no fewer than 6 and no more than 16 vertices for each distinct coronal hole.	
SUVI.15.04.04	The algorithm shall report for each coronal hole an estimate of the coronal hole area.	
SUVI.15.04.05	The algorithm shall report for each coronal hole the north, south, east, and west extents in heliographic coordinates.	
5. Error Handling		
SUVI.15.05.01	The algorithm shall report an error indicating when a thematic map more recent than 6 hours cannot be found.	
SUVI.15.05.02	The algorithm shall report an error indicating when data necessary to the conversion from pixel to heliographic coordinates is not found.	
6. Ancillary Data		
SUVI.15.06.01	The algorithm shall, if necessary, ingest the *Astronomical Almanac* data to perform the conversion from pixel to heliographic coordinate from an external data file.	

1.6 Revision History

Revision Number	Date	Author	Revision Description	Reason for Revision
0.2′	12/11/13	J. Darnel	Additions	Comments

2 OBSERVING SYSTEM OVERVIEW

2.1 Product Generated

A SUVI Coronal Hole Boundary is a table of Heliographic Latitude and Longitude pairs (Stonyhurst and Carrington), and an estimate of heliographic area.

2.2 Instrument Characteristics

The SUVI instrument operational requirements are detailed in section 3.4.2.4 of the GOES-R Series Mission Requirements Document (MRD) Version 3.9, dated November 2009. The SUVI Coronal Hole Boundaries algorithm has as an essential input another Level 2 product: the SUVI thematic map (SUVI.19). The Coronal Hole Boundaries algorithm assumes all inputs to the thematic map meet these minimum requirements, although in all likelihood, the delivered instrument performance will exceed these requirements. To start, individual SUVI spectral channel specifications are repeated in Table 2 for ease of reference, along with the type of EUV solar feature that motivated each spectral channel choice.

Table 2: SUVI image spectral channels

Channel	Wavelength	Sample Use
SUVI (Fe XVIII)	9.39 nm	Flares (~$6x10^6$ °K)
SUVI (Fe VIII)	13.1 nm	Flares (~$10x10^6$ °K) and Hot flares (~$15x10^6$ °K)
SUVI (Fe IX)	17.1 nm	Active regions (~$6x10^5$ °K)
SUVI (Fe XII)	19.5 nm	Active regions (~$1x10^6$ °K) and Hot Flares (~$20x10^6$ °K)
SUVI (Fe XV)	28.4 nm	Coronal holes (~$2x10^6$ °K)
SUVI (He II)	30.4 nm	Filaments (~$6x10^4$ °K)

The spatial resolution of a SUVI image is 2.5 square arcseconds, and the SUVI field-of-view requirement is 1.3 solar radii. It was thus decided that the SUVI image array would consist of 1280x1280 pixels to meet the field-of-view requirement. A measurement range of 0.3-10^6 photons/s/cm^2/$arcsec^2$ is specified for SUVI by the MRD to cover the range of features listed in Table 1. A single exposure is not capable of meeting this MRD requirement given SUVI's hardware limitations. A separate Level 2 SUVI data product, SUVI.07, utilizing multiple exposures to create high dynamic range composite images will be used to meet the required measurement range as input to the higher-level data products.

While SUVI has not yet flown aboard any NOAA satellite, the Solar Dynamics Observatory (SDO) was launched in the spring of 2010, and it carries the Atmospheric Imaging Array (AIA), a scientific instrument that has mostly compatible spectral channels to SUVI, and a very similar optical design and hardware selection. Details about the SUVI proxy data generated from SDO/AIA images will be presented later, but Figure 1 offers a preview of the kind of solar image data that will be returned by GOES-R SUVI, and that will serve as input for the SUVI thematic map algorithm.

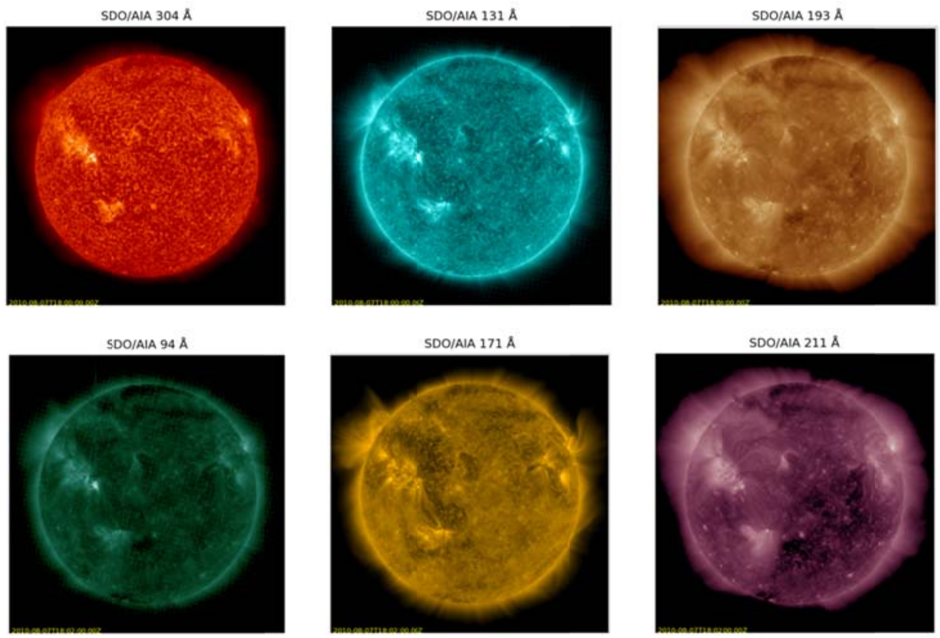

Figure 1 SUVI proxy data
Six select EUV channels from the Solar Dynamics Observatory (SDO) Atmospheric Imaging Array (AIA) are spatially and temporally binned/re-sampled to closely match SUVI's expected characteristics. The spectral band passes also nearly match those expected on SUVI, with the exception of SUVI's 28.4 nm channel, which we have chosen to replace with AIA's 21.1 nm channel for development and testing purposes.

3 ALGORITHM DESCRIPTION

3.1 Algorithm Overview

The SUVI Coronal Hole Boundaries algorithm uses as input the SUVI.19 Thematic Maps Level 2 product. From the SUVI Thematic Map, the features known as coronal holes are easily identified. The boundaries to those features are derived, and a method to identify the most critical vertices to describe each coronal hole is used to restrict the number of reported boundary points as required for the algorithm. These points are then converted to heliographic coordinates and reported along with an approximate estimate of the area of the coronal hole.

3.2 Processing Outline

Steps required to construct SUVI Coronal Hole Boundaries are:

1. Pre-Processing of the SUVI.19 Thematic Map product:
    a. Read in thematic map data
    b. Read in thematic map metadata
    c. Read in *Astronomical Almanac* solar angle data either from the thematic map metadata or an external data file
    d. Differentiate individual coronal hole features in thematic map data
2. Process Coronal Hole data:
    a. Determine Coronal Hole Boundary pixels
    b. Determine vertices by finding extrema in CH Boundary pixels
    c. Convert vertices to heliographic latitude and longitude
    d. Calculate heliographic area
    e. Repeat 2a-d until no more coronal hole features remain unprocessed.
3. Write out the Coronal Hole Boundary product with associated metadata

The process flow is depicted graphically in Figure 2.

