Effective Date: June 3, 2011
Revision -

GSFC JPSS CMO
January 3, 2012
Released

Joint Polar Satellite System (JPSS) Ground Project
Code 474
474-00097

Joint Polar Satellite System (JPSS) Operational Algorithm Description Document for Common Adjacency

For Public Release

The information provided herein does not contain technical data as defined in the International Traffic in Arms Regulations (ITAR) 22 CFC 120.10. This document has been approved For Public Release.

Goddard Space Flight Center
Greenbelt, Maryland

National Aeronautics and Space Administration

This page intentionally left blank.

Joint Polar Satellite System (JPSS) Operational Algorithm Description Document for Common Adjacency

JPSS Electronic Signature Page

Prepared By:

Neal Baker
JPSS Data Products and Algorithms, Senior Engineering Advisor
(Electronic Approvals available online at (https://jpssmis.gsfc.nasa.gov/mainmenu_dsp.cfm)

Approved By:

Heather Kilcoyne
DPA Manager
(Electronic Approvals available online at (https://jpssmis.gsfc.nasa.gov/mainmenu_dsp.cfm)

Goddard Space Flight Center
Greenbelt, Maryland

Check the JPSS MIS Server at https://jpssmis.gsfc.nasa.gov/frontmenu_dsp.cfm to verify that this is the correct version prior to use.

This page intentionally left blank.

Check the JPSS MIS Server at https://jpssmis.gsfc.nasa.gov/frontmenu_dsp.cfm to verify that this is the correct version prior to use.

Preface

This document is under JPSS Ground AERB configuration control. Once this document is approved, JPSS approved changes are handled in accordance with Class I and Class II change control requirements as described in the JPSS Configuration Management Procedures, and changes to this document shall be made by complete revision.

Any questions should be addressed to:

JPSS Ground Project Configuration Management Office
NASA/GSFC
Code 474
Greenbelt, MD 20771

Check the JPSS MIS Server at https://jpssmis.gsfc.nasa.gov/frontmenu_dsp.cfm to verify that this is the correct version prior to use.

This page intentionally left blank.

Check the JPSS MIS Server at https://jpssmis.gsfc.nasa.gov/frontmenu_dsp.cfm to verify that this is the correct version prior to use.

Change History Log

Revision	Effective Date	Description of Changes (Reference the CCR & CCB/ERB Approve Date)
Original	06/03/2011	This version incorporates **474-CCR-11-0112** which converts D48316, NPOESS Operational Algorithm Description Document for Common Adjacency, Rev -- dated 02/03/2010 to a JPSS document, Rev -. This was approved by the JPSS Ground Algorithm ERB on June 3, 2011.

Check the JPSS MIS Server at https://jpssmis.gsfc.nasa.gov/frontmenu_dsp.cfm to verify that this is the correct version prior to use.

This page intentionally left blank.

Check the JPSS MIS Server at https://jpssmis.gsfc.nasa.gov/frontmenu_dsp.cfm to verify that this is the correct version prior to use.

NATIONAL POLAR-ORBITING OPERATIONAL ENVIRONMENTAL SATELLITE SYSTEM (NPOESS)

OPERATIONAL ALGORITHM DESCRIPTION DOCUMENT FOR COMMON ADJACENCY (D48316 Rev ---)

CDRL No. A032

Northrop Grumman Space & Mission Systems Corporation
One Space Park
Redondo Beach, California 90278

Copyright © 2004-2010
Northrop Grumman Corporation and Raytheon Company
Unpublished Work
ALL RIGHTS RESERVED

Portions of this work are the copyrighted work of Northrop Grumman and Raytheon. However, other entities may own copyrights in this work.

This documentation/technical data was developed pursuant to Contract Number F04701-02-C-0502 with the US Government. The US Government's rights in and to this copyrighted data are as specified in DFAR 252.227-7013, which was made part of the above contract.

This document has been identified per the NPOESS Common Data Format Control Book – External Volume 5 Metadata, D34862-05, Appendix B as a document to be provided to the NOAA Comprehensive Large Array-data Stewardship System (CLASS) via the delivery of NPOESS Document Release Packages to CLASS.

The information provided herein does not contain technical data as defined in the International Traffic in Arms Regulations (ITAR) 22 CFR 120.10.

This document has been approved by the Unites States Government for public release in accordance with NOAA NPOESS Integrated Program Office.

Distribution: Statement A: Approved for public release; distribution is unlimited.

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

NATIONAL POLAR-ORBITING OPERATIONAL ENVIRONMENTAL SATELLITE SYSTEM (NPOESS)

OPERATIONAL ALGORITHM DESCRIPTION DOCUMENT FOR COMMON ADJACENCY (D48316 Rev ---)

ELECTRONIC APPROVAL SIGNATURES:

_____

Roy Tsugawa Date
Algorithm & Data Processing IPT Lead &
Algorithm Change Control Board Chairperson

_____

Ben James Date
Operations & Support IPT Lead

The following individual is recognized for his contributions to the current or previous versions of this document.

Ward Dare

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

Revision/Change Record

Document Number **D48316**

Revision	Document Date	Revision/Change Description	Pages Affected
---	02-03-10	Initial Release. ECR A-249A. Approved for Public Release per Contracts Letter 100610-02.	ALL

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

Table of Contents

1.0 INTRODUCTION 1
1.1 Objective 1
1.2 Scope 1
1.3 References 1
1.3.1 Document References 1
1.3.2 Source Code References 2
2.0 ALGORITHM OVERVIEW 3
2.1 Common Adjacency Description 3
2.1.1.1 Inputs 3
2.1.1.1.1 Imagery Resolution Inputs 4
2.1.1.1.2 Moderate Resolution Inputs 4
2.1.1.1.3 Configuration Guide 4
2.1.1.2 Outputs 4
2.1.2 Algorithm Processing 5
2.1.3 Implementation 7
2.1.3.1 Main Module - ProCmnAdjFactory 9
2.1.3.2 ProCmnAdjTable 9
2.1.3.2.1 initTable 9
2.1.3.2.2 getDistance 13
2.1.3.2.3 setCurrentPixel 14
2.1.3.2.4 getNextAdjPixel 14
2.1.3.2.5 resetAdjIndex 14
2.1.3.3 ProCmnAdjIMGTable 14
2.1.3.4 ProCmnAdjMODTable 15
2.1.3.5 ProCmnAdjPixel 15
2.1.3.6 ProCmnAdj 15
2.1.3.7 ProCmnAdjInf 15
2.1.3.7.1 ProCmnAdjInf_resetAdjIndex () 15
2.1.3.7.2 ProCmnAdjInf_setCurrentPixel () 15
2.1.3.7.3 ProCmnAdjInf_getNextAdjPixel () 15
2.1.3.8 Interfaces 15
2.1.3.8.1 Interfacing with Common Adjacency 15

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

2.1.3.8.2 Get the Common Adjacency table 16
2.1.3.8.3 Using a Common Adjacency table 17
2.1.3.8.4 Find adjacent pixel and perform algorithm specific processing 17
2.1.4 Graceful Degradation 18
2.1.4.1 Graceful Degradation Input 18
2.1.4.2 Graceful Degradation Processing 18
2.1.4.3 Graceful Degradation Output 18
2.1.5 Exception Handling 18
2.1.6 Data Quality Monitoring 18
2.1.7 Computational Precision Requirements 18
2.1.8 Algorithm Support Considerations 18
2.1.9 Assumptions and Limitations 18
2.1.10 Science Support References 18
3.0 GLOSSARY/ACRONYM LIST 19
3.1 Glossary 19
3.2 Acronyms 22
4.0 OPEN ISSUES 23

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

List of Figures

Figure 1: Common Adjacency Algorithm Usage Flow Diagram 6
Figure 2: Class Diagram for CMN Adjacency 8
Figure 3: Determining Nearest Row Index in Previous or Next Scan 10
Figure 4: Determining Column Offset in Previous or Next Scan 11
Figure 5: Current Pixel Relationship to Scan and Adjacency Grid 12
Figure 6: Common Adjacency Sequence Diagram 16

List of Tables

Table 1: Reference Documents 2
Table 2: Source Code References 2
Table 3: Imagery Band Geolocation Data 4
Table 4: Imagery Band Grid-Row-Column Data 4
Table 5: Moderate Band Geolocation Data 4
Table 6: Moderate Band Grid-Row-Column Data 4
Table 7: Common Adjacency Output Pixel 5
Table 8: Glossary 19
Table 9: Acronyms 22
Table 10: List of OAD TBD/TBR 23

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

1.0 INTRODUCTION

1.1 Objective

The purpose of the Operational Algorithm Description (OAD) document is to express, in computer-science terms, the remote sensing algorithms that produce the National Polar-Orbiting Operational Environmental Satellite System (NPOESS) end-user data products. These products are individually known as Raw Data Records (RDRs), Temperature Data Records (TDRs), Sensor Data Records (SDRs) and Environmental Data Records (EDRs). In addition, any Intermediate Products (IPs) produced in the process are also described in the OAD.

