I. Executive Summary

A. Background

In October of 1999, NRO AS&T and NIMA cosponsored the Multi-Modal Image Fusion Conference in Rochester NY. The conference focused on understanding 'non-traditional' companies, methodologies, and business practices that could potentially be applied to government missions and requirements. ImageLinks, Inc. of Melbourne Florida attracted the interest of numerous government organizations with their on-demand touchless image processing capabilities (Active Archive™) and their open source approach to software development. This subsequently resulted in the award of the Open Source Software Image Map (OSSIM) experiment.

B. OSSIM Capabilities

The OSSIM library consists of over 20,000 lines of new C++ object oriented source code. It connects to a GRASS library that allows direct access to over 70 GIS, Remote Sensing, and Image Processing functions. The library can handle arbitrarily large images with arbitrary radiometries and is capable of hyperspectral image data manipulation. A dozen existing applications and utilities already have been created to take advantage of the library. A bridge to the GDAL library allows direct input and output of over 20 geo-spatial data formats. All of the GEOTRANS map projections and datums that have been certified by NIMA are implemented and the library provides for dynamic reprojection capabilities. Caches and reduced resolution data sets have been implemented allowing high performance panning and zooming of extremely large data sets. The OSSIM library also contains arbitrary grid overlays for displaying latitude and longitude on the displayed or processed imagery. A real time tracking cursor displays the exact latitude and longitude anywhere within the displayed image. Florida Tech has implemented automated feature extraction capabilities into the library. The detailed documentation of the code consists of over 4000 pages of cross referenced source code and descriptions. Online configuration management tools have captured all submitted code and snapshots of every development release. All source code, documentation, and test data sets are available online and are open source.

II. OSSIM Software Development

This section describes the features of OSSIM as of the end of Phase I. With the dawn of Phase II, we expect OSSIM to incorporate many more features, as we progress toward an increasingly robust, reliable, scalable remote-sensing data processor.

A. Phase 1 Capabilities

The capabilities we discuss here include:

- Image formats

- Georeferenced information: projections, datum (reference points), and latitude / longitude
- Keyword lists, session state history and recall
- Graphical User Interface
- Scripting and XML support
- Command-line utilities
- Feature extraction
- Integration with GIS-processing (GRASS) commands

1. Supported formats (Input and Output)

Remotely sensed images are captured for computer processing by filtering radiation emanating from the image into various electromagnetic wavelength bands, converting the overall intensity for each band to digital format, and storing the values on computer compatible media such as magnetic tape. (Excerpted from the GRASS web site, www.baylor.edu/~grass/.)

This section describes the image formats that OSSIM can read. Write capability also exists for every format except where noted as read-only. OSSIM has three means to interpret various image formats:

- Internal code support for Image Handlers and Image Writers
- Integration with the Geospatial Data Abstraction Library (GDAL)
- Integration with the Geographic Resources Analysis Support System (GRASS)

a. OSSIM Internal Code Support

OSSIM has no native format, which allows the OSSIM system to incorporate unlimited image handlers and image writers for unique image types. The image formats described here are tested with OSSIM for Phase I. We expect OSSIM to handle many more image formats in the future.

ADRG -- Digitized Raster Graphics

Data originally produced by the National Imagery and Mapping Agency (NIMA) as paper maps. A paper product is scanned into digital data files at a nominal resolution of 100 microns (254 lines per inch) and in 24-bit RGB color. Data collected from a single chart or map series and scale become part of a worldwide seamless database of raster graphic data with each pixel having a distinct geographic location. Location is calculated according to the Equal Arc-Second Raster Chart/Map (ARC) system frame of reference with reference to the WGS84 datum. (See Datum Support on page 7.)

The ARC system divides each hemisphere into nine zones of ten degrees of latitude to minimize distortion. ADRG data is divided into geographic data sets called Distribution Rectangles (DRs), then one or more DRs are stored on a single CD-ROM.

BIL -- Band Interleaved by Line

A Raw Raster image that is organized into consecutive line sets. For example, in a red-blue-green (RGB) image: each red line is followed by a green line, and then by a blue line.

BIP -- Band Interleaved by Pixel

A Raw Raster image that is organized into pixel sets. For example, in an RGB image each point in the image is represented by a red pixel, green pixel, and blue pixel.

BSQ -- Band Sequential

A Raw Raster image that consists of separate sections for each band. For example, in an RGB image, all red pixels (the red band) exist in a block, followed by the entire green band, then the entire blue band.

BSQ-multifile

A Raw Raster image that is comprised of a separate file for each band For example, an RGB image consists of three input files. One file contains all the red pixels, another, the green pixels, and a third file contains all the blue pixels.