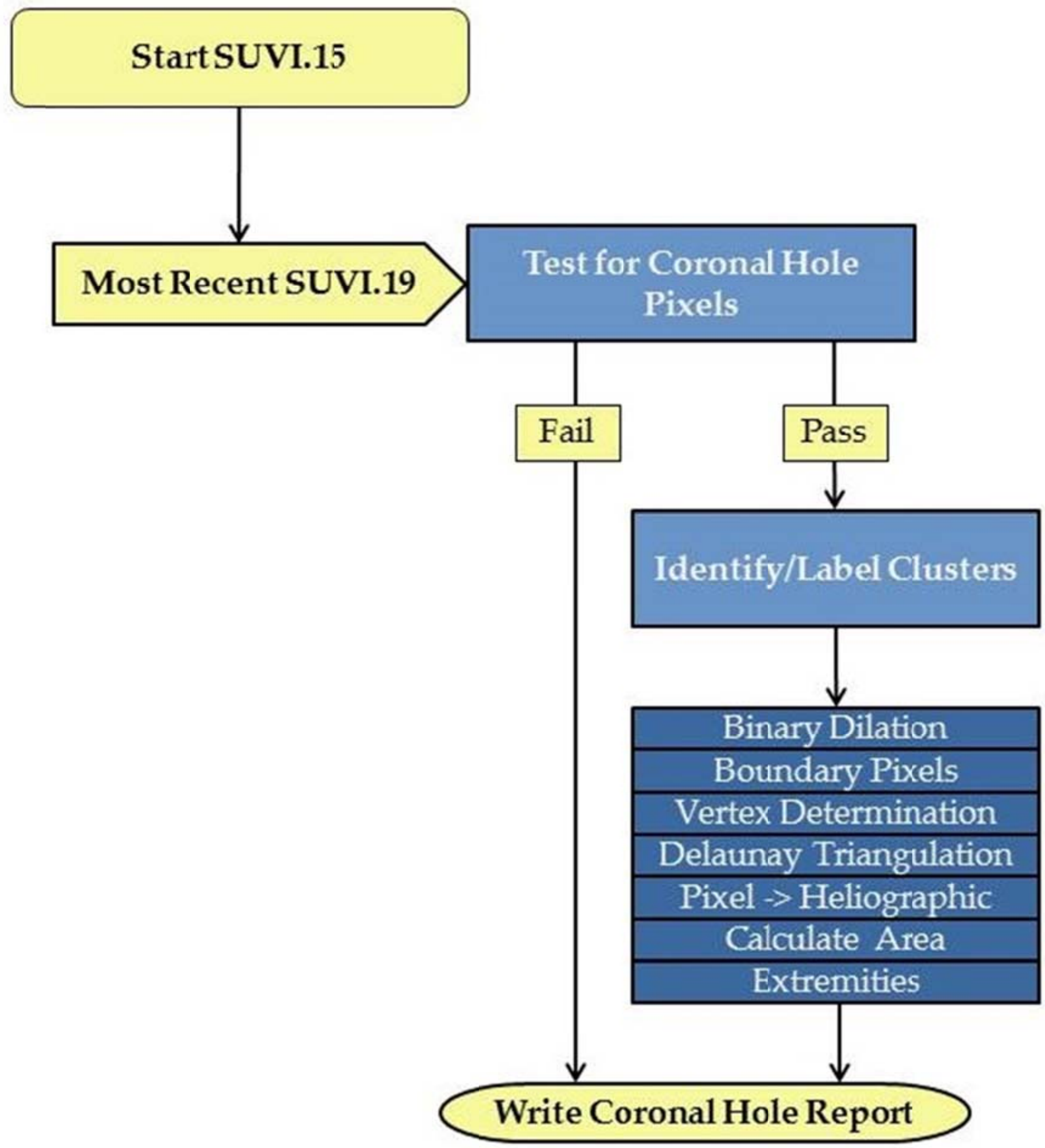

Figure 2 Flow diagram of SUVI Coronal Hole Boundaries algorithm
This flow diagram depicts at a high level steps required to generate a SUVI Coronal Hole Boundaries product.

3.3 Algorithm Input

3.3.1 Primary Sensor Data

The SUVI Coronal Hole Boundaries algorithm requires as its primary input the SUVI.19 Thematic Map, a separate level 2 SUVI data product. A listing of the information expected in any input image is provided in Table 3.

Table 3: SUVI L2 inputs to the coronal hole boundaries algorithm

Name	Description	TYPE (Dimension)
Date/Time	Image date and time tags (e.g. YYYY,MM,DD,hh,mm,ss.sss)	CHAR(1) or Integer(7)
Pixels	Thematic Map pixel values	REAL (n_x, n_y)
Δ_{pix}	Pseudo-angle describing a pixel's plate scale in the X & Y directions	REAL (2)
I_{rot}	Angle of instrument rotation relative to the satellite	REAL (1)
S_{rot}	Angle of the satellite's rotation relative to terrestrial north	REAL (1)
L_0	Carrington longitude of Earth	REAL (1)
ϕ_S	Stonyhurst longitude of SUVI (0 if SUVI and Earth are co-located)	REAL (1)
D_S	Distance from sun center to SUVI	REAL (1)
(x_Θ, y_Θ)	Solar Center	INTEGER(2)

3.3.2 Ancillary Data

Ancillary data are assumed to be data not generated by SUVI or the GOES-R spacecraft. For the Coronal Hole Boundaries product, the data that are needed are quantities from the Astronomical Almanac that describe the orientation of the Sun as seen here on Earth. These quantities vary throughout the year, and may not be available electronically. It is suggested that action is taken to ensure that the ancillary data listed in Table 3 be made available in the SUVI.19 product, which is an input to the Coronal Hole Boundaries algorithm.

Table 4: Ancillary inputs to SUVI Coronal Hole Boundary algorithm

Name	Description	TYPE (Dimension)
P_0	Position angle between solar north terrestrial north	REAL (1)
β	Heliographic latitude of SUVI	REAL (1)

3.4 Theoretical Description

The Coronal Hole Boundaries algorithm is essentially performing image analysis upon the SUVI.19 Thematic Map product and assumes that the information presented within the Thematic Map is truth. The Thematic Map provides a classification of the solar environment into a finite number of features. This is represented by a 2-D array of integers (currently 0-7) that correspond to certain solar environment features (Outer Space, Quiet Sun, etc.). The Coronal Hole Boundaries algorithm uses the Thematic Map to define individual coronal hole features and perform the necessary analysis upon each.