The science basis of an algorithm is described in a corresponding Algorithm Theoretical Basis Document (ATBD). The OAD provides a software description of that science as implemented in the operational ground system -- the Data Processing Element (DPE).

The purpose of an OAD is two-fold:

1. Provide initial implementation design guidance to the operational software developer.
2. Capture the “as-built” operational implementation of the algorithm reflecting any changes needed to meet operational performance/design requirements.

An individual OAD document describes one or more algorithms used in the production of one or more data products. There is a general, but not strict, one-to-one correspondence between OAD and ATBD documents.

1.2 Scope

The scope of this document is limited to the description of the Common (CMN) Adjacency algorithm which is used to fill in a neighborhood of VIIRS pixels for later processing (e.g. determine if adjacent pixels to the pixel being processed are cloudy). Since several VIIRS algorithms use a neighborhood of pixels in their processing, it was decided to develop a common module for those algorithms that require cross-scan/granule processing. A key part of the algorithm’s functionality would be to replace bow-tie deleted pixels in the current scan with similar observations (i.e. pixels viewing essentially the same part of the earth) from adjacent scans. A VIIRS scan contains multiple detectors and as the size of the fields of view of these detectors increases as the sensor moves off-nadir, the scan’s footprint resembles a bow-tie with adjacent scans overlapping—i.e. the same geography being viewed in consecutive scans. In fact, the same geography may appear in up to three consecutive scans at the off-nadir edges of the scan. To reduce the data flow from the spacecraft, approximately 13% of the VIIRS pixels are trimmed on-board (i.e. duplicate earth views are removed from the data flow) (on-board pixel trim). To further reduce the processing of ‘duplicate’ observations, an additional 7% of the pixels are ‘trimmed’ during product processing (extended pixel trim). CMN Adjacency is a modification of the method originally employed by the dropped Active Fires science algorithm. The underlying theory is captured in this document rather than in an Algorithm Theoretical Basis Document (ATBD*).*

1.3 References

1.3.1 Document References

The documents relevant to the algorithm described in this OAD are listed in Table 1.

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

Table 1: Reference Documents

Document Title	Document Number/Revision	Revision Date
Operational Algorithm Description Document for VIIRS Active Fires	D36981 Rev B2	01 Dec 2009
Tech Memo Cross-granule Algorithm Processing	NP-EMD.2005.510.0038	7 Mar 2005
D35836_G_NPOESS_Glossary	D35836 Rev. G	10 Sep 2008
D35838_G_NPOESS_Acronyms	D35838 Rev. G	10 Sep 2008
Operational Algorithm Description Document for VIIRS SDR GEO	D41868_Rev A20	04 Nov 2009

1.3.2 Source Code References

The Common Adjacency algorithm has yet to be assigned a version number as algorithm versioning will begin after transition to Operations and Sustainment (O&S). The active fires algorithm drop that was the starting point for the Common Adjacency algorithm is listed in Table 2.

Table 2: Source Code References

Reference Title	Reference Tag/Revision	Revision Date
Active Fires Science Code (VIIRS-AER-AF-1.02) (ECR-A007A)	ISTN_VIIRS_NGST_2.3	30 Sep 2003
IDPS Operational Software	Build 1.5 ; 'Beyond Final' Spiral (OAD Rev. ---, 10 Jun 09)	19 Jul 2007
ACCB (No Code updates)	OAD Rev ---	03 Feb 2010

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

2.0 ALGORITHM OVERVIEW

Some algorithms use neighboring observations in the creation of their products. CMN Adjacency is used to identify neighboring observations and this includes identifying replacements for bow-tie removed observations from adjacent scans. The size of the neighborhood of observations used in subsequent processing varies from the eight adjacent pixels to multiple scans with all bow-tie removed pixels being replaced.

The bow-tie replacement portion of the CMN Adjacency algorithm was developed based on the original pixel replacement approach used by the Active Fires algorithm. This approach used a look-up table to index into the adjacent scan to retrieve data based on location within the current scan, however, this approach did not account for variation in the VIIRS telescope rotation speed or the rotation of the earth. The CMN Adjacency algorithm accounts for these variations by using distance calculations between pixels to determine the nearest neighbors.

The Common Adjacency algorithm is used by the following data processing algorithms:

1. VIIRS Active Fires
2. VIIRS Cloud Mask
3. VIIRS SDR Bright Pixel

2.1 Common Adjacency Description

The Common Adjacency algorithm is used to determine the row/column values of neighboring pixels given the primary pixel of interest and the size of the pixel neighborhood as specified by the radius supplied by the calling algorithm. A radius of one means to return the eight immediate surrounding pixels and a radius of two expands the neighborhood to 25 pixels (5x5). The formula for determining the height/width of the pixel neighborhood based on its radius is as follows:

$$\text{Pixel Neighborhood Width or Height} = (2 \cdot radius) + 1 \qquad \text{Eqn. 1.}$$

Based on the radius supplied by the calling algorithm, Common Adjacency determines and stores to memory the row and column values of each adjacent pixel in the neighborhood. Note that these adjacent pixels may be located in the current, previous, or next scan. Each adjacent pixel row/column pair is then retrieved from the Common Adjacency algorithm one at a time by the calling algorithm. A more detailed description of the algorithm is found in Section 2.1.2.

2.1.1.1 Inputs

The inputs for the Common Adjacency Algorithm are the geolocation information for the current granule and the needed adjacent scans from adjacent granules, and the radius of the adjacent pixel neighborhood. Note that three full geolocation granules are required for CMN adjacency processing: the current granule, the previous granule and the next granule. The previous and the next granules are the two along track neighboring granules, referred to as cross granules. The type of geolocation information is dependent on whether the distance between pixels is determined using the grid-row-column product (see VIIRS SDR OAD, D41868, GEO section) or the latitude and longitude from the VIIRS geolocation product. Note that the grid-row-column product is an intermediate product that is produced by the VIIRS geolocation algorithm to improve latency. For each granule, the geolocation algorithm produces a polar stereographic map grid that contains floating point row-column coordinates for all the pixels in the granule. The latitude-longitude of all of the samples in the granule can be determined from the grid-row-

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

column values. The operational default is to use the grid-row-column product for computing the distances since it is computationally faster (see Section 2.1.3.2.2 for details). The lat/lon data from the VIIRS geolocation product is only used when the grid-row column product is not available. The latter occurs during science to operational code conversion as the dropped science test data does not include a grid-row-column product. Table 3 and Table 4 list the inputs for image resolution pixel neighborhoods and Table 5 and Table 6 list the inputs for moderate resolution pixel neighborhoods.