Chip/Chunk Format (ccf)

Both in the Harris MET™ and the ImageLinks EMET™ systems store images in this proprietary format. A chip contains 32 by 32 pixels, a chunk contains 8 by 8 chips, or 256 by 256 pixels. Using this scheme, you can view a maximum of 4 chunks on a 512 by 512 screen. Note that while you can load ccf images into OSSIM, conversion to ccf is not required by OSSIM.

DOQ- Digital Orthophoto Quadrangle

A DOQ digital, uniform-scale image is created from aerial photos. The DOQ format produces a photographic map in which ground features are displayed in their true ground position, because relief displacements caused by the camera and terrain of an aerial photograph have been removed. Because DOQ image data combines the image characteristics of a photograph with the geometric qualities of a map, software that can interpret DOQ information can extract direct measurements of distances, areas, angles, and positions from the image.

GeoTIFF

A georeferenced version of the popular Tagged-Image File Format (TIFF). A program that understands GeoTIFF can read the geographic reference information contained in the image along with the image pixel data.

Raw Raster

A binary matrix of pixels that contains no header to tell OSSIM how to project the image. You must create a keyword list from the raster keyword template (see Keyword Support on page 10) to describe the image for OSSIM.

b. Grass Cells

The GRASS programs can read LANDSAT multi-spectral scanner (MSS) data, LANDSAT thematic mapper (TM) data, and other formats, such as scanned aerial photography or SPOT satellite data. They extract the band data into raster files in a GRASS database. Each band becomes a separate raster file, with standard GRASS map layer support, and can be displayed and analyzed just like any other raster file.

The band data extracted from tapes are assumed to be unregistered data. This means that the GRASS software does not know the earth coordinates for pixels in the image. The only coordinates known at the time of extraction are the columns and the rows relative to the way the data was stored on the tape. Data can only be extracted into a database which has an x,y coordinate system, and not into a projected database (such as a UTM database). However, OSSIM enables the user to select a projection for a GRASS image by supplying a keyword list.

c. GDAL Formats

Integration with GDAL provides OSSIM with the capacity to integrate the additional image formats described in this section.

CEOS image
: Committee on Earth Observation Satellites (CEOS) membership encompasses the world's government agencies responsible for civil Earth observation (EO) satellite programs, along with agencies that receive and process data acquired remotely from space.

SAR_CEOS -- CEOS SAR Image
: GDAL provides read-only access to CEOS Synthetic Aperture Radar (SAR) image files, which include most Radarsat International and European Space Agency satellite (ERS) data products.

DTED - Digital Terrain Elevation Data
: Created by or for the U. S. Department of Defense (National Imagery and Mapping Agency), DTED images contain a uniform matrix of terrain elevation values. They provide basic quantitative data for all military systems that require terrain elevation, slope, and/or gross surface roughness information.

EFF -- EOSAT Fast Format
: EOSAT is a satellite imaging company that distributes Landsat images. EOSAT datasets normally consist of one or more raster data files For example, a BSQ (band sequential) multifile image consists of a separate file to contain each band of data (BAND1.DAT, BAND2,DAT, BAND3.DAT). A corresponding header file, which must be called HEADER.DAT, accompanies each dataset. To open an EFF dataset, the user selects the HEADER.DAT file.

EHdr. - ESRI headed data
: ESRI is a GIS and Mapping software developer. Their flagship product is ArcView. GDAL enables OSSIM to read and write images that are ArcView-compatible.

GIO - Arc/Info Binary Grid

Another format for ESRI data, GIO format encompasses ArcInfo GIS grids. The Arc/Info Binary Grid format is the internal working format of the Arc/Info Grid product. It is also usable and creatable within the spatial analyst component of ArcView.

GXF -- Grid eXchange File

This raster exchange format, propagated by Geosoft, is now a standard format in the gravity/magnetics field. GDAL can read, write, GXF, including support for georeferencing information, and projections.

HKV -- Atlantis HKV Blob

The HKV raster file format that is primarily used internally by the Atlantis Scientific InSAR processing system.

MFF -- Atlantis MFF Raster

Atlantis Scientific's MFF raster format dataset consist of a header file (typically with the extension .hdr) and a set of data files with extensions like .x00, .b00 and so on.

PAux -- PCI.aux raw raster format

PCI Geomatics is a world leading developer of Geomatics software (geographic modeling, measurement, analysis, and output) and solutions based on its remote sensing, digital photogrammetry[1], spatial analysis, and cartographic editing programs. The aux file contains projection and view information about the raw image to which it corresponds.

