Spectral Motion

Effective Analysis and Visualization of Hyperspectral Imagery on a Mobile Device

Master's Project

Department of Computer Science

Montclair State University

Kale Evans

Spring 2015

Advisor: Dr. Robila

Abstract

The proliferation of multispectral and hyperspectral imagery has been quickly increasing as both private companies, public companies, and military organizations continue to launch more satellites into the sky for the purposes of collecting imagery for terrain assessment, mission planning, and material detection. Furthermore, hyperspectral and multispectral sensors are also being used on handheld devices in labs all over the world for the purposes of investigating and advancing the fields of facial recognition, plant physiology, robot and drone research, and many other fields. As the abundance of hyperspectral and multispectral images continues to grow, it seems a logical progression that the flexibility of and the type of applications available on the market should grow as well.

In this report, we document the steps taken, functionality achieved, and the future directions intended with regard to “Spectral Motion”, an iOS based hyperspectral and multispectral Imaging and Analysis software. The goal of this project was to start work on the first mobile hyperspectral data analysis tool, compile multiple hyperspectral data visualization techniques into an easy to use mobile application, discover what limitations exists with developing such a data intensive application on an embedded platform, and to expand the landscape of remote sensing software to the mobile platforms.

Contents

List of Figures:

1. Introduction

The proliferation of Multispectral and Hyperspectral imagery has been quickly increasing as both private companies, public companies, and military organizations continue to launch more satellites into the sky for the purposes of collecting imagery for terrain assessment, mission planning, and material detection. In addition to this, these images are also growing in quantity in sectors that utilize images of a smaller scale such as pharmaceuticals, manufacturing, facial recognition, plant physiology, digital photography, the food industry, and many other markets. Lastly, military units such as the United States Army, the Marine Corps, the US Navy, and the US Air Force greatly utilize hyperspectral and multispectral imagery for navigation and targeting [30].

As the number of multispectral and hyperspectral images continues to rise, it would stand to reason that so should the methods of visualizing and interpreting that data. Consequently tools that allow hyperspectral data visualization should also be available on a multitude of platforms. In this report, we document the steps taken and challenges overcome in the creation of a hyperspectral and multispectral data analysis tool for iOS devices that we call, “Spectral Motion”.

When developing Spectral Motion, some of the technical objectives were (1) to ensure the application required minimal computation time to complete any particular function, (2) to create an application with a base architecture that required minimal memory to ensure the stability and scalability of the application, (3) maintain compatibility with popular existing file formats, hyperspectral sensors, and software applications for maximum interoperability, (4) and very importantly, to craft an easy-to-use, interactive user interface in accordance with Apple’s UI Guidelines and prototypical of the mobile software experience today.

The iOS platform was chosen for this initial venture into hyperspectral mobile applications as it provides ready to use bindings for the C programming language and compilation via LLVM GCC [31]. This was an important point to achieve objectives (1) and (2) above, as C is historically known to be a language that allows lower level manipulation of memory and data, which can lend itself to quicker and more responsive functionality when used effectively. In addition, as Object-Oriented functionality was indispensable to achieve a high level of functionality in a short amount of time, Spectral Motion was also written in C++, and Objective-C programming languages. Many of the image processing methods were accomplished with the use of Intel’s OpenCV framework, which is an open-source C, C++ and Python based image, computer vision, and machine learning software library available under

the BSD license [26]. OpenCV was cross compiled for use on iOS devices for this project. As OpenCV doesn't have direct support for hyperspectral or multispectral data files or an API for data manipulation, most of the C code written for the application served in order to correctly format the data for OpenCV API processing.

At the time of this writing, no publicly available mobile hyperspectral or multispectral data analysis software exists on the Apple App Store, Google Play Store, or online. Therefore, we hope this application can serve as a proof of concept of the feasibility of such an application, as well as open up the possibility for the advantages of field use of such a mobile application in both scientific as well as classroom settings.

This report is organized as follows:

In chapter 2, we briefly describe some common concepts in hyperspectral and multispectral imaging, including the typical format of hyperspectral images, the specifications for well-known hyperspectral sensors, and some popular software packages currently available for analyzing remote sensing images. In chapter 3, we describe the goals of the Spectral Motion application, including the initial system requirements and current specifications. In chapter 4, we describe the system design, the supported data schemes and file types, as well as other supported functionality such as the image display types and plotting. Finally, in chapter 5, we state our conclusions as well as discuss future directions.

2. Common Hyperspectral and Multispectral Technology

2.1. Hyperspectral and Multispectral Images

Spectral imaging is the process of collecting image information across many bands or wavelengths in the electromagnetic spectrum. This image information is very useful with regards to finding objects, identifying materials, or detecting processes in a scene [1]. This being the case, many scientist, engineers, and researchers utilize these images, in concert with spectral libraries, to learn more about the constituent materials in an object [1].

The two main types of images used in this kind of analysis are called hyperspectral Images and multispectral Images. Although there is no strong defining characteristic difference between hyperspectral images and multispectral images, it is generally accepted that multispectral imaging deals with a relatively small amount of images, typically utilizing between 6 to 31 discrete bands across the electromagnetic spectrum, while hyperspectral imaging typically deals with over 100 narrow and contiguous bands across the electromagnetic spectrum [2]. As Spectral Motion looked to accomplish scalability as stated in (2) of the introduction paragraph, all testing and development was geared around the visualization and analysis of hyperspectral images. The higher number of images composite in the hyperspectral data required a more stringent approach for memory constraints and data manipulation. Once the base functionality could be accomplished for hyperspectral data, we found it easily to be the case that the same could be said of the smaller data files produced from multispectral sensors. Thus, solely the term “Hyperspectral Image” will be used for the rest of this report.

Hyperspectral images are most often represented, colloquially, as well as in software applications and research, as a “data cube”, with the x and y axes representing the spatial coordinates of a hyperspectral image, and the z axis representing the bands of the electromagnetic spectrum [3].

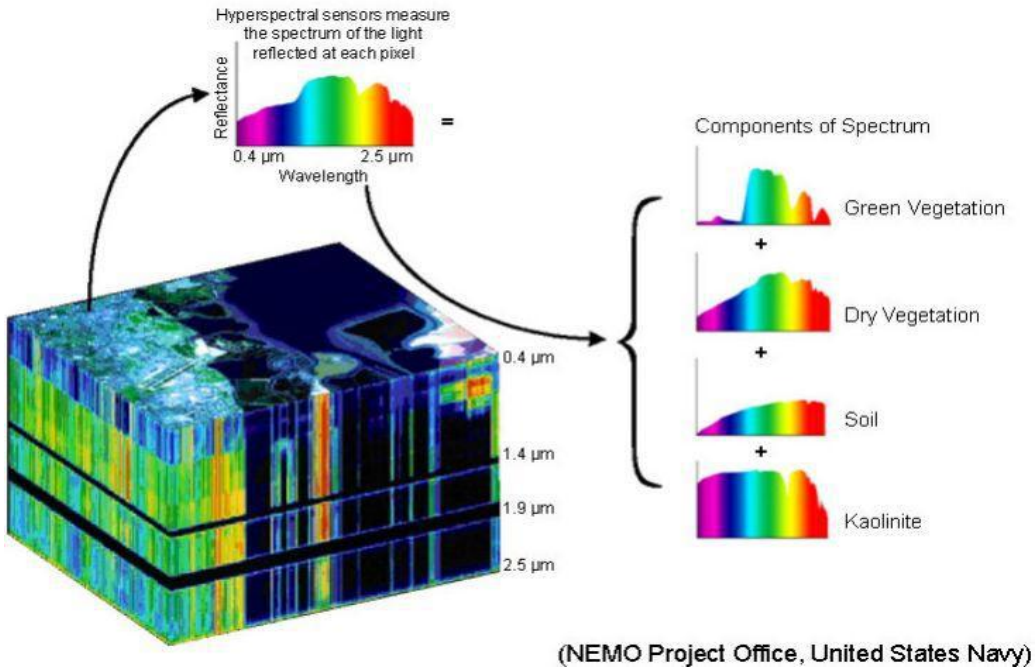


Figure 1. Hyperspectral Data Cube from Aviris Satellite Image[3].

Although hyperspectral image files are simply large binary files, hyperspectral data is often represented as cubes in this way as it makes the processing and indexing of the various electromagnetic bands much easier to manage from a software perspective. Therefore, a huge objective of the Spectral Motion application was to read and process the entire popular binary file formats used in hyperspectral imaging and convert them into 3-D arrays for processing at run-time.