3.4.1 Physics of the Problem

Due to the nature of the problem, the Coronal Hole Boundaries algorithm involves using image analysis to describe the coronal hole structures within the Thematic Map product. Thus, the "problem" of defining coronal hole boundaries is not based upon physics, but image processing. The physics of the pixel classification has been addressed in the SUVI Thematic Map ATBD.

3.4.2 Mathematical Description

The SUVI Coronal Hole Boundaries algorithm takes advantage of the definitiveness of the SUVI Thematic Map to identify the coronal hole features on the solar disk. Since the Thematic Map uses a finite set of integers to represent the classification of individual pixels into solar features, it is a fairly straightforward procedure to 1) find the coronal hole elements and 2) differentiate the separate coronal holes. Table 4 contains the possible values of the Thematic Map elements.

Feature Name	Thematic Map value
Outer Space	0
Coronal Hole (on disk)	1
Coronal Hole (off disk)	2
Quiet Corona	3
Prominence of Filament	4
Active Region	5
Flare	6

Table 5: Thematic Map features and corresponding values

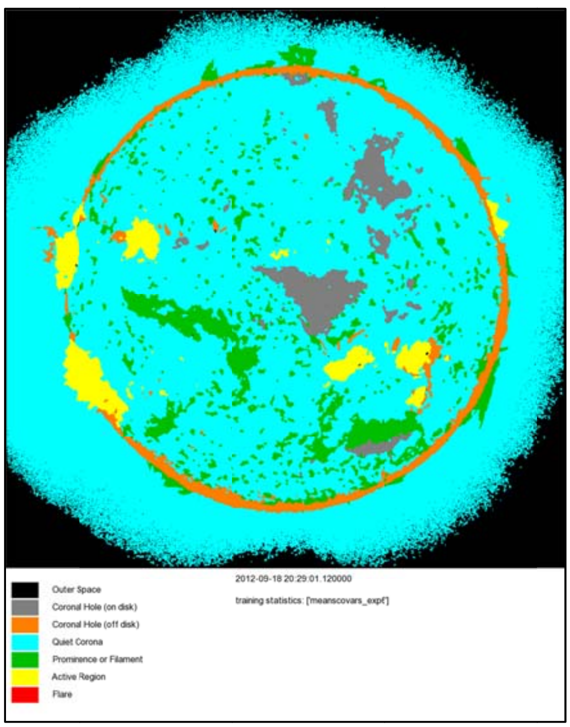

Figure 3: Sample Thematic Map from the SWPC Proving Grounds

Currently included within the Thematic Map product created by the Space Weather Prediction Center's Proving Ground is a table of the features whose order coincides with the feature pixel value. Thus, if the values happen to change or the order shifts, algorithms such as SUVI Coronal Hole Boundaries will be able to adapt to those changes. If, as seen in Table 4, the on-disk coronal holes pixels all have a value of "1" we can then single out all the coronal hole pixels by setting all others to "0":

$$A_{TM}[A_{TM} \neq 1] = 0$$

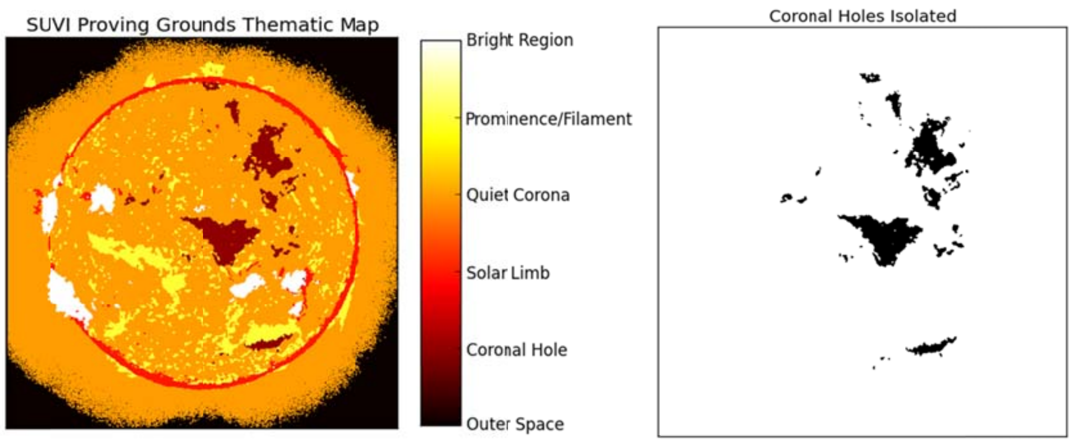

Figure 4: (*left*) SWPC PG Thematic Map before isolating coronal hole pixels. (*right*) After isolating coronal hole pixels, all coronal hole pixels stand out and are easily identifiable.

After setting all non-coronal hole pixels to the value "0", it is fairly easy to isolate individual coronal hole features. Figure 3 shows a coronal hole feature identified in the thematic map that is near the center of the solar disk.

In order to efficiently isolate every coronal hole feature, we currently use an image processing routine called "labeling". Similar routines can and have been written from scratch. It should be noted, however, that built-in routines tend to

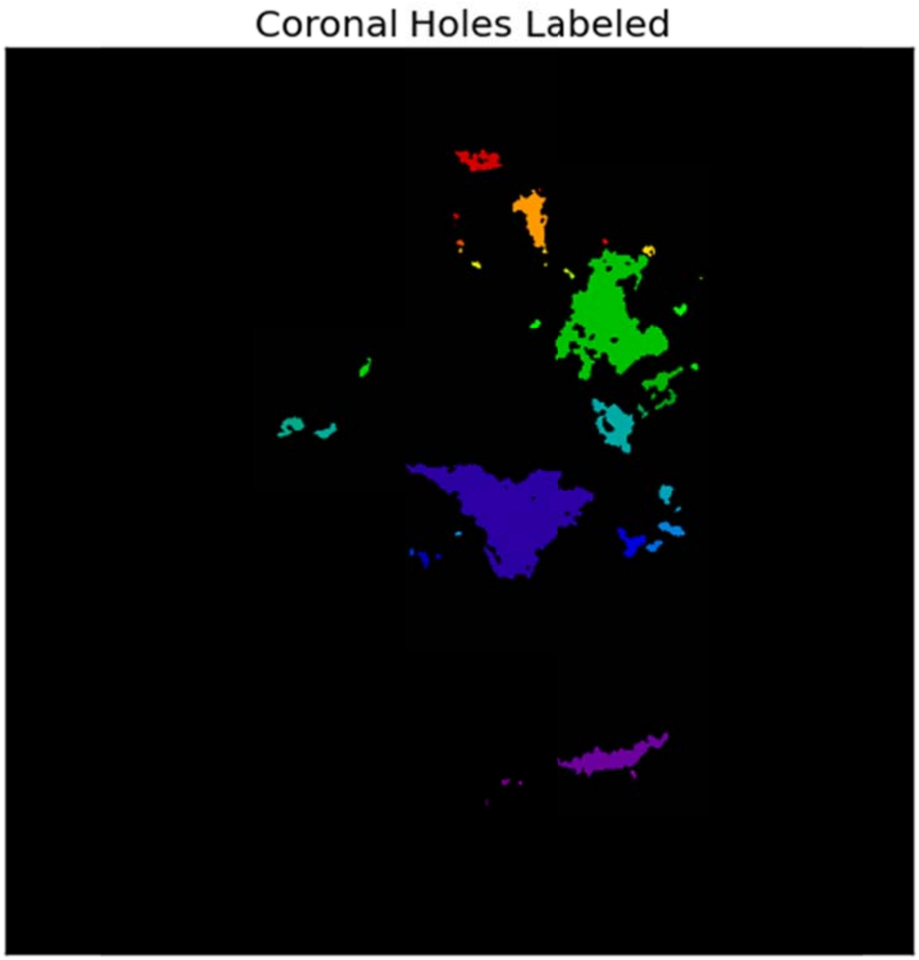

Figure 5: Coronal Hole features in the thematic map are labeled (indicated by different colors).

be more highly optimized and efficient than those written from scratch. We describe one strategy to label individual clusters in **Appendix A**.