PNG -- Portable Network Graphics

GDAL supports greyscale, pseudo-colored, RGB and RGBA PNG files in precisions of eight and sixteen bits per sample. For the Web, PNG really has three main advantages over GIF: alpha channels (variable transparency), gamma correction (cross-platform control of image brightness), and two-dimensional interlacing (a method of progressive display). PNG compression is lossless[2] and slightly more efficient than GIF. Saving, restoring and re-saving an image will not degrade its quality, unlike standard JPEG (even at its highest quality settings).

SDTS -- USGS SDTS DEM

USGS (Unites States Geological Survey) DEMs (Digital Elevation Models) are always returned with a data type of signed sixteen bit integer. Projection and georeferencing information is also returned.

---

[1] Photogrammetry is the process of making maps or scale drawings from photographs, especially aerial photographs or the process of making precise measurements by means of photography.

[2] A lossless compression method protects against errors from lost data.

2. GUI

The OSSIM graphical user interface (GUI) provides an easy-to-use viewer for images. (Refer to Figure [to be provided].) A user can open one image in many different windows on the computer screen, each with a different projection, band selection, area within the image, and resolution. Simultaneously, a user can open different images in different windows.

The GUI is intuitive to use and provides the following capabilities:

- Open an image format or keyword list.
- Dynamically reassign sensor radial bands to predominance within the image, in order to emphasize the features of interest to the viewing user.
- Alter the projection, datum, or image type of the input image and write the new view to disk for later reuse. (Refer to Geospacial Modeling Support on page 6 for an explanation of projection and datum.)
- Display dynamically the latitude and longitude of any point in the image.
- Pan and zoom an image in the view window.
- Turn on or off an overlaid grid of latitude/longitude lines.

The OSSIM team built the GUI in order to test the engine that will become integrated with Active Archive. As an added benefit, end users can now manipulate images using OSSIM without Active Archive. Having the GUI provided a rapid prototyping mechanism for the open-source contributors to OSSIM.

3. Geospacial Modeling Support

OSSIM incorporates the intelligence required to interpret all map-image information. This section describes some of those map-image features— projection, datum, latitude/longitude coordinates, and gridlines.

a. Projection Support

The Projection object defines the information that a particular algorithm uses to convert a point in a model space to a flat representation of that model that you can view on flat monitor screen or print to flat paper. OSSIM can convert map projections to a computer screen and even change an image from one projection model to another. A map projection converts a point (latitude / longitude) on the curved earth surface to a two-dimensional point (x,y) in a flat simulation of the earth.

There are many algorithms to perform the curved to flat projection of an area on the earth. OSSIM provides all of the map projection models in the USGS Professional Paper 1395, *Map Projections-A Working Manual*, by J.P. Snyder, as Figure 1 on page 8 shows.

b. Datum Support

A datum is reference point used by the surveyor who created the projection data for this area of the world. Each datum has an acronym that identifies it among the finite set of known datum reference locations. OSSIM uses the datum latitude/longitude as an offset to the world origin (0,0) when it calculates each latitude/longitude reading that you see in a viewport as you move your pointer over an image. For example, WGE refers to the World Geodetic System recorded in 1984 (WGS '84). For a more information about the various datum that are used to calculate projection latitude/longitude readings, refer to the USGS Professional Paper 1395, *Map Projections-A Working Manual*, by J.P. Snyder.

Figure 2 (page 10) shows how simply a user selects or changes the datum reference point for a particular image.

c. Latitude / Longitude and Grid Support

As a user moves a computer pointer, such as a mouse, over an image, the OSSIM graphical interface displays the precise latitude and longitude coordinates under the pointer position.