2.2. Hyperspectral and Multispectral Imaging Sensors

At the onset of hyperspectral imaging in the 1970's, a majority of the images being produced were generated by remote sensing satellites and airborne sensors manufactured by NASA, the Navy at The Naval Research Center, the Airforce at the Airforce Research Lab, the USGS (United States Geological Survey), and other military and government organizations [27] [4]. Many of these organizations have published their images and research data and made them publicly available for free in the interest of furthering scientific research in fields that benefit from the analysis of hyperspectral images.

Some of the popular hyperspectral Sensors are:

a. Aviris (Airborne Visible / Infra-Red Imaging Spectrometer.

Aviris is perhaps the most popular hyperspectral imaging sensor in the field of remote sensing. Its images are widely used in scientific research, and its calibration files and satellite information is easily downloadable from the currently maintained Aviris website [5] [4]. Many images produced from Aviris flights between the years 1992 – 2005 were very instrumental in the development of Spectral Motion.

Specs: Aviris has 224 bands ranging from 0.4 to 2.45 micrometers at 10 nm intervals. Aviris is maintained and flown by NASA's Jet Propulsion Laboratory in California.

b. HYDICE (The Hyperspectral Digital Imagery Collection Experiment).

HYDICE was developed by the U.S Naval Research Laboratory (NRL) and operated by Michigan Technical University [6] .

Specs: HYDICE has 210 bands from 0.4 – 2.5 micrometers.

c. Landsat 8.

Landsat 8 is a multispectral satellite, and part of the Landsat program, which is the longest running enterprise for the acquisition of satellite imagery on Earth. It is a joint collaboration between NASA and the USGS [7].

Specs: Landsat has 9 broad bands ranging from 0.4 – 1.39 micrometers.

d. Hyperion.

Hyperion is a hyperspectral sensor mounted on NASA's Earth Observing-1 satellite. One of the main benefits of this sensor is that it has two separate grating image spectrometers for improving signal-to-noise ratio.[32]

Specs: Hyperion provides a high resolution images with 220 bands from 0.4 – 2.5 micrometers.

e. Compact Airborne Spectrograhic Imager (CASI).

CASI was built by Iteres Research in Calgary, and is owned by several private and government agencies [4].

Specs: CASI has a limited wavelength from 0.4 – 0.87 micrometers in 288 bands.

Although the resultant images from these satellites are very popular in the research community, for in-depth development and validation studies many researchers also employ smaller devices in the form of digital cameras or hand-held devices in the laboratory [1].

2.3. Popular Hyperspectral Data Analysis Software

Whereas fifteen years ago remote sensing software tools were only available to the experts of the field, the last decade has seen a tremendous rise in the number of remote sensing applications that have been made publicly available [8]. Many of these application providers seek compatibility with other remote sensing applications with regard to data file, calibration file, and header file format, to aid in making their software easily adaptable and adoptable for the end-user. This cross-compatibility principally, among other things, is accomplished by using similar file extensions, data storage schemes (such as the method of representing the hyperspectral image data), and header formats for easy parsing.

As there is a lack of standardization in the fields of remote sensing resulting from of the variation in sensor types, specifications of those sensors, the investigative purposes of those sensors, and the details of any particular experiment such as atmospheric conditions, it is difficult for there to exist one standard for the representation and description of images. It naturally follows as a result of this experimental landscape that there would be a lack of standard methods of analyzing hyperspectral or multispectral data in data analysis software. However, the following software applications are generally accepted as some of the de facto popular choices with respect to analyzing hyperspectral image data, and consequently have popularized some standard data and text file formats that are adhered to by the Spectral Motion application.

a. ENVI

ENVI is one of the more popular commercial remote sensing applications that, among other things, includes the ability to process hyperspectral data. It also contains spectral libraries which can be used for object detection [11]. The ENVI software is a prime example of an application with which to maintain compatibility with in regards to loading hyperspectral data files, due to its ubiquitous use in the remote sensing community. This being the case, the ENVI hyperspectral data header file format (extension .hdr) was chosen as the principal header format accepted by Spectral Motion.

The ENVI hyperspectral header format is a simple, text-based, key-value format that specifies the minimum information required for reading the hyperspectral data file. Because the listed information in the ENVI Header file tends to include information that is minimally required for data processing, we believe that there is a great chance that this minimum information can be extracted from other header file formats as well. If not directly, it should be the case that other header files can be parsed with minimum changes to the Spectral Motion codebase by accommodating for some of the different verbiage or additional information therein.

Among some other more ancillary pieces of information, the main and most widely used pieces of information contained in an ENVI file header include (1) the number of bands, (2) the number of raster lines per image band, (3) the number of pixels per raster line, (4) how many bits are used per pixel, (5) whether each pixel is in big-endian or little-endian format, (6) the method of storing each pixel or interleave type, and (7) the wavelengths corresponding to each band of the hyperspectral image. While #7 is not required for reading and displaying the hyperspectral data file in a majority of the image display options available in Spectral Motion, it is required for plotting the hyperspectral data for spectral signature analysis. At times, we have found to be true that the wavelength information is not included in the same header file as the accompanying info. In these cases, a possible future option in the Spectral Motion application will be to allow the end-user to select a well-known hyperspectral sensor to auto-populate the wavelengths associated with the hyperspectral image.

A brief abridged version of the ENVI file header is below:

```
ENVI
description = {
    Sensor = Generic}
samples = 614
lines   = 512
bands   = 224
header offset = 0
file type = ENVI Standard
data type = 2
interleave = bip
default bands = {29, 19, 9}
byte order = 1
wavelength = {
0.3698500,
0.3796900,
0.3895300,
0.3993700,
0.4092100,
0.4190600,
0.4289100,
0.4387600,
0.4486100,
0.4584600,
```

Figure 2. Sample ENVI file Header.

b. Erdas

Erdas is another commercial software similar to ENVI, that also allows for the manipulation of hyperspectral data and spectral profiling in an easy to use desktop application. Erdas can read from the tape or CD-ROM generated by a hyperspectral data sensor directly and creates a proprietary file format with the extension “img” that contains the header information from the sensor, as well as the image data created by the sensor. Although the file format of Erdas’ img files is made public online, Spectral Motion does not support the loading of image files

generated by the Erdas application due to time constraints, and looks to do so in a future iteration.

c. Opticks

Opticks is an open-source alternative to commercial remote sensing applications, and is licensed under the GNU Lesser General Public License [12]. Unlike some of the other commercial remote sensing application options, Opticks supports the ability for developers to construct their own plug-ins for Opticks [13]. Opticks is also capable of reading header and image file formats specific to the ENVI software [13], and thus the Spectral Motion application is also partially compatible with Opticks.

d. Multispec

Multispec is another alternative to commercial remote sensing applications. It is developed and maintained at Purdue University in their School of Electrical and Computer Engineering, and is available for free on their website. MultiSpec was funded in part by NASA [33].

3. System Requirements, and Expectations

In order for hyperspectral images to prove helpful, it is important for the visualization of the hyperspectral images to contain adequate contrast and other visual cues to aid the end-user in analyzing the scene. To serve this purpose, one of the initial principal expectations and requirement for the Spectral Motion application was that it display hyperspectral images in a variety of display types, that do not lose intended information, and accentuate desired attributes. The general manifestation of this ability in this application is that the user has the ability to choose the method they want applied to a given hyperspectral image from a list of methods. The adequate visualization of the hyperspectral image data can be determined by the end user, based on the ability of the user to effectively analyze the scene, which is generated in real time based on their choice of method.

Some of the imaging techniques used to analyze hyperspectral images can be very computationally expensive both in terms of processing power as well as memory requirements. This being the case, we decided to focus on some of the basic and most useful imaging techniques that could be implemented using the somewhat limited processing power of iOS devices initially. The first basic image type considered was grayscale. It was considered important for the user to have the ability to simply map a particular spectral band to a grayscale image, and to easily view contiguous bands in this way, in order to give the user a quick and useful snapshot of the scene at particular bands. In the same fashion, true color and false color RGB images were seen as integral as well, given the ease of implementation, as well as the utility of the ability to choose a particular channel for a given wavelength. This is a strong asset to a user as it allows them to use map desired colors to wavelengths that they wish to investigate.

Then it was also important for the visualization techniques to filter out undesired aspects of the image, and accentuate the parts of the image that convey the most information. To that end, Principle Component Analysis was viewed as an important technique for both its wide use, as well as its ability to extract the variability in the large amount of image bands for user investigation in a smaller amount of images, thus adding to the user's ability to effectively visualize a scene. In addition to this, the ability to effectively visualize vegetation in a scene is very useful in parsing the boundaries in an image [1]. This being the case, NDVI was also determined to be an effective visualization technique that would easily allow the end-user to determine where regions of vegetation exist in the image.