Once separate coronal holes have been identified and labeled from the thematic map, one can define boundaries and work towards the Coronal Hole Boundaries product. Let us take as a starting example a single isolated coronal hole (Figure 6):

Figure 6: An isolated coronal hole

To establish the boundaries for any coronal hole, a *binary dilation* is performed:

$$A'_{TM}[i,j] = \begin{cases} 1 \; if \; A_{TM}[i \pm 1, j \pm 1] = 1 \\ 0 \; otherwise \end{cases}$$

That is, if any of the cardinal neighbors are nonzero, then that pixel is also nonzero. This has the effect of growing the coronal hole by a pixel in each direction. Subtracting the new coronal hole by the original leaves only the boundary pixels that were added by the dilation operation (Figure 7):

$$A_{DIFF} = A_{TM} - A'_{TM}$$

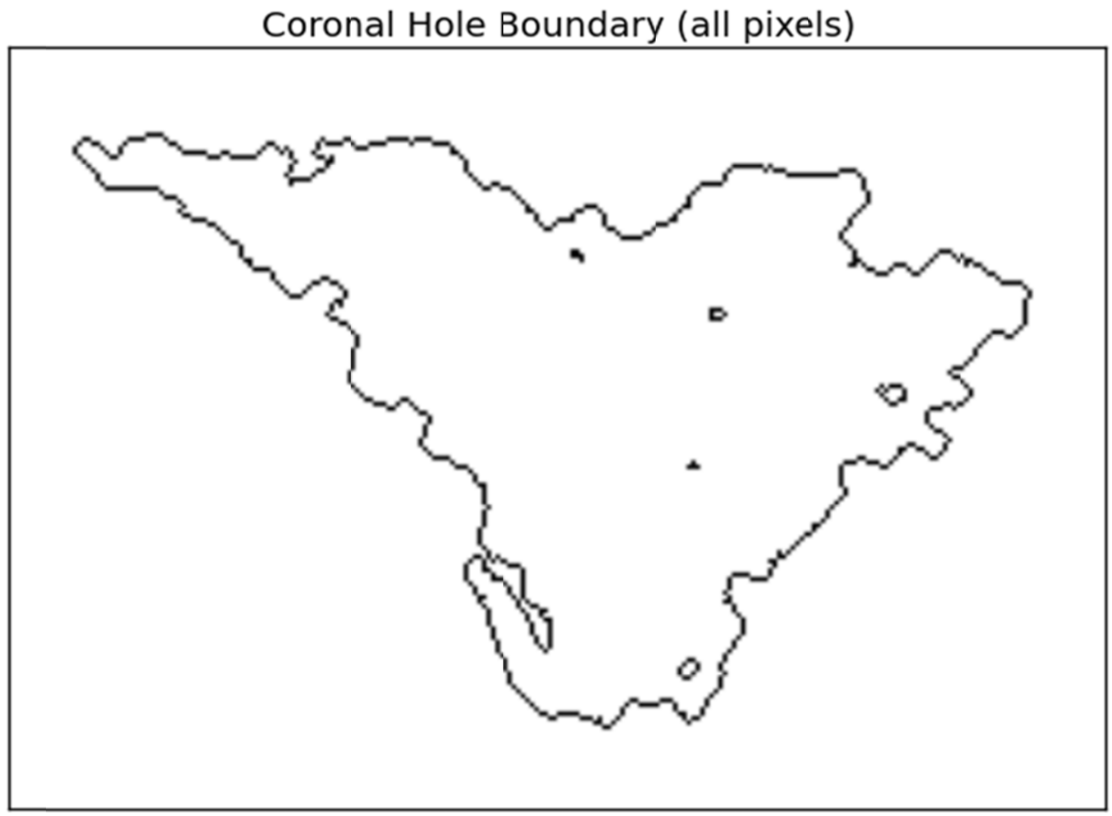

Figure 7: Coronal Hole Boundary pixels

These boundary pixels are easily read out into a list of indices. Most programming languages will not read the indices of the boundary pixels out in a form that describes a path around the shape of the coronal hole. The order of the pixels proceeds by taking each row in turn and marching through the columns for that row before moving on to the next row and repeating the process. In order to re-order the pixel list, a next-nearest-neighbor procedure is used. **Appendix B** describes the read-out order native to most programming languages in more detail and then describes the next-nearest neighbor reordering procedure.

To determine the extents of each coronal hole, we determine the coordinates of the min and max in each of the x/y directions:

$$Extents = \{p[x_{\min(x)}, y_{\min(x)}], p[x_{\max(x)}, y_{\max(x)}], p[x_{\min(y)}, y_{\min(y)}], p[x_{\max(y)}, y_{\max(y)}]\}$$

The newly ordered boundary pixel list can contain hundreds to over a thousand elements for larger coronal holes. The boundary pixel list needs to be reduced in size to a finite number that can still accurately describe the coronal hole. This finite number for vertices is currently 16, but should be flexible in case the needs of the SWFO changes. Thus, it is recommended that the operational version of the SUVI Coronal Hole Boundaries algorithm should allow for either a run-time option to set the number of vertices to something other than the default number of vertices, or a writeable configuration file that contains the number of vertices. A detailed explanation of how to reduce the number of vertices for the final product is presented in **Appendix C**.

In order to get an estimate of the area with any accuracy, it is best to use the vertices to generate a set of triangles. An easy way to achieve this set is to perform something known as Delaunay Triangulation. This is a triangulation method for a finite set of points that can be quite complex. Delaunay Triangulation is discussed in **Appendix D**.