2.1.1.1.1 Imagery Resolution Inputs

Table 3: Imagery Band Geolocation Data

Input	Type	Description	Units: Valid Range
Radius	Integer	Adjacency Grid Size Determinant	Unitless:1 to 15
lat	Float32[1536][6400]	Latitude	Radians : -Pi/2 to Pi/2
lon	Float32[1536][6400]	Longitude	Radians: 0 to 2Pi

Table 4: Imagery Band Grid-Row-Column Data

Input	Type	Description	Units: Valid Range
Radius	Integer	Adjacency Grid Size Determinant	Unitless: 1 to 15
Grow	Float64[1536][6400]	Grid Row	Unitless: 0 to 65535
Gcol	Float64[1536][6400]	Grid Col	Unitless: 0 to 65535

2.1.1.1.2 Moderate Resolution Inputs

Table 5: Moderate Band Geolocation Data

Input	Type	Description	Units: Valid Range
Radius	Integer	Adjacency Grid Size Determinant	Unitless: 1 to 7
Lat	Float32[768][3200]	Latitude	Radians : -Pi/2 to Pi/2
Lon	Float32[768][3200]	Longitude	Radians: 0 to 2Pi

Table 6: Moderate Band Grid-Row-Column Data

Input	Type	Description	Units: Valid Range
Radius	Integer	Adjacency Grid Size Determinant	Unitless: 1 to 7
Grow	Float64[768][3200]	Grid Row	Unitless: 0 to 65535
Gcol	Float64[768][3200]	Grid Col	Unitless: 0 to 65535

2.1.1.1.3 Configuration Guide

The CMN Adjacency Algorithm reads the XML format PRO_CMN_ADJ_CFG.xml configuration file. This file is the source of the pixel trim information used within the algorithm.

2.1.1.2 Outputs

The CMN Adjacency Algorithm creates a granule size array (either MOD or IMG resolution) containing the row and column indices for all pixels, with excluded pixels such as bow-tie deleted observations replaced with row and column data from adjacent scans. This array is

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

stored in memory and is used by the CMN Adjacency algorithm for extracting the adjacent pixel row-column pairs for the neighborhood required by the calling algorithm. The row-column pair of each adjacent pixel, along with a FillPixel flag, is output to the calling routine one at a time. The FillPixel Flag indicates whether a valid row-column pair could be found (set to *True* if valid row-column pair does *not* exist). This flag is triggered when the adjacent pixel's column is beyond the granule's cross-track boundaries (i.e., target pixels at the beginning or end of a scan will have neighborhoods that extend beyond the granule boundary). In addition, CMN Adjacency will also set this flag when the adjacent pixel's row exceeds the cross-granule's along-track boundary. The outputs are shown in Table 7.

Table 7: Common Adjacency Output Pixel

Output	Type/Size	Description	Units: Valid Range
Row	Int32	Row for adjacent pixel	Unitless : 0 to 1535
Column	Int32	Column for adjacent pixel	Unitless : 0 to 6399
FillPixel	Bool	Set to true if either the adjacent pixel row extends beyond either cross granule along-track boundary or the adjacent column extends past the granule cross-track boundaries. If this flag is set to true, this means that a valid replacement row-column pair could *not* be found for the adjacent pixel. Note that when this flag is triggered, the row-column values are set to -1.	Boolean : true/false

2.1.2 Algorithm Processing

Figure 1 is a flow chart of the CMN Adjacency algorithm. Obtaining the row and column pair for the adjacent pixels is a four step process. First, the calling algorithm provides the grid-row-column data (or lat/lon data if no grid-row-column data is available) to the CMN Adjacency initialization code. Next, CMN Adjacency creates a granule size table of row and column indices, where the excluded pixels are replaced with row-column pairs from adjacent scans and granules if necessary. In order to do this, the algorithm reads the Pixel Trim Configuration file, PRO_CMN_ADJ_CFG.xml, to determine if a pixel is bow-tie deleted. If it is, then the CMN Adjacency algorithm determines a candidate replacement from the adjacent scan by finding the closest pixel in the same column as the bow-tie deleted pixel in the current scan. Once the nearest pixel in the same column is determined, CMN Adjacency determines if a pixel in an adjacent column is closer than the one in the same column. This column shift accounts for the rotation of the earth and variation in the VIIRS telescope rotation speed. The end result is an adjacency table of all scans from which the adjacent pixels can be easily retrieved. In the third step, the calling algorithm provides the target pixel's row and column indices and the radius of the pixel neighborhood to CMN Adjacency, which uses these inputs to create a vector of rows and columns for all adjacent observations. Lastly, the adjacent pixels row and column information is fed to the calling algorithm one adjacent pixel at a time. Processing is done on a scan basis with two loops—one to process through each pixel in the current scan (*Processing Another Target Pixel* decision in Figure 1) and the other to feed back the row and column information on each adjacent pixel to the calling algorithm. The process then repeats for the next target pixel until all pixels in the current scan and subsequently all the scans in a granule have been processed.