The other large requirement for Spectral Motion was the ability to introduce other visual cues that could aid the end-user in determining the end members included in a scene. Although a number of ways exist to accomplish this task, perhaps the most visually acceptable method is the ability to plot spectra on a graph in order to view its reflectance values. Therefore, Spectral Motion has the ability to plot a given pixel's reflectance values across the hyperspectral image's bands when the user taps on a chosen portion of the image. This ability,

paired with the future ability to plot reflectance values of other known materials will further aid the end-user in effectively studying the scene.

Lastly, effective visualization of hyperspectral data also includes the ease-of-use of the application itself, therefore, another huge design requirement for Spectral Motion was effective UI design, and an intuitive interface for the user to explore the image. Part of accomplishing this goal was utilizing some of the best features of the iOS platform, including multi-touch gestures. This allows the user to easily move and zoom into the image in order to view it effectively. It also allows the user to easily shrink the image in order to increase space on the screen for other images or graphs.

Hyperspectral imaging software is often filled with many visualization techniques and target detection methods. It was a general expectation of course that Spectral Motion could not accomplish a majority of this functionality during the time span of the project. However, the general expectation and scope of the project was to cover the basic ideas of hyperspectral image visualization, including the common image display methods as well as establish some way of investigating end-members in a scene. The scope of the project, of course, also included creating a visually pleasing and easy to use application. We believe that the imaging and plotting capabilities inherent in Spectral Motion accomplish the first goals, and that the adherence to mobile development UI paradigms during the development of the application creates a good base user experience for end-users interested in viewing hyperspectral data, thus accomplishing our design goals.

4. System design and structure

4.1. Overall Structure

Spectral Motion is an application that allows the user to visualize hyperspectral image data as a typical rgb and grayscale image, as well as a graph of plotted reflectance values from the selected image. The workflow for the Spectral Motion application is as follows:

1. First, when the end user opens the application, they can select either a sample image included in the application, or an image that the user previously imported via Dropbox and stored into the application directory. Upon selecting the image, Spectral Motion parses the header text file included with the hyperspectral binary file for the pertinent information required to load the hyperspectral data into memory such as number of lines, rows, and bands included in the image.

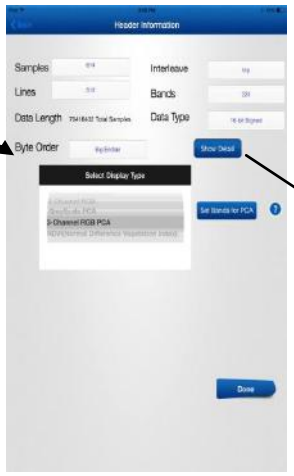
2. Once the header file is parsed, the end-user has the ability to view the parsed information in the "Header Information" viewer, and modify it as necessary. At this stage, the user must also select their desired visualization technique out of the list of five techniques. If required, the user must also add some additional information for the visualization technique (ex. Choosing the desired bands to submit for principal component analysis). After the user is satisfied they have chosen the correct parameters, they can submit their request.

3. Spectral Motion loads the hyperspectral image file into memory and also applies the technique with accompanying parameters as determined by the user in step #2. The user is then brought to the "ImageViewer" which is a view encompassing the entire coordinate space of tablet screen for viewing the current hyperspectral image. In the "ImageViewer" the user can visualize the image as well as move, zoom, or shrink the image as necessary.

4. If desired, the user can long-press the image produced from step #3, and select the "Display Graph" option from the context menu to view the data plot for the hyperspectral image. To view the plot for different pixel locations, the user can tap on the image to regenerate the plot corresponding to that spatial location in the image.

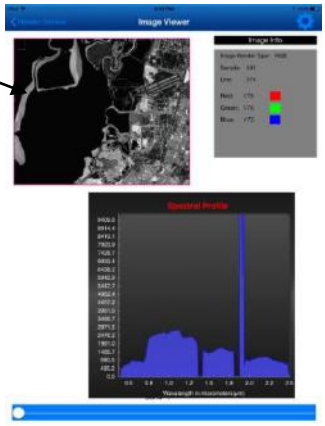
5. As desired, the user can remove existing renderings, and add additional renderings of the chosen hyperspectral image in order to view and compare images with different image qualities side by side.

#1. Main Screen



#2. Header Viewer

#3. Image Viewer



#4 Graph View

4.2. Code Structure

Spectral Motion was implemented in C, C++, and Objective-C programming languages. The C programming language was chosen for partial implementation of the project principally because of the speed in computing time resulting from the lack object-orientation and other higher level software layers, because it has very well supported functionality with regard to reading and parsing files, and because of its ease of access to raw data pointers, in this case, the hyperspectral binary data. C++ was required and used implicitly with use of the OpenCV framework for managing and manipulating the image data after its loading into the appropriate format. The Objective-C programming language was used principally for managing application specific components such as the GUI.

Central to Spectral Motion's architecture is the c structure HDRINFO and the class MSHyperspectralData implemented in C, C++, and Objective-C. The HDRINFO structure is used to hold the pertinent information parsed from the hyperspectral header file in ENVI format upon the user's selection of a hyperspectral image, and is implemented as below:

```
typedef struct HDRINFO
{
    int samples;//columns
    int lines;// rows
    int bands;
    int header_Offset;
    int dataType;
    int *defaultBands;
    int byteOrder;
    float *wavelength;

    const char *interleave;
    const char *fileType;

}HDRINFO;
```

Figure 19. HDRINFO structure.

The class MSENVIFileParser is used solely to parse a given hyperspectral header file and create a HDRINFO object to encapsulate the parsed information. Upon the user's selection of the image file for viewing, the MSENVIFileParser class loads and parses the ENVI file associated with the user selection, sets the HDRINFO structure's members, and returns a copy to a newly allocated MSHyperspectralData object, which stores the copy and maintains it until the MSHyperspectralData object is released.

The MSHyperspectralData class encapsulates an instance of hyperspectral data including the hyperspectral binary information represented as a 3-D array, and the metadata associated with the header file thru the HDRINFO structure. The MSHyperspectralData class is also responsible for processing the hyperspectral data using various, user chosen methods for rendering the hyperspectral image to screen. At this time, only one instance of a MSHyperspectralData object is maintained globally in the application at a given time given the large amount of data associated with it. Thus, a pointer to the global MSHyperspectralData

object can be passed down to various viewer classes, which can call image processing methods directly on the MSHyperspectralData object, which will return a platform specific image format for rendering to screen.

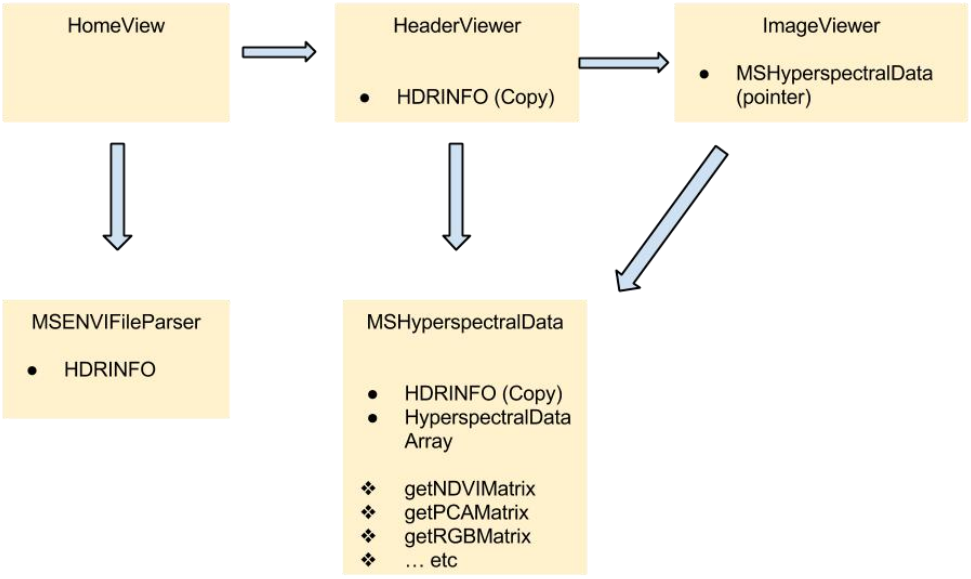


Figure 20. HDRINFO structure.