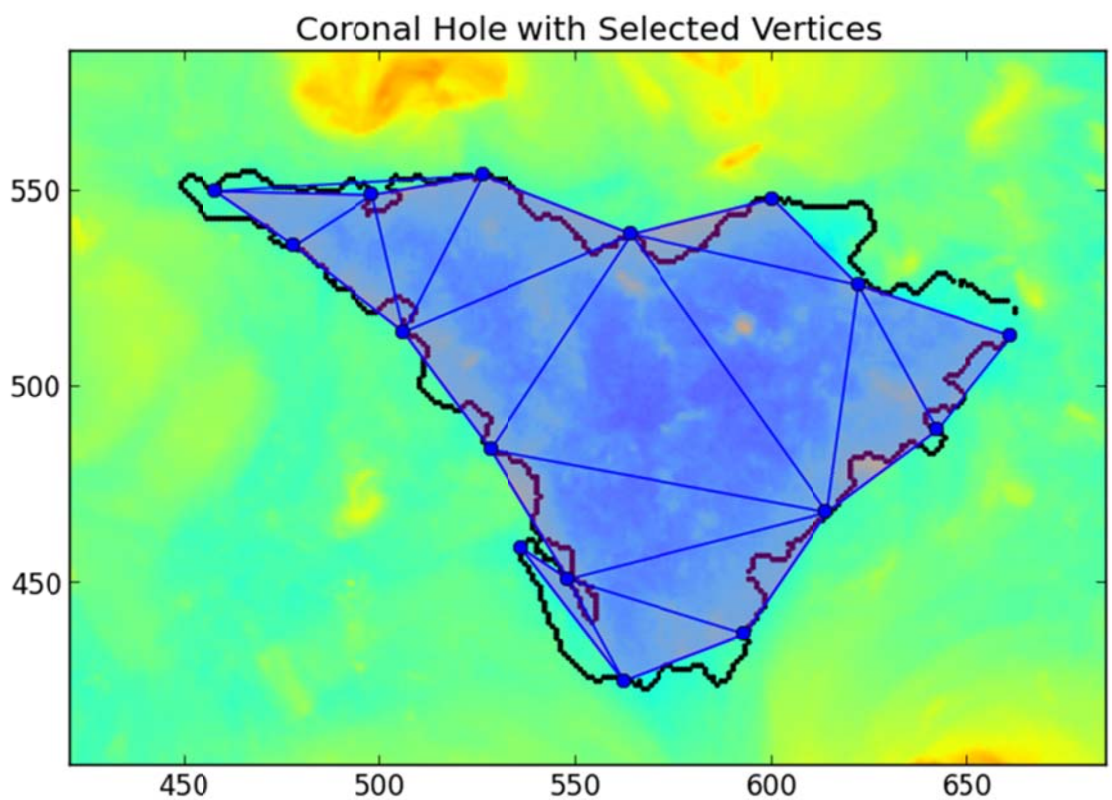

Figure 8: Coronal Hole with reduced vertices.

The next step to produce the SUVI Coronal Hole Boundaries product is to now convert the points stored in the vertex table, extents table, and the triangulation table into heliographic coordinates. This conversion is dependent upon many quantities: the solar P and β angles, the distance from the Sun, and the rotation of the instrument relative to the ecliptic. The solar P and β angles are retrieved by using the observation date to reference against a Look-Up Table (LUT) of Astronomical Almanac values or a forward model. All these quantities, the distance to the Sun, instrument rotation and the solar P and β angles, need to be included in the SUVI.19 product metadata. See **Appendix E** for an explanation on how to perform the conversion to heliographic coordinates.

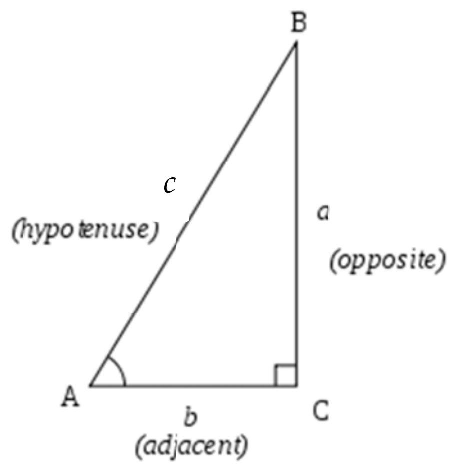

Figure 9: Example triangle

To calculate the heliographic area of the features, it is best to utilize the triangulation performed earlier. For each coronal hole feature, the areas of each triangle are summed to approximate the area of the entire coronal hole using Heron's Formula where a, b, and c are edges of the triangle. We sum up the areas of all the triangles that constitute the coronal hole to obtain an estimate of the total coronal hole area:

$$Area = \sum_{i=0}^{n} 0.25 * \sqrt{(a_i + b_i - c_i) * (a_i - b_i + c_i) * (-a_i + b_i + c_i) * (a_i + b_i + c_i)}$$

3.4.3 Algorithm Output

The SUVI Coronal Hole Boundaries algorithm generates two Nx2 arrays (shape files) of coronal hole boundaries vertices in heliographic coordinates for each coronal hole region identified from the SUVI.19 product. In addition, the algorithm will provide area estimates for each coronal hole in terms of heliographic square degrees and the extents of each coronal hole. Metadata will also need to be added to the product that documents the chain of production from the SUVI L1b product to the SUVI Coronal Hole Boundaries product. It is also advisable to document in the metadata the version of the operational algorithm and configuration file that was used to produce the product. All outputs are listed in Table 6.

Table 6: Level 2 outputs of SUVI thematic map algorithm.

Name	Description	TYPE (Dimension)
Date/Time	Date and time tags of latest composite image	CHAR (1)
SUVI.19 product	Filename of the Thematic Map used in the SUVI.15 algorithm	CHAR (1)
$\mathbf{N_{features}}$	Number of Coronal Holes identified from the Thematic Map	INT (1)
$(\varphi,\lambda)_{Stonyhurst}$	Stonyhurst Latitude and Longitude pairs	FLOAT $(N_{features})*(N,2)$
$(\varphi,\lambda)_{Carrington}$	Carrington Latitude and Longitude pairs	FLOAT $(N_{features})*$ (N,2)
$(\varphi,\lambda)_{Extents,\ Stonyhurst}$	Stonyhurst Latitude and Longitude pairs for NWES extents	FLOAT $(N_{features})*$ (4,2)
$(\varphi,\lambda)_{Extents,\ Carrington}$	Carrington Latitude and Longitude pairs for NWES extents	FLOAT $(N_{features})*$ (4,2)
$\mathbf{A_{est}}$	Area of the coronal holes	FLOAT $(N_{features})$

4 TEST DATA SETS AND OUTPUTS

4.1 Simulated/Proxy Input Data Sets

The SUVI.15 Coronal Hole Boundaries algorithm uses as the sole input the SUVI.19 Thematic Map. Thus, any proxy data for the SUVI.15 algorithm has to be in the form of a thematic map. The best configuration would be thematic maps that are at the same resolution as the SUVI instruments and use the same wavelengths. The ideal proxy data are thematic maps generated by demonstration implementations of the SUVI.19 product using the synoptic data from the SDO/AIA EUV instruments.

