# DGDataReader

## What this class does:

DGDataReader is a C++ class that loads Tiff imagery and associated meta-data files, it has been used primarily with images taken from Digital Globe’s World View 3 satellite.

### Open Source Projects Utilized:

Tinyxml2 (ZLib License) <https://codeload.github.com/leethomason/tinyxml2/zip/master>

LibTiff <http://bigtiff.org/libtiff.lib-4.1.zip>

Libttiff copy right stuff <http://www.libtiff.org/misc.html>

### Limitations:

The files Digital Globe provides are technically GeoTiffs. This class reads them, but not the GeoTags. The class requires the Digital Globe XML meta-data files for there “B1” data in order to extract mapping data.

## Algorithmic Walkthrough

## Class Style Guide

1. Readability was chosen over efficiency so that debuggers and porters could follow the algorithm easily
2. the class members and methods are grouped by overall function
3. All member variables are private.
4. All member variables have public get/set accessor functions.
5. When appropriate member variables have update accessor functions.
6. Static variables start with a lowercase "s".
7. Member variables start with a lowercase "m".
8. Local variables start with a lowercase "l".
9. Loop variables are a single lowercase letter. In our case we used "i"
10. Method return variables are all labeled "rvalue". They are used to insure that a method only has 1 exit point.
11. Return values (rvalue) are defaulted to indicate failure either through assignment to "false" or "NaN"
12. Accessor methods start with a lowercase "a".
13. Calculation methods start with a lowercase "c".
14. Void returns are discouraged.

## Class Structures

struct GeodeticPoint3D{

double lat;

double lon;

double alt;

## };

The GeodeticPoint3D structure is used to hold the standard latitude, longitude, and altitude coordinate tuple.

## Class Static Members

This class has no static members

## Class Instance Members

RPCMapper mMapper;

int mNumBands;

vector<vector<GeodeticPoint3D>> mGridPoints;

vector<vector<RPCMapper::ImageCoordinates>> mGridImageCoordMappings;

vector<vector<std::map<int,unsigned short>>> mGridDataMappings;

vector<unsigned short \*> mImages;

double mGSD;

vector <unsigned char \*> mScaledImages;

double mScaleFactor;

vector <unsigned char> mScaleShifts;

int mImageRowLength;

int mImageRowsTotal;

mMapper:

The RPCMapper object that is used to project the TIFF image to a 2D plane in object space

mNumBands:

The number of collected bands present in the image

mGridPoints:

A vector of a vector of 3D points. Its purpose is to create a 2D grid full of Geodetic points that can be projected back to the image via RPCs

mImageCoordMappings:

Holds a 2D grid of the same size as mGridPoints, but instead of holding the Geodetic point, it holds the image sensor location that the geodetic point maps to via the RPCs

mGridDataMappings:

Holds a 2D grid of the same size as mGridPoints, but instead of holding the Geodetic point or the image coordinates, it holds the intensity value of the image at the mapped to point. Think of this as a 2D plane that has been painted with the image in the same way a projection screen is painted by a projector.

mImages:

a vector of short pointers that hold the address of the beginning of the band data for all numBands datasets in the image.

mGSD:

The mean GSD of the image.

mScaledImages:

a vector of pointers to image data that has been scaled to 8 bits.

mScaleFactor:

Legacy variable that was used to linearly scale 12 bit data to 8 bits.

mScaleShifts:

a vector that holds – for each band – the number of right shifts needed to get 90% of that bands data under 8 bits

mImageRowLength:

Holds the number of pixels in a row.

mImageRowsTotal:

Holds the Height of the image.

## Class Accessor Methods

### Get Methods

int aGetNumBands();

const vector<vector<GeodeticPoint3D>> \* aGetGrid();

GeodeticPoint3D aGetGridPoint(int xIndex, int yIndex);

RPCMapper::ImageCoordinates aGetGridImageCoord(int xIndex, int yIndex);

unsigned short aGetGridPaint(int xIndex,

int yIndex,

int bandIndex);

double aGetGSD();

const unsigned short \* aGetImage(int bandIndex);

int aGetImageRowLength();

int aGetImageRowsTotal();

RPCMapper aGetMapper();

const unsigned char \* aGetScaledImage(int bandIndex);

double aGetScaleFactor();

unsigned char aGetScaleShift(int bandIndex);

The get elements are straight forward and do what you would expect. If they have no index associated with them, then they return the values of their corresponding variables. If they do have indexes, they return the element that corresponds to the indexed location(s) of the variable that the get method correlates with.

### Set Methods

bool aSetNumBands(int numBands);

bool aSetGridPoint(int xIndex, int yIndex, GeodeticPoint3D Point);

bool aSetGridImageCoord(int xIndex, int yIndex,

RPCMapper::ImageCoordinates ImageCoord);

bool aSetGSD(double value);

bool aSetImageRowLength(int pixelsPerRow);

bool aSetImageRowsTotal(int RowsPerImage);

bool aSetScaledImages(int gridOffsetX, int gridOffsetY,

int xSize, int ySize);

bool aSetScaleFactor(double value);

bool aSetScaleShifts(int gridOffsetX, int gridOffsetY,

int xSize, int ySize);

The set methods function how you would expect as well. If the method just takes one variable. That variable is assigned to the corresponding class member. If there is an index that gets passed to the set method, then the indexed location of the class member is assigned the value passed.

### Initialization Methods

bool aIntializeMapper();

bool aInitializeDefaultGrid(double alt);

bool aLoadImages(char\* filename);

bool aLoadXMLDoc(char\* filename);

aInitializeMapper:

Initializes the RPC mapper and builds the RPC Bounding box.

aInitializeDefaultGrid:

Generates a default 2D grid at the supplied altitude. The grid’s lateral boundary is defined by the RPC Bounding box latitude and longitude extrema, and the grid is populated with one point for every gsd footprint between the lateral boundary.

aLoadImage:

Loads a tiff image and populates all the relevant class members with information about the loaded file.

aLoadXMLDocument:

Loads the image meta-data such as RPC coefficients and mean GSD

## Class Calculation Methods

bool cMapData();

bool cGenerateGridData();

GeodeticPoint3D cDeltas(double lat);

cMapData:

Creates and populates the grid that hosts the image coordinates that the LLA grid would map to at shared indices.

cGenerateGridData:

Creates and populates (paints) the grid that hosts the image intensities that the LLA grid would map to at shared indices.

cDelta:

Generates the meters per degree conversion factors for latitude and longitude movements at the supplied latitude. This is used to determine how to carve up the grids between grid boundries.

## How to Use This Class:

### Prerequisites:

1. This code requires the header files DGReader.h, tiffio.h and tinyXML2.h to be included via the #include directive and for all three to be in the include library path of the compiler being used.
2. libTIFF has a .lib file that must be available during compilation i.e. it must be in a directory that the linker will check.

libTIFF:

<http://bigtiff.org/libtiff.lib-4.1.zip>

tinyxml2:

<https://codeload.github.com/leethomason/tinyxml2/zip/master>

### Code Execution:

//load the XML & TIFF stuff

DGDataReader \* SWIRImage = new DGDataReader();

SWIRImage->aLoadXMLDoc(PathtoXMLFile);

SWIRImage->aIntializeMapper();

SWIRImage->aLoadImages(PathtoTIFFFile);

This class also has a capability to project an image onto an arbitrary constant height 2D plane via the following commands:

//Map Image to Groundplane

SWIRImage->aInitializeMapper();

SWIRImage->aInitializeDefaultGrid(altitude);

SWIRImage->cGenerateGridData();