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

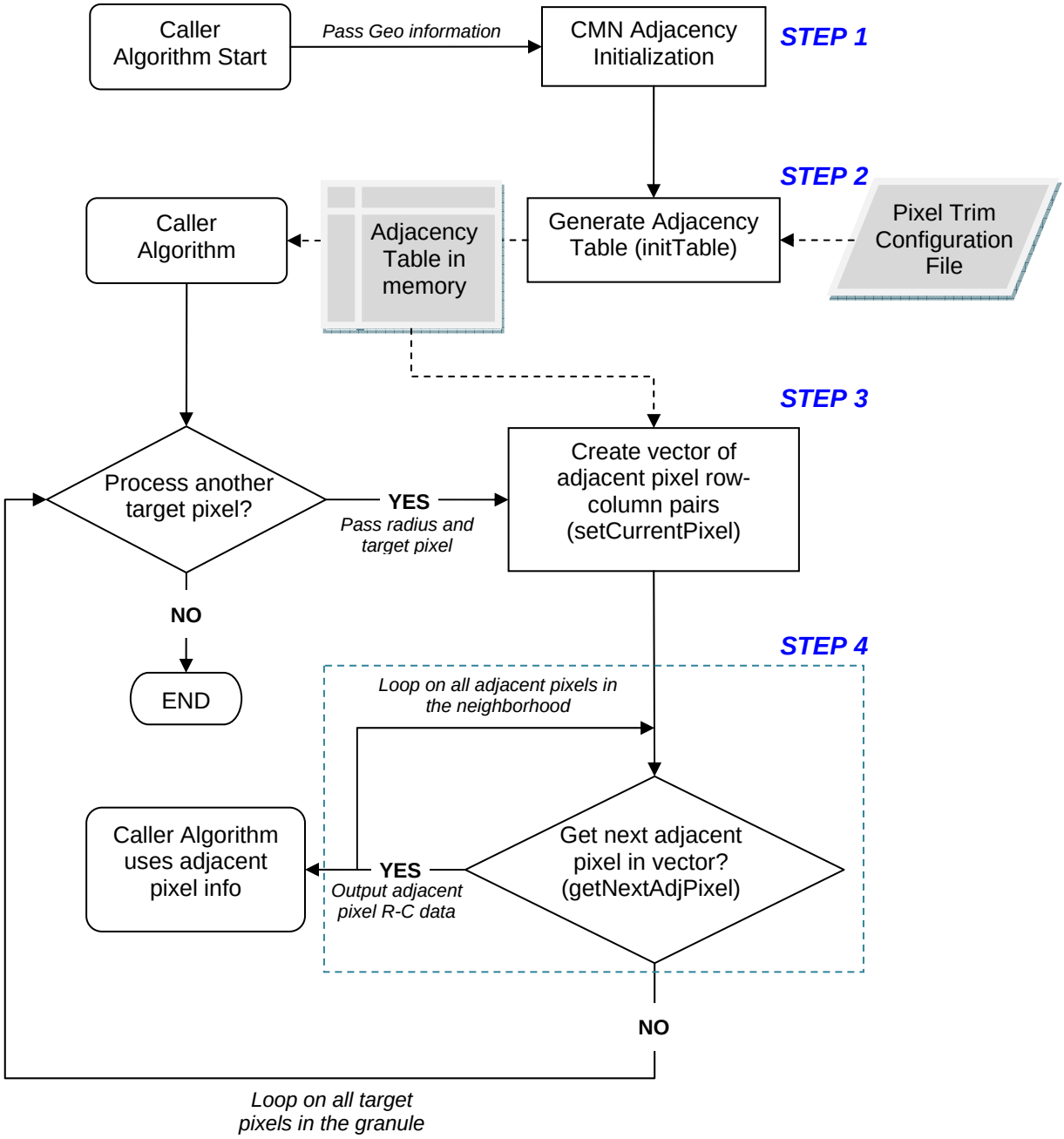

Figure 1: Common Adjacency Algorithm Usage Flow Diagram

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

2.1.3 Implementation

The CMN Adjacency logic is written in object-oriented C++. The class diagram is shown in Figure 2. The calling function initially gains access to the CMN Adjacency operations through the singleton class ProCmnAdjFactory by the static method getInstance. Once the calling algorithm has obtained an instance to the ProCmnAdjFactory object, ProCmnAdjFactory's getTable method is called to construct an actual ProCmnAdjTable derived object. The getTable method takes as parameters both a pointer to a data structure that provides access to the calling algorithm's geolocation data, as well as a flag that indicates whether a VIIRS imagery band dimensioned adjacency table, ProCmnAdjIMGTable, or a VIIRS moderate band dimensioned adjacency table, ProCmnAdjMODTable, should be created. The getTable method passes the pointer to the geolocation data to the appropriate adjacency table constructor. After constructing an adjacency table, getTable calls the adjacency table's initTable method and then returns the initialized adjacency table to the calling algorithm. Once the calling algorithm has access to an initialized ProCmnAdjTable object it may call that object's setCurrentPixel method, passing as arguments the row and column of the target (center) pixel as well as the adjacency radius. After calling setCurrentPixel, the calling algorithm may call getNextAdjPixel repeatedly, retrieving one ProCmnAdjPixel object at a time, until the list of pixels adjacent to the target pixel is exhausted.

Sections 2.1.3.1 – 2.1.3.7 describe methods, classes and interfaces within the singleton class. Section 2.1.3.8 is an example of how an algorithm may interface with Common Adjacency.

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

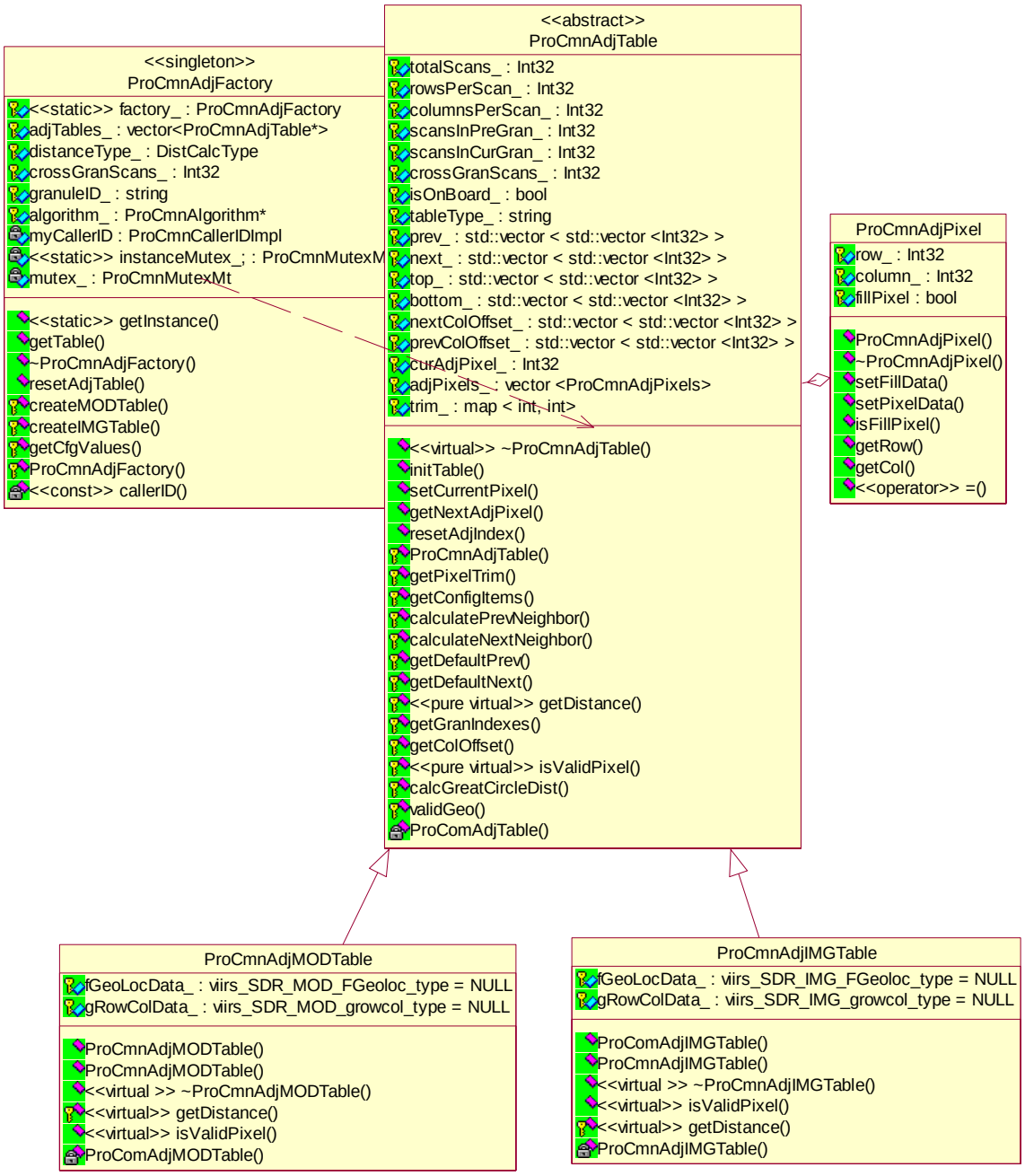

Figure 2: Class Diagram for CMN Adjacency

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

2.1.3.1 Main Module - ProCmnAdjFactory

Singleton class that returns either ProCmnAdjIMGTable or ProCmnAdjMODTable objects to the calling algorithm. There is one object per adjacent pixel. So, if the radius were one, eight tables with the row and column of the adjacent pixel would be returned.

2.1.3.2 ProCmnAdjTable

This is the base class from which ProCmnAdjIMGTable and ProCmnAdjMODTable inherit, allowing polymorphic behavior of overridden virtual methods.

2.1.3.2.1 initTable

The initTable method creates a granule size array of all scans with excluded pixels such as bow-tie deleted observations replaced with row and column data from adjacent scans. The initTable method determines the first row (minimum row index) and last row (maximum row index) where the current scan has valid data based on VIIRS pixel trim information. Rows for three consecutive scans (previous, current, and next) are numbered from zero. For example, at column number 639, the VIIRS moderate resolution extended pixel trim affects two rows at the bottom and top of the scan and therefore, the minimum row index is 18 and the maximum row index is 29 (see Figure 3). Using the minimum and maximum row information for each column, the method determines whether a pixel from either the previous or next scan needs to be substituted for any adjacent pixels within the current scan. For example, in Figure 5, out of the eight adjacent pixels to the current/target pixel, one resides in the pixel trim area and therefore, the CMN Adjacency method is used to provide a pixel viewing essentially the same geography from the adjacent scan.