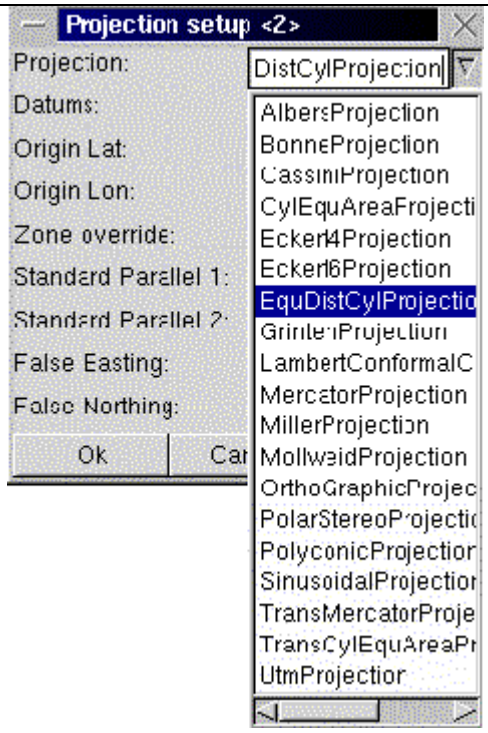

Figure 1: OSSIM Projection Choices

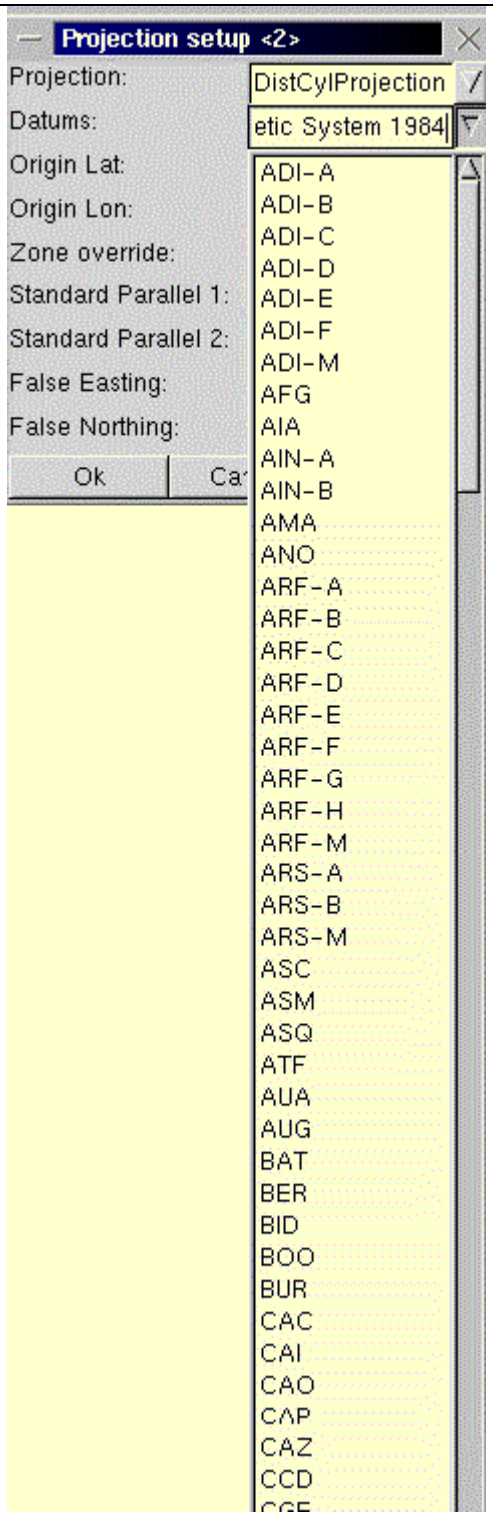

Figure 2: Selecting a Datum in OSSIM

If the user elects to also display a grid over the image view, that horizontal and vertical lines of the grid remain at true latitude/longitude distances from each other no matter what resolution the user zooms into or out of while viewing the image. (Figure 3.)