4.3. Hyperspectral Image Formats

Hyperspectral images are most often stored in 3 different raster schemes; BIL, BIP, and BSQ format [14]. The raster format used is largely dependent on the hyperspectral sensor type, and the sensor manufacturer makes this information available to software providers via the sensor's header file. Spectral Motion supports all 3 raster formats, and modifies its pixel extraction method based on the information retrieved from the image header file by setting a function pointer at run time.

BIL (Band Interleaved by Line) - BIL stores pixel information band by band for each row or column of the image [14]. This essentially means all of the information for a band is compiled in a given row or column. For example, when reading a hyperspectral data file with n rows, b bands, and $n * b$ total columns, one can form one grayscale image from the first hyperspectral band by reading from column 1 to n and row 1 to n.

Let B be the total number of bands, S be the number of pixels per image, and A be the number of bits required to represent each individual pixel. Then, to get the index of a pointer to the single pixel at row y, column x, and band z of a multi-band image, where the indexing starts at 0, one can use the function

$$(\boldsymbol{y} * \boldsymbol{B} + \boldsymbol{z} * \boldsymbol{x}) * \boldsymbol{S} * \boldsymbol{A} \quad (1)$$

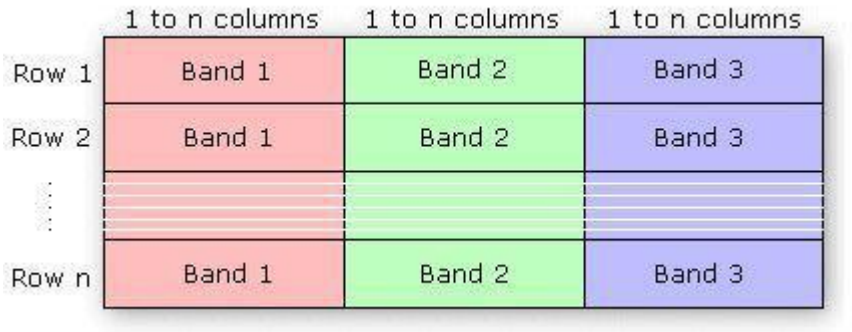


Figure 3. BIL raster format diagram[14].

BIL pixel index function is implemented in Spectral Motion as below:

```
int getBILPixelIndex(int x, int y, int z, int width, int height, int depth)
{
    //(row*bands+band*column)*samples*allocation
    //return ((   (y * depth) + z * width   ) * width );
return ((   (y * depth) + z * x   ) * width );

}
```

Figure 4. Spectral Motion BIL function.

BIP (Band Interleaved by Pixel) – BIP stores pixel data in more of the traditional image scheme, with the bands for each pixel written adjacent to each other, and each pixel also written adjacent to each other [14]. Then, to get the index of a pointer to the single pixel at row y, column x, and band z of a multi-band image, where the indexing starts at 0 one can use the function

$$((\boldsymbol{y} * \boldsymbol{S} + \boldsymbol{x}) + \boldsymbol{x}) * \boldsymbol{B} + \boldsymbol{z} \quad (2)$$

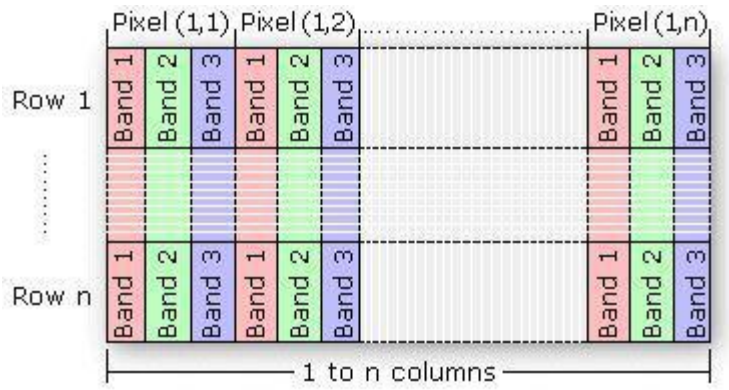


Figure 5. BIP raster format diagram[14].

BIP pixel index function is implemented in Spectral Motion as below:

```
int getBIPPixelIndex(int x, int y, int z, int width, int height, int depth)
{
    //(row*samples + column)*allocation*bands+band
    return ((   (y * width) + x   ) * depth + z);
}
```

Figure 6. Spectral Motion BIP funtion[14].

BSQ (Band Sequential) – BSQ stores information for the image one full band at a time. This means all pixels in band 1 are stored first, then all pixels in band 2, and so on [14]. Let R denote the total number of rows for each band image. Then, to get the index of a pointer to the single pixel at row y, column x, and band z of a multi-band image, where the indexing starts at 0 one can use the function

$$\boldsymbol{R} * \boldsymbol{z} * \boldsymbol{S} + \boldsymbol{y} * \boldsymbol{S} + \boldsymbol{x} \qquad (3)$$

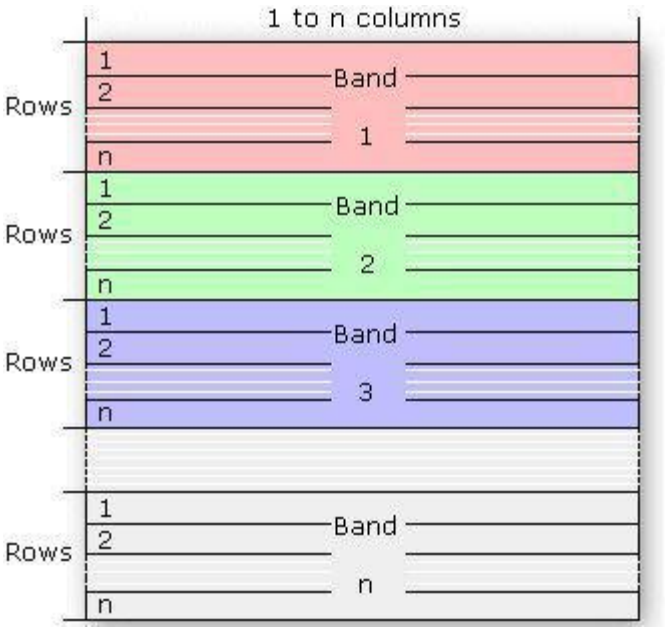


Figure 7. BSQ raster format diagram[14].

BSQ pixel index function is implemented in Spectral Motion as below:

```
int getBSQPixelIndex(int x, int y, int z, int width, int height, int depth)
{
    // (lines*band*samples+row*samples + column)*allocation
    return (height * z * width + (y * width) + x);
}
```

Figure 17. Spectral Motion BSQ function[14].

The explicit image file extensions supported by Spectral Motion include .bip, .rfl, .bil, and .bsq. Spectral motion can easily be modified to support all file extensions in a future iteration, as long as the image data scheme follows one of the 3 formats mentioned above.

4.4. Regular Expression Module and Supported Document Formats

Spectral Motion supports parsing of the ENVI header file format, which stores information pertinent to reading and processing the corresponding hyperspectral binary file. Because the ENVI header file, similarly to other hyperspectral data header files, is not entirely standardized, and allows for the addition or removal of some image metadata, it is necessary to implement a regular expression module to parse the file, with allowances for the presence or absence of some information.

The Spectral Motion ENVI file parser parses minimally for the (1) number of lines per image, (2) the number of pixels per image, (3) the total number of bands in the hyperspectral data file, (4) the header file offset, (5) the data type of each pixel, (6) the byte order of each pixel, (7) the file type of the header file, (8) the pixel interleave type, (9) the default bands of the image, and (10) the list of wavelengths for the image.

The possible values for #5 are listed in the table below:

1	8-bit unsigned integer
2	16-bit signed integer
3	32-bit signed integer
4	32-bit real
5	64-bit real
12	16-bit unsigned integer
13	32-bit unsigned integer

Figure 8. ENVI Header data types [15]

16-bit signed integers are used in many popular hyperspectral sensors such as Aviris [17] and HYDICE [16]. The image processing and parsing methods in Spectral Motion are determined at runtime via function pointers.