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

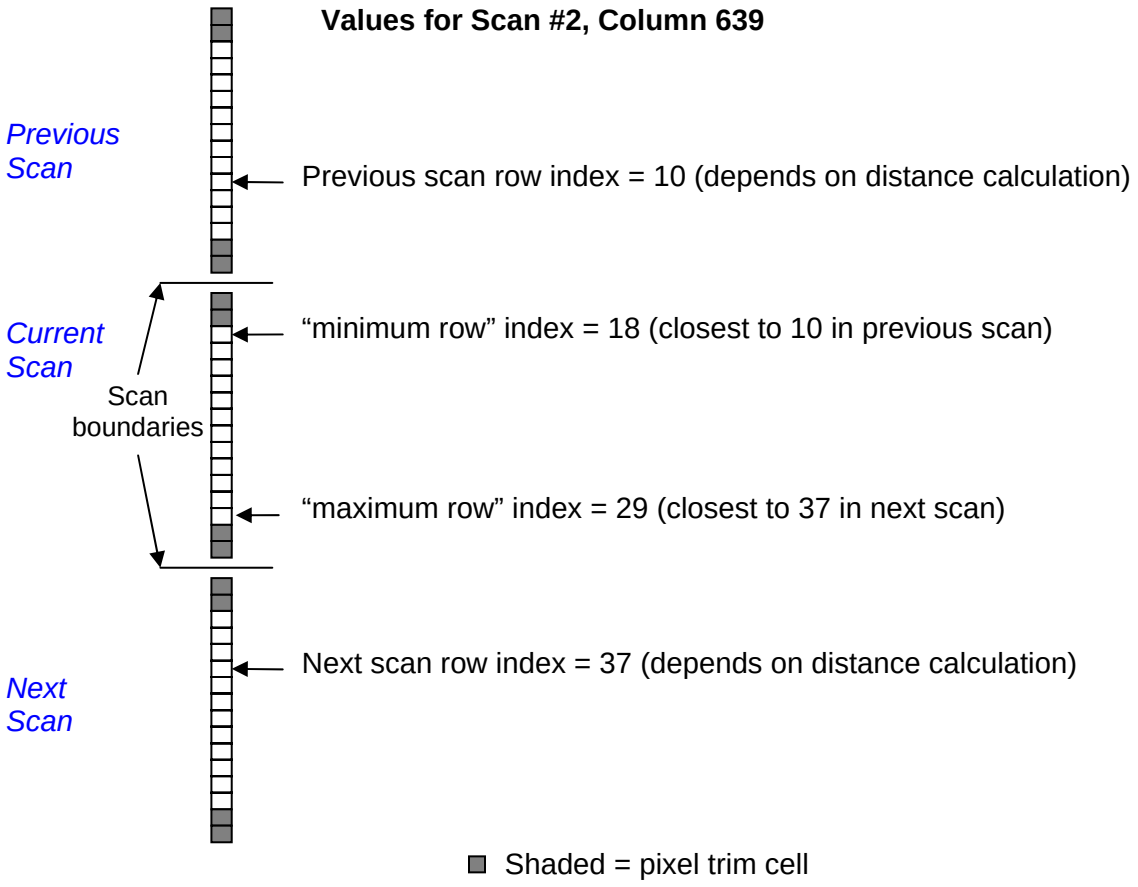

Figure 3: Determining Nearest Row Index in Previous or Next Scan

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

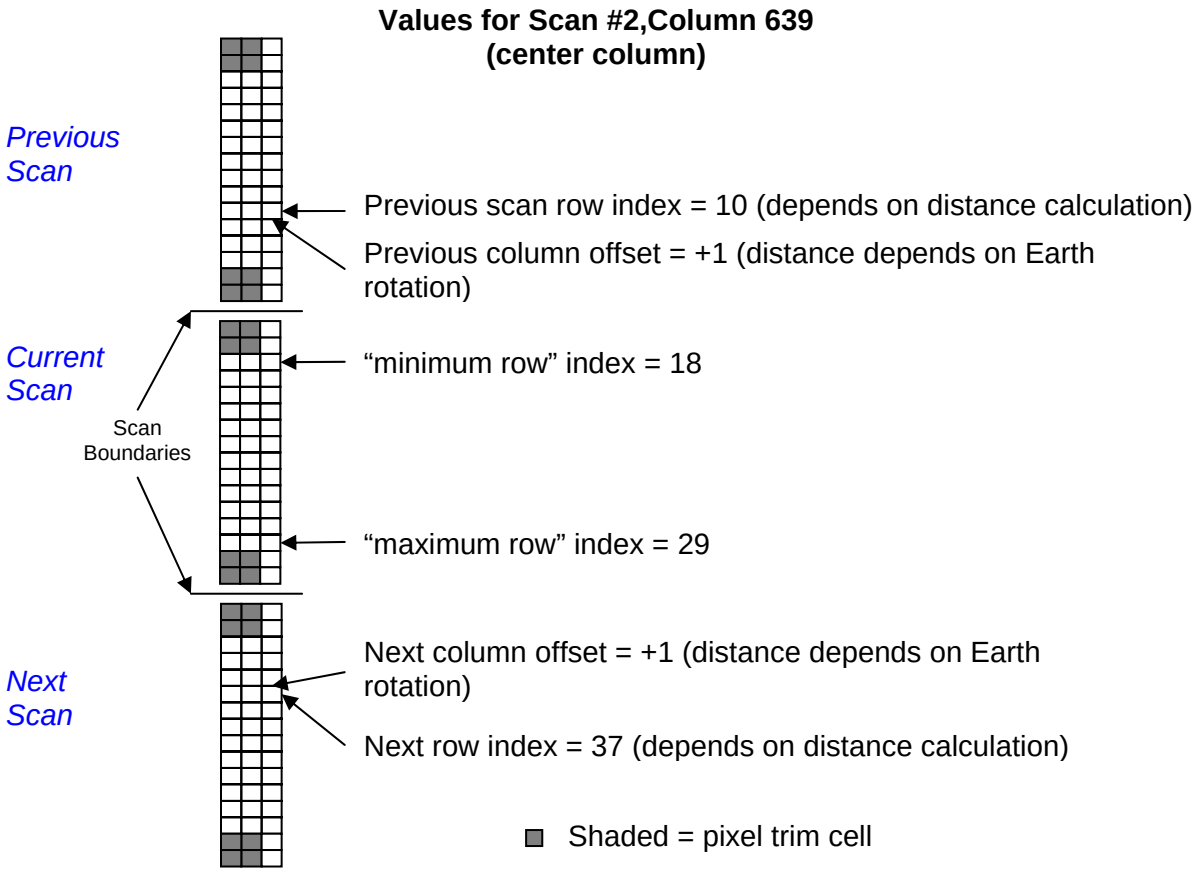

Figure 4: Determining Column Offset in Previous or Next Scan

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

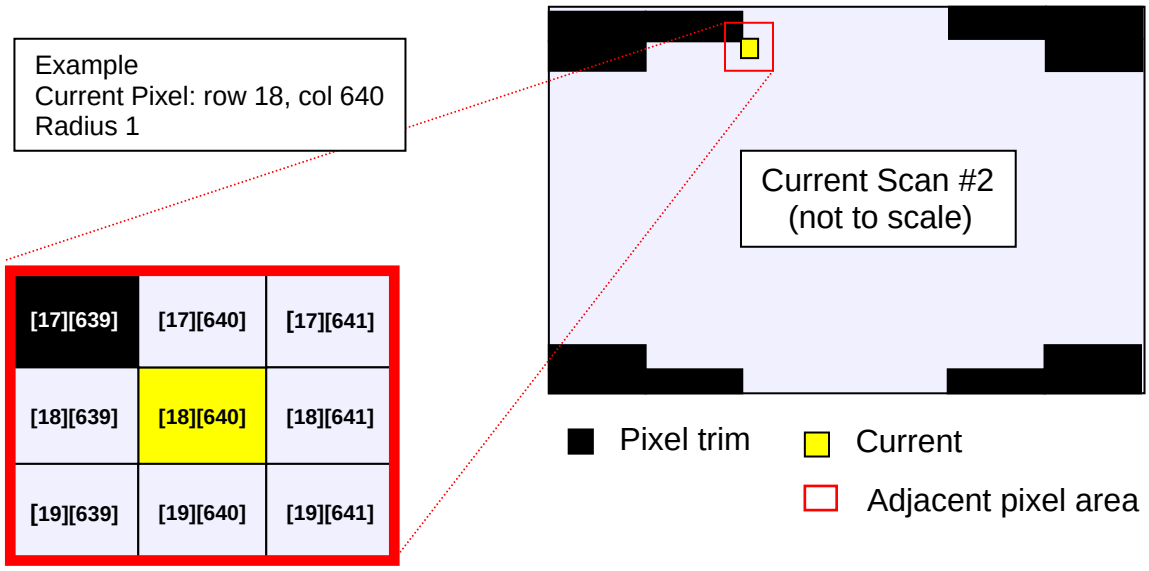

Figure 5: Current Pixel Relationship to Scan and Adjacency Grid

The row for any pixel within the desired pixel neighborhood will fall into one of three cases: either an adjacent pixel's row will be less than the minimum row for that pixel's column within the scan, the adjacent pixel's row will be equal or greater than the minimum row but less than or equal to the maximum row for that pixel's column within the scan, or the adjacent pixel's row will be greater than the maximum row for that pixel's column within the scan.