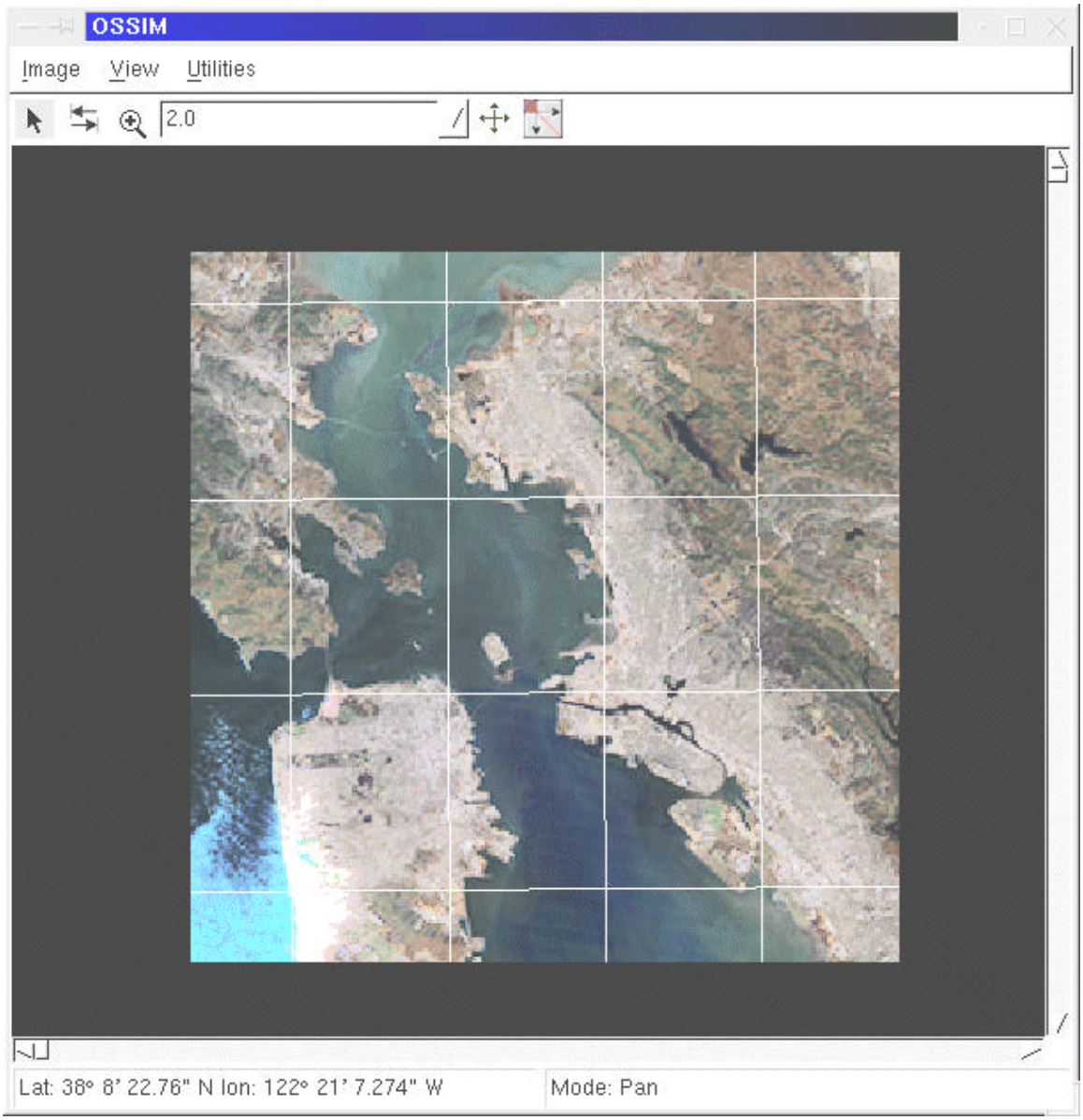

Figure 3: Image with Grid Overlay

4. Platform Independence (wxWindows)

OSSIM open source code is written in C++ and can be compiled with any C++ compiler. It is quite portable.

At present, you can compile and run OSSIM on a Linux, UNIX, Microsoft Windows or Macintosh host, though you must emulate Linux on your system. The emulation software is as openly distributed, accessible, and easy to install as OSSIM itself. In the future OSRS hopes to create a version that runs directly as a Microsoft Windows application without the necessity to emulate Linux. As a consortium of user-developers, we consider each request for an environment port seriously and try to solicit an OSRS participant to create a requested port. For example, a Mac devotee is welcome to create a port of OSSIM to use with the Macintosh operating and windowing system, then share it with other Mac users who have an interest in OSSIM.

5. Keyword Support

In generic terms, a keyword list consists of pairs of ASCII phrases. Each pair has a keyword parameter name and an optional value for the parameter. OSSIM interprets the keyword parameters and exhibits appropriate behavior for the particular value of each parameter.

For purposes of OSSIM, you specify a keyword with the format `name:[value]`. The square brackets indicate that not all keywords have a parameter value.

OSSIM has its own standardized keyword lists to describe the contents of an image, session state, and a variety of other variables that tell the OSSIM software how to display an image. For large volumes of data, or just to let a system process remote-sensing data in the background without human intervention, a user can run OSSIM in a command line shell rather than from the GUI and include a keyword list file as an input parameter to the `ossim` command.

As the OSSIM developers test various keyword lists with OSSIM, they add templates for the keyword lists in directory reserved for this purpose so that end users or other developers can fill in an appropriate set of keyword values to describe their images, projection specifications, or other information that enables OSSIM to recognize and handle incoming data. Figure 4 on page 10 shows a template that end-users fill in to describe a raster image for which no sensor intelligence exists in the image file itself.

Note: Figure 4 spans multiple pages.

Figure 4: Raster Keyword Template

A keyword list for a generic raster file includes parameters that tell OSSIM how to turn the binary information in the file into the appropriate sized rectangle of pixels. Some raster file keyword lists also provide information that enables OSSIM to distinguish bands in the image and other details. To submit a raster image to OSSIM,

1. Copy ossim/etc/templates/general_raster_template.kwl to ossim/bin/*your_raster*.kwl.
2. Fill in any of the following keyword values that pertain to your image.