The byte order is required for verifying that the image data is in the proper format for the host mobile device. iOS, as well as most devices with ARM CPUs use little-endian byte order by default, so the data must be converted accordingly for the image to render appropriately.

```
338 -(void)convertWordToLittleEndian:(int16_t&)word
339 {
340     /* below washes image out, and there is not enough detail
341     if(word < 0)
342     {
343         word = 0;
344         return;
345     }
346      */
347
348     word = ( (word & 0x00FF) << 8 ) | ( (word & 0xFF00) >> 8 );
349
350 }
```

Figure 9. Spectral Motion function converting Word to Little-Endian

The file type of the header, the default bands on the image, and the list of wavelengths of the image are not required for loading the image, however, the list of default bands is used for

preselecting the image bands used to generate a true color image, and the list of wavelengths used for the image is used for analyzing the spectral signature of pixels in the image.

The regular expressions used for parsing some of the options for an ENVI header file are below:

```
12
13  #define LINE_SIZE_REGEXP @"lines\\s*=\\s*[0-9]{1,5}\\s*\n"
14  #define SAMPLE_SIZE_REGEXP @"samples\\s*=\\s*[0-9]{1,5}\\s*\n"
15  #define BAND_SIZE_REGEXP @"bands\\s*=\\s*[0-9]{1,5}\\s*\n"
16  #define HEADER_OFFSET_REGEXP @"header\\s*offset\\s*=\\s*[0-9]{1,5}\\s*\n"
17  #define DATA_TYPE_REGEXP @"data\\s*type\\s*=\\s*[0-9]{1,5}\\s*\n"
18  #define BYTE_ORDER_REGEXP @"byte\\s*order\\s*=\\s*[0-9]{1,5}\\s*\n"
19
```

Figure 10. Spectral Motion Regular Expressions

Spectral Motion also supports reading from Aster Spectral Library files. The Aster Spectral Library is a compilation of over 2400 spectra of natural and man-made materials. The Aster Spectral Library includes data from three other spectral Libraries at Johns Hopkins University, the Jet Propulsion Laboratory Spectral Library, and the United States Geological Survey Spectral Library [18]. They are single text files that contain information about the class and type of material documented in the file, some information regarding the origin of the data, and the recorded spectral signature of the given material for each specified wavelength, typically as reflectance percentage values of the radiant energy incident with the material during the study. Spectral Motion includes Aster Spectral Library files for the following classes and types of materials:

- Vegetation
    - Conifer
    - Decidous
    - Dry Grass
    - Grass
- Man-Made
    - Construction Asphalt
    - Construction Concrete
    - Construction Tar
    - White Marble
    - Black Paint
- Soil

    - Red and Orange Sandy Loam
- Rocks
    - Granite
    - Marble

4.5. Available Image Display Types

One of the more integral features of Spectral Motion, and most central to accessing the various functionality and analysis features of the application is the ability to generate human viewable images. Spectral Motion supports 5 preliminary image display options.

i. Grayscale - With this method, the user can select one of the hyperspectral image bands to render into a grayscale image. If there is a list of default bands found in the header file, the auto-filled in choice is the default green band. With grayscale images, the end-user can also interact with a slider that will change the mapped band for the grayscale image, and render it in real time.

Figure 11. Grayscale image generated by Spectral Motion

ii. True Color/ False Color Image – With this method, the user can select three bands from the hyperspectral image to map into the RGB color space for rendering. If the default bands are located in the header file, the auto-filled in choices are the three default bands listed in the file and Spectral Motion will produce a true color image. Otherwise, the user can select to produce a false color image.

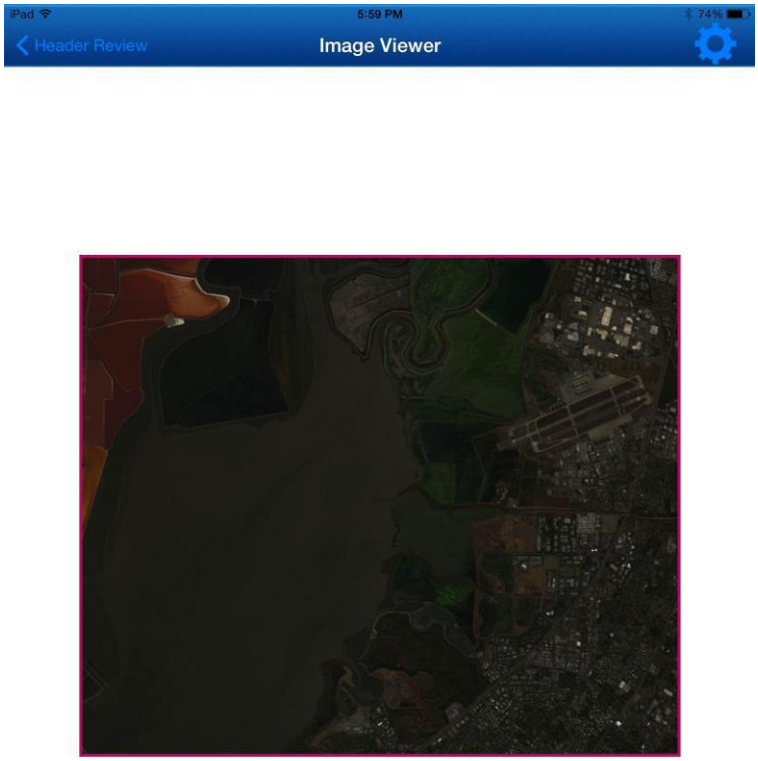


Figure 12. True Color image generated by Spectral Motion

iii. Principal Component Analysis (PCA) – Principal Component Analysis is a very well adopted method used in all fields of image and video processing, that converts a set of observations of possibly correlated variables into linearly uncorrelated variables called principal components [20]. These principal components are not only highly uncorrelated in most cases, but as a result contain the most varied information in a smaller subset of the original images. The principal components are normally arranged from the component with the most variance to the component with the least variance.

The general objective is to take a group images, and extract the most varied information from them, so that the 1st principal component has the most variance, and in turn the highest contrast, the 2nd principal component has the 2nd most variance, and so on. This is a great and indispensable tool when used for the compression of images and data, as a

majority of the information can be exhibited in a smaller data space. For this same reason, it is also popular in hyperspectral imaging as it has the potential to reduce the dimensionality of the original image data set, while maintaining and conveying the most important information.

The steps required for generating the Principal Components of a hyperspectral image in the Spectral Motion application were as follows:

Step 1: Let r be the number of lines in a given hyperspectral image, let c be the number of pixels in a given hyperspectral image, let b be the number of bands in the hyperspectral image, and let the number of pixels in a single band i be defined by $i = r * c$.

Represent each image band of the hyperspectral image as a single row vector and compile them so that you generate a $b\ x\ i$ matrix. The column vectors that make up this matrix represent our random variables.

```
for(int bandIdx = 0; bandIdx < arraySize; bandIdx++)
{
    for(int pixelInBand = 0; pixelInBand < nPixelsInBand; pixelInBand++)
    {
        rowIdx = pixelInBand/samples;
        columnIdx = pixelInBand % samples;

        prePCAMatrix.at<uint16_t>(bandIdx,pixelInBand) = ((uint16_t***)m_HyperspectralCube)[rowIdx][columnIdx][bandArray[bandIdx]];
    }
}
```

Figure 18. Spectral Motion compiling 16-bit image data for PCA processing

Step 2: Calculate the mean vector for each image vector compiled from Step 1. Let's call this mean vector m_x.

Step 3: Calculate the covariance matrix C_x which is defined as

$$\boldsymbol{C_x = E\{(x - m_x)(x - m_x)^T\}} \qquad (4)$$

where the E operator denotes the expected value.

Step 4: Calculate the eigenvectors and eigenvalues of C_x. C_x is a real and symmetric matrix given the fact it is the product of a given matrix and its transpose, therefore, the eigenvectors and eigenvalues can always be calculated. Once these values have been calculated, arrange the eigenvectors into a matrix A such that the 1st row corresponds to the eigenvector with the largest eigenvalue, the 2nd row corresponds to the eigenvector with the 2nd largest eigenvalue, and so on.

Step 5: Project the matrix A onto the difference matrices $x_i - m_x$ for every vector x_i from our original image to generate a group of vector $y's$. This projection is referred to as *the Hotelling transform* as below:

$$\boldsymbol{y = A(x - m_x)} \quad (5)$$

These vectors y, represent the principal components of the original image.The following are display options are available for PCA:

- Grayscale PCA
- False Color PCA

Grayscale PCA

Instead of using all of the eigenvectors in C_x to form matrix A, it is general practice to choose a subset k of them instead, which will generate k dimensional vector $y's$ which represent the linearly uncorrelated image vectors. In Spectral Motion, for generating a grayscale PCA image, the end user gets to decide on a subset of the hyperspectral bands to submit for PCA image generation. Spectral Motion chooses $k = 1$, and maps the returned 1st principal component directly to a grayscale image. At this time, when testing with an iPad 3rd generation tablet running a 1 GHz dual-core ARM Cortex-A9 and 1024 MB LPDDR2 RAM, it appears processing over approximately 150 bands for PCA analysis causes memory consumption and stability issues in the application.