If an adjacent pixel's row is less than its column's minimum row in the current scan, then the nearest row of a geographically like pixel from the previous scan, as predetermined in the initTable process, is stored instead of the row of the pixel in the current scan. The exact formula for determining the row for the substitute pixel is shown below. Also, any column offset determined in initTable is applied to the column value that is stored for the adjacent pixel.

$$\text{Substitute Pixel's Row} = 1 + adj_row - minimum_row + closest_row \qquad \text{Eqn. 2.}$$

Where:

adj row = The row number of the original adjacent pixel requiring substitution in the current scan.

minimum row = The row number of first non-pixel trim row of the current scan.

closest row = Closest row in the previous scan that is geographically closest to the minimum row in the current column for the current scan.

If an adjacent pixel's row value falls between its column's minimum and maximum rows, as predetermined in the initTable process, then no substitution is necessary; and that pixel's row and column location is stored in the vector for later retrieval.

Lastly, if an adjacent pixel's row is greater than its column's maximum row within the current scan, then the nearest row of a geographically like pixel from the next scan, as predetermined in

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

the initTable process, is substituted. This substitute pixel's row and offset corrected column is then stored in the collecting vector in place of the original pixel's location. The exact formula for determining the row of the substitute pixel is as follows.

$$\text{Substitute Pixel's Row} = 1 + adj_row - maximum_row - 1 + closest_row \qquad \text{Eqn. 3.}$$

Where
adj row = The row number of the adjacent pixel requiring substitution in the current scan.
maximum row = The row number of the last non-pixel trim row of the current scan.
closest row = Closest row in the next scan that is geographically closest to the maximum row in the current column for the current scan.

2.1.3.2.2 getDistance

The getDistance method is used by initTable to determine the distance between the minimum and maximum row indexed pixels and all pixels in the same column in the previous (for minimum row indexed pixel) and next (for maximum row indexed pixel) scans. The previous scan index and next row index are then assigned based on minimum distance between the respective minimum and maximum row indexed pixels. From Figure 3, the nearest neighbor in column 639 for minimum row index pixel 18 is 10 (previous scan row index) and the nearest neighbor in column 639 for maximum row index pixel 29 is 37 (next scan row index).

Due to the rotation of the earth, the identified previous scan row index and next scan row index may not be the nearest neighbor to the minimum row index and maximum row index pixels. Subsequently, the distances between the pixels in the adjacent columns of the previous and next scan indexed pixels and the minimum and maximum row index pixels are calculated and if the distance is less than that for the previous and next scan indexed pixels, a column offset is applied. For example, if the distance calculation showed that the distance between the pixel in row 10, column 640 and the minimum row indexed pixel (column 639) was less than that for the pixel in row 10, column 639, then a previous scan column offset of +1 would be applied (see Figure 4).

In order to determine the adjacent scan row that is geographically nearest to a current scan's minimum or maximum row, as well as the nearest adjacent scan column to the current scan's column, geographic distances are calculated using the getDistance method. The distance formula used by the getDistance method is dependent on whether the Common Adjacency algorithm constructor code was supplied grid row column data or geolocation data.

If the Common Adjacency table is constructed using grid row column data (the operational default), where granule rows and columns may be converted to global grid row and column values, then the Pythagorean Theorem is used to determine distances.

$$Distance^2 = \left((gridrow1 - gridrow2)^2 + (gridcol1 - gridcol2)^2\right) \qquad \text{Eqn. 4.}$$

Where:
gridrow1 = global grid row of first pixel location
gridcol1 = global grid column of first pixel location
gridrow2 = global grid row of second pixel location
gridcol2 = global grid column of second pixel location

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

Note that the square root of the sum is not necessary because the calling Common Adjacency methods are only concerned with relative differences in distance.

If the grid-row-column data are not available, then the great circle distance formula from spherical geometry is used with provided geolocation data. To compute the actual arc length, the angular distance (denoted by AngDistance) should be multiplied by the radius of the sphere (e.g. the Earth radius). However, since the Common Adjacency methods are only concerned with relative differences in distance, comparing the angular distance is sufficient.

$$AngDistanœ = \cos^{-1}\left(\sin(lat1)\cdot\sin(lat2)+\cos(lat1)\cdot\cos(lat2)\cdot\cos(lon1-lon2)\right) \qquad \text{Eqn. 5.}$$

Where:

lat1 = latitude of first pixel location
lon1 = longitude of first pixel location
lat2 = latitude of second pixel location
lon2 = longitude of second pixel location

Again, the operational default is to use the grid-row column as this is more efficient since the grid row column distance calculation only involves two integer multiplications, two integer subtractions, and one integer addition whereas the geolocation distance calculation involves six floating point trigonometric functions, three floating point multiplications, one floating point addition and one floating point subtraction.

2.1.3.2.3 setCurrentPixel

The setCurrentPixel method takes as parameters the row and column of the pixel being processed by the calling algorithm (i.e. target pixel) and the pixel neighborhood radius. The method then saves the row and column value for of all pixels in the required neighborhood of the target.

2.1.3.2.4 getNextAdjPixel

The getNextAdjPixel method supplies the row and column of each successive adjacent pixel that was previously stored in a vector by the setCurrentPixel method, incrementing an internal counter on each call. The method returns *True* for each pixel data retrieval and returns *False* when the internal counter equals the vector size indicating that all of the pixel row/column pairs have been retrieved.

2.1.3.2.5 resetAdjIndex

This method resets to zero the internal counter that is incremented every time that getNextAdjPixel is called. This allows the calling algorithm to retrieve all of the stored adjacent pixel data a second time.

2.1.3.3 ProCmnAdjIMGTable

Derived from ProCmnAdjTable, this class is used by ProCmnAdjFactory to build the adjacency table for Imagery resolution scans. This table accounts for both onboard pixel trim and also earth rotation between scans.

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

2.1.3.4 ProCmnAdjMODTable

Derived from ProCmnAdjTable, this class is used by ProCmnAdjFactory to build the adjacency table for Moderate resolution scans. This table accounts for both onboard pixel trim and also earth rotation between scans.

2.1.3.5 ProCmnAdjPixel

A container class for adjacent pixel row/column data, one pixel per object.

2.1.3.6 ProCmnAdj

A set of C code wrappers for the C++ ProCmnAdjIMGTable and ProCmnAdjMODTable functionality primarily for use in Fortran code. The Fortran interface is implemented via ProCmnAdjInf.

2.1.3.7 ProCmnAdjInf

A Fortran interface to the Pro Common Adjacency functionality.

2.1.3.7.1 ProCmnAdjInf_resetAdjIndex ()

Fortran interface to resetAdjIndex.

2.1.3.7.2 ProCmnAdjInf_setCurrentPixel ()

Fortran interface to setCurrentPixel.

2.1.3.7.3 ProCmnAdjInf_getNextAdjPixel ()

Fortran interface to getNextAdjPixel.