Figure 4: Raster Keyword Template

```
// Source raster image file.  Required.
image_file:  /data/image1/image1.ras

// Pixel type.  Default = uchar (unsigned_char).
// Valid types:
// uchar = unsigned charactor  - 8 bits,  range: 0 - 255
// ushort16 = unsigned short   - 16 bits, range: 0 - 65535
// sshort16 =  signed short - 16 bits, range:  -32768 - 32767
// ushort11 - unsigned short 11 bits, range 0 - 2047
// float - 16 bit floating point
//       approx:  1.2 e-38 to 3.4e38 with 7 digit precision.
pixel_type:  uchar

// Number of bands. Default = 1.
//NOTE: You must specify bands if your image has multiple bands.
//The number of pixels per x,y position generally
//  equals number_bands. For example,
// if the image has 3 separate bands, one each for red, green,
// blue,it will also have 3 pixels for each point in the image.
//Exception: A psuedo-color image has 3 bands but only one pixel
// per position. Each pixel contains all three band values: red,
// green, and blue.
number_bands:  1
// Interleave type.   Default = bil .
// Valid types:
// bil = Band Interleaved by Line
//  Image is organized into consecutive line sets.
//  For example, in a rgb image, each red line is followed
//   by a green line, and then by a blue line.
//bip = Band Interleaved by Pixel
//  Image is organized in pixel sets
//  For example, in an rgb image each point in the image is
//  represented by a red pixel, green pixel, and blue pixel
// bsq = Band Sequential.
// Image consists of separate sections for each band.
//  For example, in an rgb image,
//  all red pixels (the red band) exist in a block, followed by
//  the  entire green band, then the entire blue band.
// bsq_multifile = each band is stored as a separate file
// For example, an rgb image consists of three input files
// One file contains all the red pixels, another, the green
//  pixels, a third file contains all the blue pixels.
interleave_type:  bil

//NOTE: The number_lines or number_samples are required.
//  If either value does not match the actual image resolution,
//  OSSIM displays garbage because it cannot project the image.
number_lines:  100
number_samples:  100
```

Figure 4: Raster Keyword Template

```
// NOTE: Use the valid_start/stop _line/sample keywords to
//       extract a rectangle from a raster file.
// Default for both valid_start_line and valid_start_sample is 0
//  because the upper left corner of the image is point (0, 0)
// Default for valid_stop_line and valid_stop_sample
// is the lower right corner of the image rectangle.
// REMOVE the comment slashes if you set any a value **
// valid_start_line:  0
// valid_start_sample:  0

// Valid stop line.  Default is computed to
//    (number_of_lines - 1).
// In this example, the default number_lines = 100.
// valid_stop_line:  99
// Valid stop sample. Default is computed to
//   (number of samples - 1).
// In this example, the default number_samples = 100.
// valid_stop_sample:  99
//NOTE: Use model_image_start/_stop _line/sample
// define the image start relative to geometry file.
// model_image_start_line: 0
// Model image stop line.   Default is (image_lines - 1).
// In this example, image_lines = 100.
// model_image_stop_line: 99
// model_image_start_sample: 0
// Model image stop sample.  Default is (image_samples - 1).
// In this example, image_samples = 100.
// model_image_stop_sample: 99
// Header size.  Default is 0.
// Use to specify a number of bytes that OSSIM should ignore
//      at the beginning of the image file.
// header_size: 0
// Set nulls mode.  Default = mode 0.
// mode 0:  Leave pixels as read.
// mode 1:  Flip zeroes to pixel-type  minimum (usually 1).
// mode 2:  Set pixels outside the edge to null value
//      (usually 0).
set_fill_to_nulls_mode: 0
// set_fill_to_nulls_mode: 1
// set_fill_to_nulls_mode: 2

// Pixels to chop.  Not required.  Default = 0.
// Set pixels on the edge of the imageline to null.
// Eliminates resampling error from the edge of the image while
// retaining the image dimensions.
// pixels_to_chop:
```

6. State Snapshot Capabilities

OSSIM automatically saves the exact view of an image whenever you save the viewport as a session. As Figure 5 shows, OSSIM employs a simple, text keyword list to keep the session state in a disk file. By opening a session keyword list instead of an image file, a user can reload a saved session view on demand.

Figure 5; Sample Session Snapshot

This sample keyword list shows values for a UTM projection. OSSIM automatically creates a keyword list when you load an image or change its projection in a viewport window. Keywords specify such map-projection information as projection type, meters per line (y) and sample (x), projection origin, and so forth. OSSIM knows from the values in this keyword list how to convert from latitude/longitude readings to easting northing points and vice versa.