Ideally, it would be best to run PCA analysis over the entire image space, therefore, future iterations of Spectral Motion will have to consider how to manage some of the large processing requirements from algorithms such as PCA. Possible solutions include finding a clever way to break up and store the result of individual calculations (perhaps maybe even on disk) so as to not use up so much memory for a given function. Another solution would be to pre-process the hyperspectral data so that only the images with highest contrast or most information will be submitted for PCA. This is a potentially good solution as many bands in a hyperspectral image sometimes don't convey any information at all due to atmospheric absorption. Another potential solution, albeit less optimal solution, might be to select a random subset of the hyperspectral images for processing.

Figure 13. PCA Grayscale image generated by Spectral Motion

False Color PCA

Similarly to the method exhibited in Section 3.1, Spectral Motion also allows the end-user to select a subset of the bands and have them mapped individually to RGB channels for image rendering. The end-user first selects particular bands to be submitted for PCA, and then chooses which RGB channel they'd like that band mapped into after PCA processing has completed. For example, an end user can choose a group 1 of bands 1, 5, and 10, a group 2 of bands 2, 6, and 10, and a group 3 of bands 3, 4, 15 for PCA analysis. The end user can also choose that group 1 maps to the red channel, group 2 maps to the blue channel, and group 3 maps the green channel after PCA processing has completed for each individual group.

In the interest of maintaining relatively quick processing times, each group that the user selects for PCA is submitted into its own background thread for calculation. This being the case, the time it takes to complete PCA processing for False Color PCA image generation is equal to the time it takes the largest group to process.

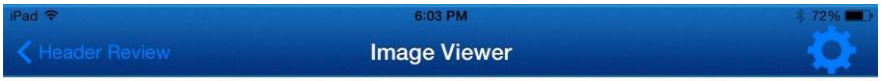


Figure 14. PCA False color image generated by Spectral Motion

For future iterations, instead of having the user choose 3 separate groups to map to the RGB channels for False Color PCA image generation, Spectral Motion can instead allow the end-user to select one group of channels to submit for PCA analysis(similarly to grayscale PCA) and instead map the 1st, 2nd, and 3rd resultant principal components to the red, green, and blue channels respectively. It can also be the case that the end-user decides what channels to map respective principal components to. As each principal component looses more image information and contrast, it was decided that the above currently implemented false color PCA method would generate higher contrast images for initial demonstration, and was chosen as a priority implementation for the first iteration of Spectral Motion.

iv. Normalize Difference Vegetation Index (NDVI)

The NDVI is another method to generate grayscale images from hyperspectral data that is intended to indicate whether or not the target material contains live green vegetation or not [21]. It is possible to indicate vegetation graphically in a grayscale image due to the nature of green vegetation and how it reacts to radiation. Green plants absorb and utilize radiation for photosynthesis. However, live green plants have evolved to reflect radiation in the near-infrared spectrum as this radiation has insufficient energy for plants to use in the photosynthesis process [21]. Because plants reflect radiation in the near-infrared spectrum, they appear brighter in the near-infrared spectrum and darker in others. Thus, the below ratio can be calculated on a pixel by pixel basis in order to generate grayscale images that tend to be brighter where vegetation exists, and darker where it does not.

$$\boldsymbol{NDVI} = \frac{(\boldsymbol{NIR}-\boldsymbol{VIS})}{(\boldsymbol{NIR}+\boldsymbol{VIS})} \text{ [21]} \qquad (6)$$

NIR and VIS represent the near-infrared and visible light in the red spectrum respectively. Although the exact wavelength range for NIR and VIS are not absolutely definitive, Spectral Motion uses a range of 0.62 – 0.75 and 0.75 – 1.4 micrometers for VIS and NIR respectively. Under the assumption that at least a majority of hyperspectral images within specified wavelength ranges are highly correlated, Spectral Motion implements the NDVI calculation by selecting two bands in the median ranges of NIR and VIS, and implements equation (6) on a pixel by pixel basis. The resultant image, as expected, displays higher grayscale values in segments with suspected vegetation, and lower grayscale values otherwise.

Figure 17. NDVI image generated by Spectral Motion

In order for Spectral Motion to generate the correct NDVI images, it must be ensured that the header file lists the wavelengths. At this time, if no wavelength values are found in the header file, Spectral Motion will default to the Aviris wavelength values published in 2005 by the NASA JPL. In future iterations, the wavelength values for many popular sensors can be included, and chosen by the end-user.

4.6. Spectral Signature

One of the principal, if not the principal purpose and function of hyperspectral imaging is target identification. This is accomplished via the Spectral Signature of an object, which is the reflected electromagnetic radiation at a particular wavelength for that object [22]. The spectral signature can be a very strong indicator for an object's identification as even reasonably similar

materials can exhibit distinguishably different spectral plots. Given the fact that spectral signatures are a mainstay in hyperspectral imaging, Spectral Motion has started to integrate spectral analysis visualization techniques and libraries into its architecture.

By long pressing any generated image in Spectral Motion, the user is presented with the option to display the corresponding line plot for the hyperspectral data. This user action adds a graph view to the screen that will display both the RGB pixel values in a separate panel, as well as generates the plot for the pixel locations in real time as the user taps on chosen locations in the image. With the added ability of magnifying the image via iOS' multi-touch technology, it is very easy for the end-user to select pixel location of interest to investigate its spectral plot.

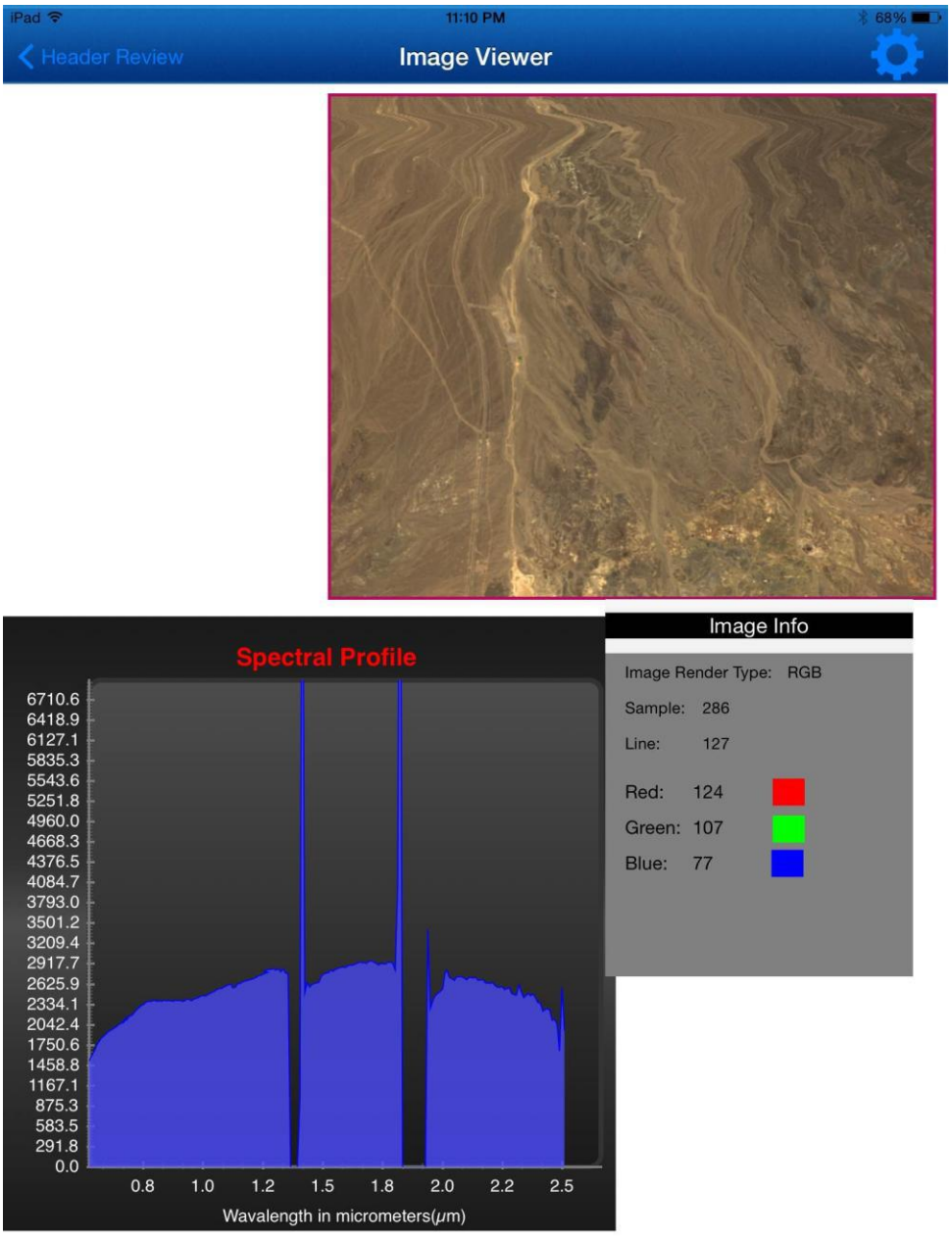


Figure 15. Image of Spectral Plot generate by Spectral Motion of Aviris Cuprite Field