2.1.3.8 Interfaces

2.1.3.8.1 Interfacing with Common Adjacency

The Common Adjacency library of functions was developed because several algorithms use adjacent pixels for their data processing. For example, Cloud Mask algorithm uses Common Adjacency utilities to process cloud adjacency. Interfacing with common adjacency can be done as shown below in Figure 6: Common Adjacency Sequence Diagram.

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

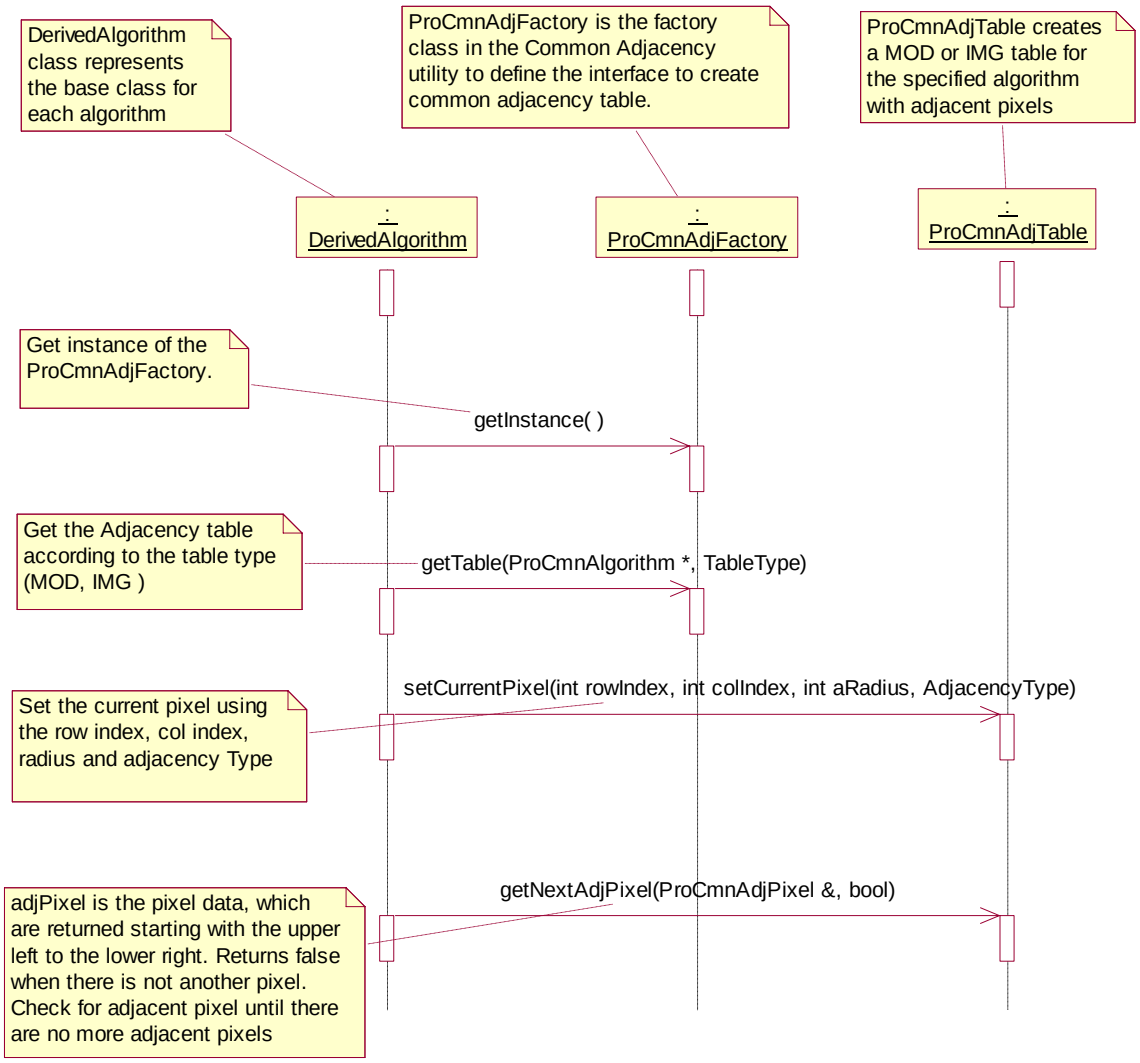

Figure 6: Common Adjacency Sequence Diagram

See the steps below for more detail.

2.1.3.8.2 Get the Common Adjacency table

The Common Adjacency utility uses the factory method pattern, which is an object oriented design pattern that defines the interface used to create the common adjacency table which contains the adjacent pixels. ProCmnAdjFactory creates either a Moderate resolution adjacency table or an Imagery resolution adjacency table, depending on the desired resolution.

adjTable = ProCmnAdjFactory::getInstance().**getTable**(alg , ProCmnAdjFactory::MOD_TABLE);

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

OR

adjTable = ProCmnAdjFactory::getInstance().**resetAdjTable**(alg , ProCmnAdjFactory::MOD_TABLE);

There are cases where more than one algorithm gets processed as part of a controller, and there may be a need to reset the Adjacency table using the **resetAdjTable**. For example, in the mask controller, Active Fires uses two adjacent scans for cross scan processing and Cloud Mask uses five adjacent scans for cross scan processing. The cross scan information for common adjacency is defined in configuration file PRO_CMN_ADJ_CFG.xml (**CrossGranScans** and the default value is 2). However, if an algorithm requires a different number of scans than the default value for cross scan processing, the **CrossGranScans** value may be overridden in the algorithm specific configuration file. In the Cloud Mask configuration file (PRO_VIIRS_CM_CFG.xml), the value of **CrossGranScans** is set to “5” to use five adjacent scans for cross scan processing. Therefore, when Cloud Mask is processed using the mask controller it is necessary to use **resetAdjTable** function which resets the adjacency table with five adjacent scans.

The first parameter of the getTable and resetAdjTable calls (in the examples above) is a pointer to the calling algorithm. The second parameter used above is either the MOD_TABLE constant, or the IMG_TABLE constant, as desired for the calling algorithm’s resolution.

2.1.3.8.3 Using a Common Adjacency table

Once the calling algorithm has obtained an adjacency table, the caller may loop through all the pixels in the granule that need to be processed and retrieve the adjacent pixels.

adjTable->**setCurrentPixel**(rowIndex, columnIndex, aRadius, ProCmnAdjTable::NO_CENTER);

VIIRS Cloud Mask uses aRadius = 1; which is the number of adjacent pixels to include in each direction.

The above example uses the AdjacencyType = NO_CENTER retrieval type constant which means retrieve every pixel within the radius except the center pixel. Other retrieval type constants are ALL_PIXELS, where all pixels within the radius including the center are returned, and EDGE_PIXELS, where only the pixels on the extremities of the radius range are returned.

2.1.3.8.4 Find adjacent pixel and perform algorithm specific processing

As the calling algorithm loops through each granule pixel, it checks for pixels adjacent to that granule pixel until there are no more adjacent pixels. When an adjacent pixel is available, the calling algorithm performs the specific processing needed (for example: Cloud Mask checks the Cloud confidence of the adjacent pixel and reports the most extreme confidence value):

adjTable->**getNextAdjPixel**(adjPixel, false);

The first parameter, adjPixel is the pixel data which is returned starting with the upper left to the lower right. Use adjPixel.getRow() and adjPixel.getCol() values to locate the corresponding

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

adjacent pixel value needed. This function returns false when there is not another pixel to retrieve. The second parameter is set to true if the caller wants fill pixels returned and false otherwise.

Below is an example how adjValue is used in Cloud Mask to determine the corresponding cloud confidence value:

```
adjValue = *(flags->cloud_confidence + (adjPixel.getRow() * M_VIIRS_SDR_COLS) +
adjPixel.getCol());
```

Check if (adjValue == CM_CONF_CLOUDY) or if (adjValue == CM_PROB_CLOUDY) or if (adjValue == CM_PROB_CLEAR) and set the adjacent pixel value and cloud adjacency flag accordingly.

2.1.4 Graceful Degradation

2.1.4.1 Graceful Degradation Input

None.