The Solar Dynamics Observatory (SDO) Atmospheric Imaging Array (AIA) was launched in the spring of 2010, and has a very similar design, both optical and electronic, to SUVI. The important differences are:

- SDO/AIA is comprised of four separate telescopes, compared to SUVI's single telescope, quadrupling the rate at which images can be acquired;

- each SDO/AIA primary mirror has ~4x the photon collecting area of the SUVI primary mirror for each spectral channel;
- the SDO/AIA CCD is 4096x4096 pixels, compared to the SUVI CCD's ~1000x1000 pixels;

The SDO/AIA synoptic images, however, have been rebinned to 1024x1024 pixels and are in physical units instead of digital number (DN). The resolution of the SDO/AIA synoptic images is very similar to the SUVI resolution and thus the synoptic images are useful in creating thematic map without further processing.

For the purpose of developing, testing, and demonstrating a prototype of the SUVI Coronal Hole Boundaries algorithm a collection of thematic maps generated during different solar events and demonstrating different solar features should be used. The purpose is to demonstrate that the Coronal Hole Boundaries algorithm will function as intended regardless of solar activity. The results of the testing and demonstration will be compared against available results of well-established coronal hole defining algorithms such as the SDO/AIA Feature-Finding Team's Coronal Hole chain codes and Dr. Krista's CHARM model. In addition, the results will be reviewed by experts in solar feature classification.

4.2 Output from Simulated/Proxy Inputs Data Sets

To Be Added.

4.2.1 Precisions and Accuracy Estimates

The GOES-R SUVI instrument performance requirements, which are levied upon the SUVI Level 1b product, state precision and accuracy only in terms of *effective area.* This does not take into account the contributions to uncertainty from the scattering inherent to the SUVI instrument. Given that the Level 1b product (from which all other SUVI products are generated) will be unable to state the measurement uncertainty in terms of physical quantities, it is very difficult to estimate the uncertainty in the Level 2+ products. Thus, any methodology for determining the accuracy or precision of the SUVI Level 2+ products will not be straightforward. To judge the performance of the algorithm in a meaningful manner, a more abstract approach is called for.

One method would be to develop a scheme to "score" the SUVI Coronal Hole Boundaries algorithm based on the number of pixels and/or amount of area correctly and incorrectly included in the algorithm calculations. This would be achieved by comparing output from the Coronal Holes Boundaries algorithm to output from an expert evaluation. The scheme for generating a score might resemble:

$$Score = \frac{p_{correct} - p_{missed} - p_{incorrect}}{p_{total}}$$

A perfect score would be "1.0", meaning that the algorithm had not missed or included non-coronal hole pixels (very unlikely). Any incorrect inclusions or missed pixels detract from that perfect score, and so would return a score significantly less than 1.0.

4.2.2 Error Budget

TBD

An error budget will be determined. This might occur after the Phase 3 ATBDs have been accepted.

5 PRACTICAL CONSIDERATIONS

5.1 Numerical Computation Considerations

- The development version of the SUVI Coronal Hole Boundaries algorithm utilizes several specialized image processing tools available in open source libraries. These are typically already optimized for speed, so consideration should be given to taking advantage of these instead of "reinventing the wheel".
- After the initial classification of the thematic map into individual features, the algorithm can then take advantage of multi-CPU processing of each coronal hole feature.

5.2 Programming and Procedural Considerations

- Consideration should be given to the case where the SUVI.19 Thematic Map has not been updated. It is theoretically possible to "evolve" either an older thematic map product, or "updating" the prior Coronal Hole Boundaries product using the standard model of solar differential rotation.

- Conversion to heliographic coordinates requires an input from the Astronomical Almanac for the Solar P and β angles. It is an assumption that, somewhere in the product chain, all the inputs required for the conversion will be supplied to the Coronal Holes Boundaries algorithm within the Thematic Map metadata. Currently, these inputs are supplied in an external LUT by the developer. If, these quantities are not to be supplied by the Thematic Map, then provisions should be made for a forward model or a rolling LUT maintained to supply these quantities.

5.3 Quality Assessment and Diagnostics

The quality of the SUVI Coronal Hole Boundaries algorithm should be assessed and tracked over multiple instances using the scoring scheme, or some equivalent, described in Section 4.2.1. The multiple instances should include assessments over time and under differing solar conditions. Re-assessing the performance of the Coronal Hole Boundaries algorithm over the SUVI Coronal Hole Boundaries algorithm's lifetime has value: changing scores would indicate problems within the product chain.

5.4 Exception Handling

The following is a list of exceptions and required actions for the SUVI Coronal Hole Boundaries algorithm:

- No Thematic Map
    a. Error Report with exception details
    b. Exit without Null Report
    c. "Evolve" older Thematic Map
- Thematic Map with no data
    a. Error Report with exception details
    b. Exit without Null Report
    c. "Evolve" older Thematic Map
- Thematic Map with mis-matching codes/values
    a. Exit with Error Report
    b. "Evolve" older Thematic Map

- Thematic Map has zero coronal hole features
    a. Exit with Null Report
- Vertex coordinates off solar disk
    a. Resolve to solar limb
- Heliographic Conversion ancillary data not available
  TBD

5.5 Algorithm Validation

The SUVI Coronal Hole Boundaries algorithm will be validated by ascertaining its operational value to the SWPC forecast office, both as an independent data product, and as input to other more derived data products. Ideally this will be done as part of a GOES Program Proving Ground or similar demonstration project.

6 ASSUMPTIONS AND LIMITATIONS

6.1 Performance

The SUVI Coronal Hole Boundaries algorithm is a collection of image processing routines/procedures that are not based on physical principles. This is by design since the SUVI.19 product is the sole input and all relation to physical units has been purged. As such, the accuracy of the SUVI Coronal Hole Boundaries product can only be as good as the quality of the SUVI.19 product.

6.2 Assumed Sensor Performance

The SUVI instrument is expected to perform to its operational requirements described in the GOES-R Mission Requirements Document (MRD), and related documentation. As of the time of this writing, the SUVI vendor has informally agreed to provide metadata with each image that informs the user or algorithm if/when these level 1b performance requirements may not be getting met. This includes, for example, channel-specific noise floors and saturation thresholds in units comparable to the pixel values.

6.3 Possible Product Improvements

6.3.1 Improvement #1

TBD

Appendix A: One Method for Clustering/Labeling

For any Thematic Map, it is simple to focus on one type of thematic map feature by setting all values not equivalent to the feature value to one value. In this instance where the thematic map values are integers, and the features that need to be analyzed in the SUVI Phase 3 products are non-zero, then it is convenient to set all values from features not of interest to zero. I.e.:

$$A_{TM}[A_{TM} \neq < feature\ value >] = 0$$

It is also advantageous to set the pixels that comprise the features of interest to "1":

$$A_{TM}[A_{TM} = < feature\ value >] = 1$$

One way to identify what we call "clusters" is to iterate either through the rows or columns of the Thematic Map's 2-D array. For this example, we will use rows. For each row, we take the entire row as a slice across the array (Figure 9):

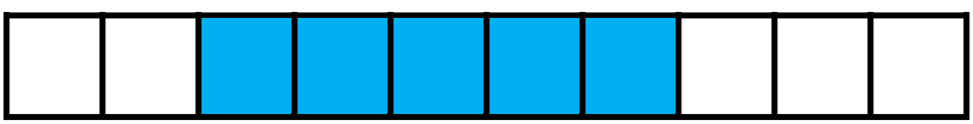

Figure 10: Representation of an array slice. The blue cells represent where the array values equal one.