OSSIM prefixes each keyword name with a viewport identifier. If you save a session in a particular OSSIM GUI viewport, the session state contains the type of keywords shown in this example, each prefixed with the same viewport number. Every time you save a session, the filename that OSSIM generates identifies the session. You can restore any session by loading a session file.

```
model.type:  OssimImage
ossim_image1.meters_per_pixel_y:  5.0000000000000
ossim_image1.meters_per_pixel_x:  6.000000000000000
ossim_image1.projection.type:  UtmProjection
ossim_image1.projection.central_meridian:  -122.338705029217
ossim_image1.projection.origin_latitude: 37.844451824360
ossim_image1.projection.datum:  WGE
ossim_image1.projection.false_easting:  0
ossim_image1.projection.false_northing:  0
ossim_image1.projection.hemisphere:  N
ossim_image1.projection.scale_factor:  1
ossim_image1.projection.std_parallel_1:  0
ossim_image1.projection.std_parallel_2:  0
ossim_image1.projection.zone:  10
ossim_image1.tie_point_x:  538185.0000
ossim_image1.tie_point_y:  4208785.0000
```

Figure 5; Sample Session Snapshot

Explanation of Keywords

meters_per_pixel_y
meters_per_pixel_x

The number of meters represented by each pixel in the projection. The y direction indicates the projection lines, or distance from north to south. The x direction indicates the number of samples that from west to east that the sensor recorded for each line. In this example, every pixel encapsulates an area of the earth that is 5 meters across the width by 6 meters in height.

projection.type

The type of coordinate system used to interpret the location of the image as a flat plane. In this example, projection type is Universal Transverse Mercator, as system that divides the earth into 60 zones. If you do not specify a projection.type value, OSSIM defaults to a Cylindrical Equal-Area Projection, as system that divides the earth into straight, equally-spaced vertical meridians and straight unequally-spaced horizontal parallels.

projection.central_merid
projection.origin_latitude

The actual longitude and latitude coordinates of the projection origin. If the keywords for projection origin have no value, OSSIM assumes that the origin is latitude / longitude 0,0, approximately where the earth's central meridian crosses the equator.

projection.datum

A reference point used by the surveyor who created the projection data for this area of the world. In this case, the reference point code, WGE, refers to the World Geodetic System recorded in 1984 (WGS '84).

Figure 5; Sample Session Snapshot

projection.false_easting
projection.false_northing

An offset that is added to the true easting and northing values to create a positive value. For example, assume an image has an northing tie point that is -10 meters from the projection,origin_latitude. The upper boundary of the image is then 10 meters south of the origin_latitude. If that particular projection has a false northing point of 100, then the keyword value for tie_point_northing is derived from the following addition operation:

100+(-10)=80

False_easting and false_northing points are known values. A UTM (Transverse Mercator) projection uses a false easting value of 500,000 feet (151,667 meters) throughout the United States; A Lambert Conformal Conic projection uses a false easting of 2,000,000 feet (606,667 meters) for images taken in the United States. Neither has a false northing because all parts of the US lie north of the equator (in the Northern Hemisphere).

projection.standard.parallel

Coordinate parameters specific to interpreting a Lambert Conformal Conic Projection.

projection.zone

Used to locate a specific UTM projection based in one of the 60 UTM zones.

tie_point_x
tie_point_y

The location of the upper left corner of the image, expressed as a horizontal (x) and vertical (y) offset from the projection origin. In this example, the upper-left corner of the image lies 538185.0000 meters east of the projection.central_meridian and 4208785.0000 meters north of the projection.latitude_origin. Because the false_easting or false_northing keywords are set to 0, the tie_point_x and tie_point_y reflect the true location of the image.

projection.hemisphere

The half-globe the contains the projection. In this example, the projection, and the image taken within the projected zone, lie north of the equator.

projection.scale.factor

Indicates whether the image portrays a full-size area of the earth, or if the image is reduced or enlarged from the actual square meters that bounds the area of the earth recorded.

7. Command-line Applications

The OSSIM open-source libraries include a set of stand-alone applications. The following application programs accompany OSSIM for Phase I:

Name	Function
btoa	Converts a binary header to ascii characters so that a human can read the header information in an image
chgkwval	Changes the keyword value within the keyword list to value specified or adds a new keyword parameter and value pair
citipix2kwl	Converts a citipix parameter file to an OSSIM-compatible keyword list
coordcalc	Converts easting / northing coordinates to the latitude / longitude equivalent of the location
deg2rad	Converts degrees to radians
float2u16	Changes the byte type of an image to unsigned integer. Can also change the number of lines, samples, and bands.
img2rr	Creates a low-resolution overview of any image format
mtrs2ft	Converts meters to feet or the higher precision US survey feet
rad2deg	Converts radians to degrees
swapbytes	Reverses the electronic encoding of an image file so that different computers can read it. Some computers expect an electronic “byte” to have the highest-order value in the left-most position while others read the right-most position as the high-order “bit”

Additionally, OSSIM command-line applications exist that provide information about DTED, CCF, and DEM image file types.