As the pixel values of hyperspectral data can sometimes vary greatly across image bands, Spectral Motion focuses and adjusts the y axis values based on the 1st standard deviation of the pixel values by default. However, the user still has the ability to zoom and pan inside the graph view to investigate the plot in more detail. The plotting capability of Spectral Motion is provided by an open source library made available under the BSD License named Core Plot. Core Plot is a plotting framework for OS X and iOS created by Eric Skroch. It provides 2D visualization of data, and is tightly integrated with Apple technologies like Core Animation, Core Data, and Cocoa Bindings [23].

Spectral Motion has also taken steps to plotting the spectral signatures of other materials on the same axis as that of the image. As stated in Section 4 of this report, Spectral Motion has the Spectral Libraries for a select number of Vegetation, Man-Made, Rock, and Soil materials. The GUI allows for the user to select one or more spectral signatures from a list as choices for plotting alongside the hyperspectral image's plotted profile. It is a work in progress at this time to implement the actual plotting feature.

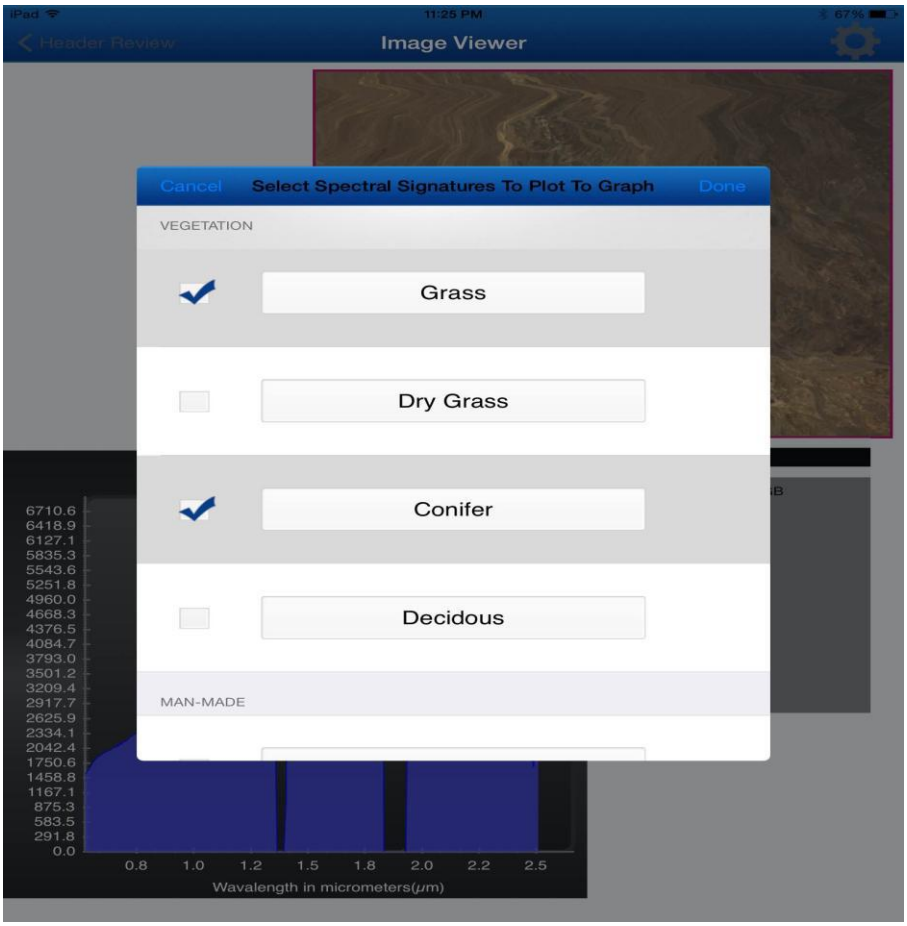


Figure 16. Spectral Motion Library selection table.

An upcoming future direction for Spectral Motion is the incorporation of the ability to easily determine the target material present in hyperspectral images. As there is a huge abundance of spectral libraries and library files available for use, and it is not computationally economical to include all of the libraries inside of the app, a future implementation will allow the user to choose and download chosen spectral library files to store in the app. In addition to this, Spectral Motion looks to aid in real time target detection via popular statistical analysis techniques such as SAM(Spectral Angle Mapper) which is an algorithm for determining the spectral similarity between two spectra by calculating the angle between the spectra [24].

One challenge with determining the target based on the spectra generated from a hyperspectral image is that the data has to be calibrated before it can be useful for spectra comparison [25]. Calibration is done by removing any interfering effects of atmospheric absorption and scattering, as well as effects from the solar spectrum and instrument bias or error [25]. The end result of this calibration is normally reflectance units as a percentage of light incident with the object, and is obtained from multiplication of gain values or similar with the original radiance pixel data. This kind of calibration is impossible to do without the various environmental information obtained from the flight. Unfortunately, this calibration information is not standardized at all, and is not easily found for some hyperspectral images. Thus no uniform technique exists that can perfectly calibrate given hyperspectral data files. This being the case, there are a few choices for obtaining valid spectral information for target detection.

1. Some hyperspectral files luckily are already converted into reflectance percentage units and are ready for immediate spectra comparison. Normally, it can be determined if properly calibrated reflectance units are used in the image by the header file, or sometimes even by the hyperspectral data file which will use extensions such as .rfl to denote they are using reflectance values as opposed to radiance or digital number values.

2. The most ideal option would be to calibrate hyperspectral data without the dependence on additional external header file information. This is known as scene-derived calibration. One simple method that can be easily implemented for non-calibrated hyperspectral data files is to divide each pixel radiance value by the images mean pixel value to estimate reflectance. This approach is known as the Internal Average Reflectance (IAR) approach [25]. Another similar method is the flat field correction approach, which looks for an area in the scene that shows minimal variation in pixel values across wavelengths, and averages the radiance values in this field to be used as a correction factor for the rest of the pixel values in the image [25]. Lastly, there are also physics based models that rely on some assumptions made about absorption by atmospheric gases. However, physics based models tend to be moderately complex. Many different methods of varying degrees of success continue to be developed every day, and Spectral Motion intends to integrate some of the more effective ones in time.

4.7. Downloading Hyperspectral Data to Spectral Motion

Although Spectral Motion includes some sample hyperspectral image files for visualization and analysis inside the app, in order for the user to view additional hyperspectral image files inside the app, the user must import them from an external server. Many popular application vendors provide a free service to end-users that give them the ability to store files

on a remote server and download them as necessary to the desired device. Some popular applications that provide this service include Google Drive, Dropbox, and OneDrive. Spectral Motion supports downloading hyperspectral image and header files from the user's Dropbox account for visualization and analysis.

First, the user must possess a free Dropbox account where they have placed hyperspectral images and header files that are compatible with the Spectral Motion application. DropBox account creation and management can be managed from www.Dropbox.com. Once this is the case, the end-user can import them from first view in the Spectral Motion application by selecting the "Import New Image" button near the top of the table of images. The end-user then authenticates their account in the Spectral Motion app, and subsequently selects the image files and the accompanying header files for serialization in the iOS device's file system. Once the user selects the image, Spectral Motion will begin downloading the image and header file in a background process. Once the download is complete, the user can select the image from the table of hyperspectral images.

At application start-up, Spectral Motion checks the file system, and if any previously downloaded hyperspectral image and header files are located therein, it will allow the user to select them without having to download them from Dropbox again.