We then perform an element-by-element difference across the array slice (Figure 10):

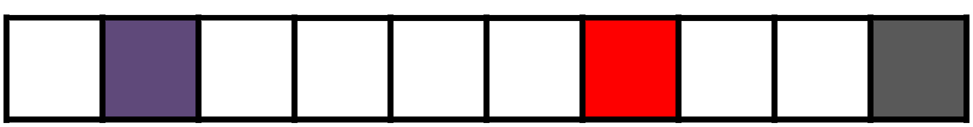

Figure 11: A representation of the differenced array slice.

Notice the different colored blocks: the gray denotes where we need to pad the array with a fill value (the difference operation leaves the result one element shorter), the purple denotes a negative value, and the red denotes a positive value. These results allow us to scan across the slice to determine the portions that belong to the feature's clusters: negative values are the start of a cluster, the positive values indicate the end (in that slice). It is important to record the index values so that it is possible to compare across adjacent slices (Figure 11):

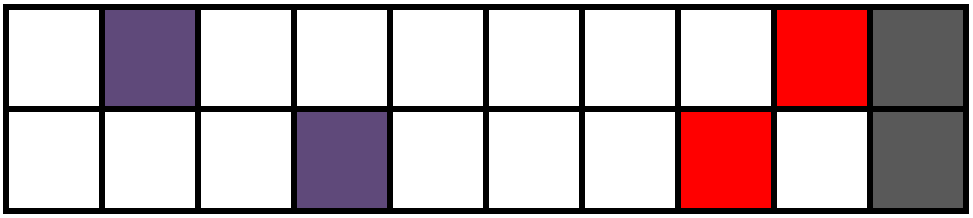

Figure 12: A representation of two difference slices with a demonstrated intersection of feature fragments.

By comparing the indexes across adjacent slices, one can determine which cluster fragments belong to the same cluster. If there is an intersection in the indexes, then the fragments belong in the same cluster. In the case where a positive value precedes a negative value in the difference slice, then it can be assumed that the cluster fragment starts at the beginning of the slice. Similarly, if the last non-zero value in the difference slice is a negative value, then the cluster fragment ends at the end of the slice.

The SUVI detector rows contain 1280 pixels, so it is completely feasible that any individual row will contain fragments from several different feature clusters. Each successive row is compared to the previous row, and where there is intersection of the indices of the cluster fragments then the fragments are added to the clusters which they share overlap. It is also possible that a cluster fragment will have overlap with more than one cluster. In this case the clusters that overlap will be merged into one cluster and will be treated thereafter as a single cluster.

Appendix B: Next Nearest-Neighbor Pixel Ordering

The development version of the SUVI Coronal Hole Boundaries algorithm determines the boundary pixels for each coronal hole by first performing a binary dilation and then subtracting the original coronal hole from that (see Figure 7). When the locations of the remaining boundary pixels are read out, the order is one of row:column. An example:

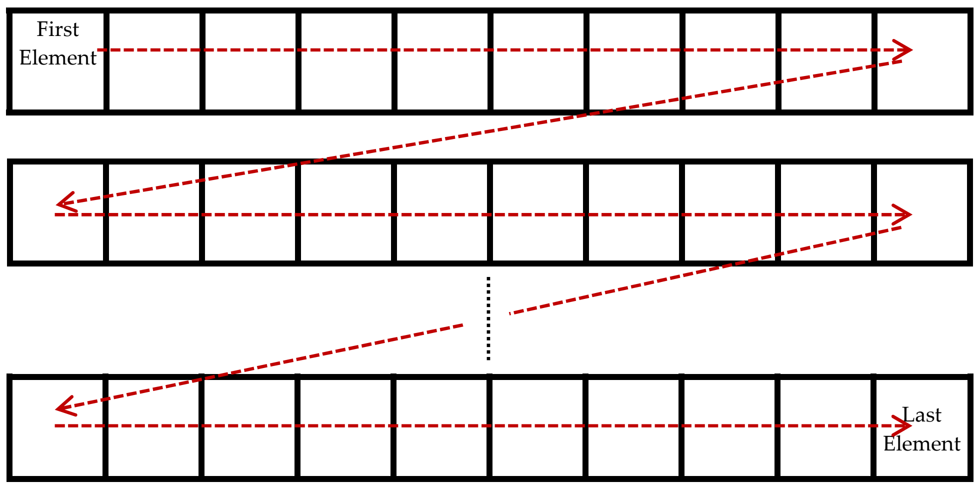

Figure 13: The read-out order for row:column**.**

What is needed is to reorder the list provided from the readout operation represented above in Figure 12 into one that describes a path around the coronal hole. This is done by picking a starting point. For the development version of the algorithm, the starting point is selected by taking the first minimum y-value at the median x-value. The next point is selected by determining the next nearest neighbor biased towards the positive x-direction:

$$P_{Start} = \mathrm{P}[\mathrm{median}(P_x), \min(P_y)]\,[0]$$

P_{Start} is removed from the pixel list (P), and then the next pixel is selected:

$$P_{Next} = P\left[min\left(\sqrt[2]{\left((P_x^2 - P_{Start,x}^2)^2 + (P_y^2 - P_{Start,y}^2)^2\right)}\right)\right]$$

Running the ordering procedure as described exactly above can lead to bad results; there is a lack of discrimination for how well the path is evolving. For this reason the development version tests the minimum distance with different limits. If the minimum distance exceeds some arbitrary value, the algorithm will backtrack along its last few points, re-testing for an alternate path. If an alternate is not found, then the algorithm assumes it is finished and returns.

Appendix C: Vertex Reduction

To reduce the total number of vertices from the potential thousands of pixels to the desired 16 vertices for the SUVI Coronal Hole Boundaries product requires some method for determining which pixels are most necessary to accurately describe the coronal hole. There might be canned routines to perform this task, and the implementer should research if such are available. Presented here is an ad-hoc method.