8. GDAL Bridge Support

OSSIM can seamlessly read and write any image format in the Geospacial Data Abstraction Library (GDAL). GDAL Formats on page 4 lists the formats that GDAL supplies in addition to those supported by native OSSIM. The user simply downloads GDAL from the OSRS web site and installs the GDAL reader using the simple instructions, also on the OSRS web site. GDAL is then accessible to OSSIM. The user then opens one of the formats from the OSSIM File|Open dialog box.

In the future, the OSSIM development team might integrate GDAL directly into the OSSIM source so that an end-user does not have to perform the extra download and installation of GDAL. We are considering whether the convenience justifies the expansion of OSSIM in size and installation time.

9. GRASS Functionality

Currently, OSSIM can read GRASS cells as described on 4. With the addition of the GRASS Bridge code, OSSIM users can also perform the following GRASS commands:

Name	Function
i.cca	Canonical components analysis (cca) program for image processing
i.composite	An imagery function that creates a color composite image from three imagery band files specified by the user
i.fft	Fast Fourier Transform (FFT) for image processing
i.ifft	Inverse Fast Fourier Transform (ifft) for image processing
i.pca	Principal components analysis (pca) program for image processing.
i.zc	Zero crossing edge-detection raster function for image processing
r.basins.fill	Generates a raster map layer showing watershed sub-basins
r.buffer	Creates a raster map layer showing buffer zones surrounding cells that contain non-NULL category values
r.clump	Arranges the data in a raster map layer by grouping cells that form physically discrete areas into unique categories
r.coin	Tabulates the mutual occurrence (coincidence) of categories for two raster map layers
r.cost	Outputs a raster map layer showing the cumulative cost of moving between different geographic locations on an input raster map layer whose cell category values represent cost.
r.covar	Outputs a covariance/correlation matrix for user-specified raster map layer(s)
r.cross	Creates a cross product of the category values from multiple raster map layers
r.drain	Traces a flow through an elevation model on a raster map layer
r.grow	Generates an output raster map layer with contiguous areas grown by one cell (pixel)
r.los	Line -of-sight raster analysis program
r.mapcalc	Raster map layer data calculator

r.neighbors	Makes each cell category value a function of the category values assigned to the cells around it, and stores new cell values in an output raster map layer
r.resample	GRASS raster map layer data resampling capability.
r.slope.aspect	Generates raster map layers of slope, aspect, curvatures and partial derivatives from a raster map layer of true elevation values
r.volume	Calculates the volume of data clumps, and (optionally) produces a file containing the calculated centroids of these clumps
r.watershed	Watershed-basin analysis program

B. Open Source Software Experiment

The development time invested in OSSIM proved that our initial estimates of efficiency using open-source techniques to develop code were conservative. Using an open-source methodology reduced development, debugging, and adaptation turn around time from the 45 man-years required to produce a similar commercial product to 6 months of effort from a development team consisting of only 2 full-time, two part-time government funded developers, all aided by the interest of the open-source community.

1. Initial Assumptions

Because many large government and university projects base their remote-sensing libraries on open source code, OSSIM developers were able to borrow those existing, tested libraries as features that you can use within OSSIM.

The OSSIM mailing group points us to any open source library that you feel OSSIM should incorporate. Indeed, members of the group submit their own modules for distribution from our FTP and CVS Repository libraries.

In addition to program code, the open source libraries provide a means to open many available images that other OSRS people accumulate from their remote-sensing hardware or other sources.

2. Software Leverage

- OSSIM incorporates aspects of or interfaces with the following open source packages:
- GDAL - a library that reads several raster geospacial image formats. OSSIM currently reads and writes these formats if GDAL is included on an OSSIM host.
- GRASS GIS (Geographic Resources Analysis Support System) is a Geographical Information System (GIS) with raster, topological vector, image processing, and graphics production functionality.
- LIMP - for intelligent caching and file format plugin architecture

- wxWindows - for all GUI development
- kdem - for Elevation functions
- NIMA Geotrans/PROJ.4 - for datum and projection handling
- NIMAMUSE - for government file format support, mainly MC&G data
- Shapelib - for shapefile support
- Grayscale Image Processing - for basic image manipulation
- Message Passing Interface (MPI)-compliant code that use the Beowulf multiprocessor architecture