2.1.4.2 Graceful Degradation Processing

None.

2.1.4.3 Graceful Degradation Output

None.

2.1.5 Exception Handling

The getDistance method sends debug message when distance type is not specified and non-valid distance is returned.

2.1.6 Data Quality Monitoring

None.

2.1.7 Computational Precision Requirements

All distance calculations are done at double precision with distances stored as 32-bit floats which are sufficient precision for determining which pixels are nearest neighbors to reference pixel.

2.1.8 Algorithm Support Considerations

None.

2.1.9 Assumptions and Limitations

None.

2.1.10 Science Support References

The common adjacency library of functions is unique to the operational baseline—i.e. no corresponding science baseline.

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

3.0 GLOSSARY/ACRONYM LIST

3.1 Glossary

The current glossary for the NPOESS program, D35836_G_NPOESS_Glossary, can be found on eRooms. Table 8 contains those terms most applicable for this OAD.

Table 8: Glossary

Term	Description
Algorithm	A formula or set of steps for solving a particular problem. Algorithms can be expressed in any language, from natural languages like English to mathematical expressions to programming languages like FORTRAN. On NPOESS, an algorithm consists of: A theoretical description (i.e., science/mathematical basis) A computer implementation description (i.e., method of solution) A computer implementation (i.e., code)
Algorithm Configuration Control Board (ACCB)	Interdisciplinary team of scientific and engineering personnel responsible for the approval and disposition of algorithm acceptance, verification, development and testing transitions. Chaired by the SEIT Lead or representative, members include representatives from all stakeholder IPTs and the IPO.
Ancillary Data	Any data which is not produced by the NPOESS System, but which is acquired from external providers and used by the NPOESS system in the production of NPOESS data products.
Auxiliary Data	Auxiliary Data is defined as data, other than data included in the sensor application packets, which is produced internally by the NPOESS system, and used to produce the NPOESS deliverable data products.
EDR Algorithm	Scientific description and corresponding software and test data necessary to produce one or more environmental data records. The scientific computational basis for the production of each data record is described in an ATBD. At a minimum, implemented software is science-grade and includes test data demonstrating data quality compliance
Environmental Data Record (EDR)	[IORD Definition] Data record produced when an algorithm is used to convert Raw Data Records (RDRs) to geophysical parameters (including ancillary parameters, e.g., cloud clear radiation, etc.). [Supplementary Definition] An Environmental Data Record (EDR) represents the state of the environment, and the related information needed to access and understand the record. Specifically, it is a set of related data items that describe one or more related estimated environmental parameters over a limited time-space range. The parameters are located by time and Earth coordinates. EDRs may have been resampled if they are created from multiple data sources with different sampling patterns. An EDR is created from one or more NPOESS SDRs or EDRs, plus ancillary environmental data provided by others. EDR metadata contains references to its processing history, spatial and temporal coverage, and quality.
Operational Code	Verified science-grade software, delivered by an algorithm provider and verified by IWPTB, is developed into operational-grade code by the IDPS IPT.
Operational-Grade Software	Code that produces data records compliant with the System Specification requirements for data quality and IDPS timeliness and operational infrastructure. The software is modular relative to the IDPS infrastructure and compliant with IDPS application programming interfaces (APIs) as specified for TDR/SDR or EDR code

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

Term	Description
Raw Data Record (RDR)	[IORD Definition] Full resolution digital sensor data, time referenced and earth located, with absolute radiometric and geometric calibration coefficients appended, but not applied, to the data. Aggregates (sums or weighted averages) of detector samples are considered to be full resolution data if the aggregation is normally performed to meet resolution and other requirements. Sensor data shall be unprocessed with the following exceptions: time delay and integration (TDI), detector array non-uniformity correction (i.e., offset and responsivity equalization), and data compression are allowed. Lossy data compression is allowed only if the total measurement error is dominated by error sources other than the data compression algorithm. All calibration data will be retained and communicated to the ground without lossy compression. [Supplementary Definition] A Raw Data Record (RDR) is a logical grouping of raw data output by a sensor, and related information needed to process the record into an SDR or TDR. Specifically, it is a set of unmodified raw data (mission and housekeeping) produced by a sensor suite, one sensor, or a reasonable subset of a sensor (e.g., channel or channel group), over a specified, limited time range. Along with the sensor data, the RDR includes auxiliary data from other portions of NPOESS (space or ground) needed to recreate the sensor measurement, to correct the measurement for known distortions, and to locate the measurement in time and space, through subsequent processing. Metadata is associated with the sensor and auxiliary data to permit its effective use.
Retrieval Algorithm	A science-based algorithm used to 'retrieve' a set of environmental/geophysical parameters (EDR) from calibrated and geolocated sensor data (SDR). Synonym for EDR processing.
Science Algorithm	The theoretical description and a corresponding software implementation needed to produce an NPP/NPOESS data product (TDR, SDR or EDR). The former is described in an ATBD. The latter is typically developed for a research setting and characterized as "science-grade".
Science Algorithm Provider	Organization responsible for development and/or delivery of TDR/SDR or EDR algorithms associated with a given sensor
Science-Grade Software	Code that produces data records in accordance with the science algorithm data quality requirements. This code, typically, has no software requirements for implementation language, targeted operating system, modularity, input and output data format or any other design discipline or assumed infrastructure
SDR/TDR Algorithm	Scientific description and corresponding software and test data necessary to produce a Temperature Data Record and/or Sensor Data Record given a sensor's Raw Data Record. The scientific computational basis for the production of each data record is described in an Algorithm Theoretical Basis Document (ATBD). At a minimum, implemented software is science-grade and includes test data demonstrating data quality compliance
Sensor Data Record (SDR)	[IORD Definition] Data record produced when an algorithm is used to convert Raw Data Records (RDRs) to calibrated brightness temperatures with associated ephemeris data. The existence of the SDRs provides reversible data tracking back from the EDRs to the Raw data. [Supplementary Definition] A Sensor Data Record (SDR) is the recreated input to a sensor, and the related information needed to access and understand the record. Specifically, it is a set of incident flux estimates made by a sensor, over a limited time interval, with annotations that permit its effective use. The environmental flux estimates at the sensor aperture are corrected for sensor effects. The estimates are reported in physically meaningful units, usually in terms of an angular or spatial and temporal distribution at the sensor location, as a function of spectrum, polarization, or delay, and always at full resolution. When meaningful, the flux is also associated with the point on the Earth geoid from which it apparently originated. Also, when meaningful, the sensor flux is converted to an equivalent top-of-atmosphere (TOA) brightness. The associated metadata includes a record of the processing and sources from which the SDR was created, and other information needed to understand the data.

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

Term	Description
Temperature Data Record (TDR)	[IORD Definition] Temperature Data Records (TDRs) are geolocated, antenna temperatures with all relevant calibration data counts and ephemeris data to revert from T-sub-a into counts. [Supplementary Definition] A Temperature Data Record (TDR) is the brightness temperature value measured by a microwave sensor, and the related information needed to access and understand the record. Specifically, it is a set of the corrected radiometric measurements made by an imaging microwave sensor, over a limited time range, with annotation that permits its effective use. A TDR is a partially-processed variant of an SDR. Instead of reporting the estimated microwave flux from a specified direction, it reports the observed antenna brightness temperature in that direction.

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

3.2 Acronyms

The current acronym list for the NPOESS program, D35838_G_NPOESS_Acronyms, can be found on eRooms. Table 9 contains those terms most applicable for this OAD.

Table 9: Acronyms

Term	Expansion
ATBD	Algorithm Theoretical Basis Document
DPE	Data Processing Element
O&S	Operations and Sustainment
VIIRS	Visible Infrared Imager Radiometer Suite

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO

4.0 OPEN ISSUES

Table 10: List of OAD TBD/TBR

No.	DESCRIPTION	Resolution Date
None		

D48316, ---. PDMO Released: 2010-06-22 (VERIFY REVISION STATUS)
PDMO