A good first start is to include the points that describe the extent of the feature:

$$V = \left[(x_{x\_min}, y_{x\_min}), \left(x_{x_{max}}, y_{x\_max}\right), \left(x_{y\_min}, y_{y\_min}\right), (x_{y\_max}, y_{y\_max})\right]$$

For the remaining vertices, a starting point needs to be selected from the set of feature points P:

$$x_{median} = median(P_x)$$
$$y_{median} = median(P_y[x_{median}])$$

This point (x_{median}, y_{median}) should lie roughly in the middle of the feature. It is worth checking that this point does belong to the list of feature points. If so, then change the order of calculation above:

$$y_{median} = median(P_y)$$
$$x_{median} = median(P_x[y_{median}])$$

The next step is to calculate the distance of the boundary pixels (B) from the mid-point:

$$R = \sqrt[2]{(B_x - x_{median})^2 + \left(B_y - y_{median}\right)^2}$$

The extrema of this radius-valued path should provide some measure of the criticality of each point to describing the region of the feature.

$$R_{crit} = [[True, R[1:] < R[:-1]]\&\& [R[:-1] < R[1:], True]] \bigcup [[True, R[1:] > R[:-1]]\&\& [[R:-1] > R[1:], True]]$$

This determines local extrema – those points whose immediate neighbors are greater or less than that point. If this produces too many vertices (more than 12 (16 minus the four extents)), then expand the extrema evaluation out to the next points and take the intersection, i.e.:

$$R_{crit} = R_{crit}(1) \bigcap R_{crit}(2)$$

Appendix D: Delaunay Triangulation

Delaunay Triangulation is a method for taking a finite set of points P and generating sets of triangles DT(P). Conditions for the triangulation state that no points in P can exist in the circumcircle of any triangle in DT. Legal triangulations maximize the size of the minimum angle for all triangles. Triangulation starts by taking three adjacent points as the basis for one triangle, and iterating through the remaining points testing whether points fall within the circumcircle:

$$\begin{vmatrix} A_x - D_x & A_y - D_y & (A_x^2 - D_x^2) + (A_y^2 - D_y^2) \\ B_x - D_x & B_y - D_y & (B_x^2 - D_x^2) + (B_y^2 - D_y^2) \\ C_x - D_x & C_y - D_y & (C_x^2 - D_x^2) + (C_y^2 - D_y^2) \end{vmatrix} > 0$$

A, B, and C are treated as the three adjacent points and D as the iterable point. Once a point is found that satisfies the circumcircle condition, then we now have two adjoining triangles to work with: $\overline{ABC}$ and $\overline{ADC}$. These two triangles, however, have already violated the condition for legal Delaunay triangulation; point D is inside the circumcircle of triangle $\overline{ABC}$. The arrangement of the triangles needs to be adjusted (called "flipping") to satisfy that condition (Figure 14).

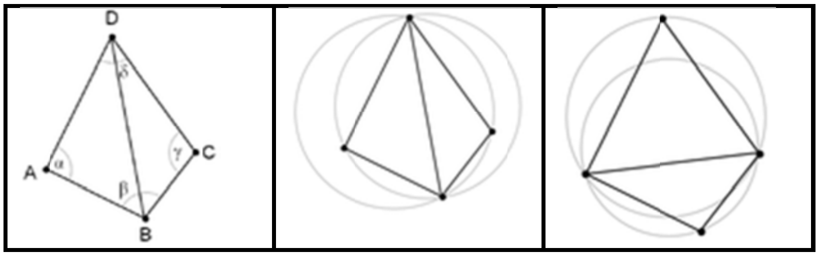

Figure 14: An example of Delaunay Triangulation. Four points (*left*) A, B, C, and D form two triangles; ABD and BDC. However, point A is clearly in the circumcircle of BDC (*middle*) and so the triangulation (*right*) needs to be flipped in order to satisfy the triangulation conditions.

The points used to generate the first set of triangles then serve as the seed for further triangulation. Points adjacent to the points of the triangles are then added and tested against the circumcircle condition. For example, a point "E" in P that is adjacent to A that is not B is considered for adding to the triangulation. Triangles ADE and ABE will be tested against the circumcircle condition. Triangles that pass the test are added to the set, and the triangulation continues until no further triangles pass the circumcircle condition.

Appendix E: Conversion to Heliographic Coordinates

To convert points in pixel coordinates to heliographic coordinates (Stonyhurst and Carrington), several pieces of information are needed. These are listed in the table below:

Name	Description
x	Column number of pixel
y	Row number of pixel
x_{center}	Column number of solar center
y_{center}	Row number of solar center
Δ_x	Instrument platescale – x direction
Δ_y	Instrument platescale – y direction
r_{sun}	Solar radius in pixels
P	Solar P angle at time of observation (radians)
β	Solar β angle at time of observation (radians)

$D_\odot$	Solar Diameter (meters)
S_{rot}	Rotation of satellite relative to ecliptic
I_{rot}	Rotation of instrument relative to satellite axis

First, we convert the radius of the Sun to arcseconds:

$$R_{Sun} = r_{sun} * \Delta_{(x,y)}$$

Second, the 2-D pixel array coordinates need to be converted to 3-D angular coordinates:

$$x_\odot = R_{Sun} * \cos\left(sin^{-1}\left(\frac{(x - x_{center})}{R_{Sun}}\right)\right)$$
$$y_\odot = R_{Sun} * \sin\left(sin^{-1}\left(\frac{x - x_{center})}{R_{Sun}}\right)\right)$$
$$z_\odot = R_{Sun} * \sin\left(sin^{-1}\left(\frac{(y - y_{center})}{R_{Sun}}\right)\right)$$

Next, the rotation of the image relative to solar north needs to be accounted for:

$$x_{derotated} = x_\odot * cos(\beta) + z_\odot * sin(\beta)$$
$$y_{derotated} = -1 * x_\odot * sin(P) * sin(\beta) + y_\odot * cos(\beta) + z_\odot * sin(P) * cos(P)$$
$$z_{derotated} = -1 * x_\odot * cos(P) * sin(\beta) - y_\odot * sin(P) + z_\odot * cos(P) * cos(\beta)$$

Finally, the conversion to Stonyhurst heliographic latitude/longitude can be performed:

$$\varphi_{Stonyhurst} = \left(\frac{360°}{2\pi}\right) * tan^{-1}\left(\frac{z_{derotated}}{\sqrt[2]{(x_{derotated}^2 + y_{derotated}^2)}}\right)$$
$$\lambda_{Stonyhurst} = \left(\frac{360°}{2\pi}\right) * tan^{-1}\left(\frac{y_{derotated}}{x_{derotated}}\right)$$

Heliographic latitude is equivalent between Stonyhurst and Carrington coordinates:

$$\varphi_{Carrington} = \varphi_{Stonyhurst}$$

For the Carrington longitude, the Stonyhurst longitude is added to an offset to account for the Carrington rotation:

$$\lambda_{Carrington} = modulus((\lambda_{Stonyhurst} + L_0), 360)$$
$$L_0 = 360 * \left(1 - \frac{t_{rot}\ (days)}{27.2753\ days}\right)$$

REFERENCES

Lemon et al. (2012), The Atmospheric Imaging Assembly (AIA) on the Solar Dynamics Observatory (SDO), *Solar Physics*

Rigler, E.J., S. Hill (2011), *GOES R SUVI Thematic Maps Algorithm Theoretical Basis Document, Version 1.0.*

Rigler, E.J., S. Hill (2011), *GOES R SUVI Thematic Maps Implementation and User's Guide, Version 1.0*.

Rigler, E.J., S. Hill (2011), *GOES R SUVI Thematic Maps Algorithm Test Plan and Results, Version 1.0*.

Thompson, W. T. (2006), Coordinate systems for solar image data, *Astronomy & Astrophysics*