Dropbox was included in the initial implementation of Spectral Motion given its popularity and overall adoption in the market. However, there are also many other popular remote storage solutions that can be considered and implemented into Spectral Motion in future iterations, that will also allow the end-user to import image files, header files, spectral library files, and any other files that will help in the analysis of the image.

5. Conclusions and Future Directions

Work on the Spectral Motion application proved very fruitful as some of the base functionality expected from a hyperspectral imaging software were successfully implemented on a mobile platform as desired. At the same time, it is evident how much more work can be completed in order to grow Spectral Motion into a truly useful application for researchers and students, as well as perhaps even a competitive product to some of the more popular remote sensing applications. Overall, we've taken some meaningful first steps in the creation of a mobile hyperspectral imagery software, perhaps one of, if not the first, one of its kind.

There are many places to improve Spectral Motion's potential effectiveness as a mobile hyperspectral data visualization and analysis engine. One obvious place for advancement is of course compatibility with more hyperspectral date file types (Erdas .img files is a good first candidate) and the addition of more visualization techniques. Two tried-and-true and popular hyperspectral data visualization techniques include Jacobson and Gupta's fixed linear spectral weighting envelopes and artificial neural networks [28]. In addition to this, many new visualization techniques are developed every day, and worth inclusion in Spectral Motion if they lend themselves to interesting and uniquely different ways of visualizing the image.

Another place for advancement is in the effective comparison of image spectra with that from established spectral libraries such as the Aster Spectral Library. A future and important future direction will be to include a generous subset of the files from a spectral library for spectra comparison. For those library files not included in Spectral Motion by default, another future direction will be the allowing the end-user to download them from an online server. Dropbox and Google Drive are popular choice for this route, however, a dedicated Spectral Motion server would prove useful with regard to organizing data and adding sample files for the user to easily download.

In accordance with comparing image spectra, methods for calibrating uncorrected hyperspectral data needs to be implemented in Spectral Motion to allow for the effective cross-comparison of spectra. Ideally, Spectral Motion would principally include scene-derived methods of calibration to remove the need for the end-user to supply experimental metadata. In addition, methods for statistically comparing two or more spectra would need to be included as well for complete spectral comparison functionality. One popular method is SAM(Spectral Angle Mapper) which compares the angle of plotted spectra, and determines the likelihood of a match between pairs of spectra by the magnitude of the angle. Other popular supervised classification techniques include minimum distance and maximum likelihood which are probabilistic and Euclidean distance based methods respectively. Supervised classification

techniques such as SAM, minimum distance, and maximum likelihood require training sample data or external libraries, but some do not. Spectral Motion will also look to support Unsupervised Classification techniques, which creates “clusters” based on statistical analysis of the image data only [29].

One desired future direction is the use of Spectral Motion in the field with a handheld or drone mounted hyperspectral or multispectral sensor. One of the main use cases for a mobile device in general is the use of the software in settings where access to a full desktop or laptop is not possible or practical. This being the case, it was determined Spectral Motion had the potential for use perhaps outside of the laboratory setting, and with sensors that could easily be carried and/or controlled by the user. Image data could be transferred via Bluetooth, USB, or an adhoc WiFi connection. This possibility could open up the flexibility of visualizing and interpreting hyperspectral image data, and even open up the field to more scientists, professors, and students who may not have access to the more expensive sensors and software licenses for visualization software. In addition to this, Spectral Motion also has great potential for use in classroom settings, given it’s a tablet-based, easy to use software application.

References:

[1] “Hyperspectral imaging“.[Online]. Available http://en.wikipedia.org/wiki/Hyperspectral_imaging Accessed February 4th, 2015

[2] Petteri Teikari. (2008). “Multispectral Imaging. Course Project for AS-75.2128 Imaging And Display Technology.” [Online]. Available http://www.petteri-teikari.com/pdf/Teikari_Multispectral_Imaging.pdf Accessed February 7th, 2015

[3]Clayton Blodgett. “What is Hyperspectral Imaging (HSI)” [Online]. Available http://www.cerc.usgs.gov/morap/Assets/UploadedFiles/Projects/Contamination_Characterization_through_Airborne_Hyperspectral_Imaging/What_Is_Hyperspectral_Imaging.pdf Accessed February 7th, 2015

[4] Zachary Bortolot. “Hyperspectral Remote Sensing”. [Online] Available http://ibis.geog.ubc.ca/courses/geog570/notes/hyperspectral.html Accessed May 1st, 2015

[5]. Aviris Airborne Visible / Infrared Imaging Spectrometer. [Online] Available http://aviris.jpl.nasa.gov/index.html Accessed February 4th, 2015

[6]. Gary L. Prost. “Remote Sensing for Geoscientists: Image Analysis and Integration, Third Edition” [Online] Available https://books.google.com/books?id=64LNBQAAQBAJ&pg=PA73&lpg=PA73&dq=is+HYDICE+still+flown&source=bl&ots=Slaj6-LQeb&sig=1sS4_8bBnAr8OgOa9Ba13QHjDIo&hl=en&sa=X&ei=ttI-VbCmDY-gyQTo9oDIDA&ved=0CB4Q6AEwAA#v=onepage&q=is%20HYDICE%20still%20flown&f=false Accessed May 1st, 2015

[7] “LandSat Program”. [Online] Availablehttp://en.wikipedia.org/wiki/Landsat_program Accessed March 12th, 2015.

[8] Peg Shippert. “Introduction to Hyperspectral Image Analysis”. [Online] Available http://spacejournal.ohio.edu/pdf/shippert.pdf Accessed January 25th, 2015

[9] “What is the most important Remote Sensing Software Tools”. [Online] Available https://fatwaramdani.wordpress.com/2013/05/23/what-is-the-most-important-remote-sensing-software-tools/ Accessed May 1st, 2015

[10] Peg Shippert. (2004). “Why Hyperpsectral Imagery”[Online] Available http://www.asprs.org/a/publications/pers/2004journal/april/highlight.pdf Accessed March 29th, 2015

[11] Ian Dowman. “Hyperspectral Imaging: Beyond the niche” [Online] Available http://geospatialworld.net/Paper/Cover-Stories/ArticleView.aspx?aid=21846 Accessed March 12th, 2015

[12] “Opticks(Software)”. [Online] Available. http://en.wikipedia.org/wiki/Opticks_(software) Accessed Feb 20th, 2015

[13] “Welcome to Opticks”. [Online] Available http://opticks.org/confluence/display/opticks/Welcome+To+Opticks Accessed February 20th , 2015

[14] ArcGIS 9.2 Desktop Help. “BIL, BIP, and BSQ raster files.” September 22, 2008. [Online] Available http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=BIL,_BIP,_and_BSQ_raster_files Accessed March 28th, 2015

[15] Joseph Collins-Unruh. Understanding Rasters. [Online] Available http://www.iro.umontreal.ca/~mignotte/IFT6150/ComplementCours/UnderstandingRasters.pdf Accessed January 25th, 2015

[16] ”HyperCube”. [Online] Available http://www.erdc.usace.army.mil/Media/FactSheets/FactSheetArticleView/tabid/9254/Article/476681/hypercube.aspx Accessed January 25th , 2015

[17] “AVIRIS Distribution Document 111013”. [Online] Available http://aviris.jpl.nasa.gov/alt_locator/111013_AV_Download.readme Accessed Feburary 25th, 2015

[18] California Institute of Technology. “Aster Spectral Library”. [Online] Available. http://speclib.jpl.nasa.gov/ Accessed March 29th, 2015

[19] Baldridge, A. M., S.J. Hook, C.I. Grove and G. Rivera, 2009.. The ASTER Spectral Library Version 2.0. Remote Sensing of Environment, vol 113, pp. 711-715 Accessed March 29th, 2015

[20] “Principal component analysis”. [Online] Available. http://en.wikipedia.org/wiki/Principal_component_analysis Accessed May 1st , 2015

[21] “Normalized Difference Vegetation Index”. [Online] Available http://en.wikipedia.org/wiki/Normalized_Difference_Vegetation_Index Accessed March 15th, 2015

[22] “Spectral Signature”. [Online]. Available. http://en.wikipedia.org/wiki/Spectral_signature / Accessed May 13th, 2015

[23] Core Plot. A plotting framework for OS X and iOS. Eric Skroch. [Online] https://code.google.com/p/core-plot/ Accessed April 2nd, 2015

[24] Exelis Visual Information Solutions. “Spectral Angle Mapper”. [Online] . Available http://www.exelisvis.com/docs/SpectralAngleMapper.html Accessed May 20th, 2015

[25] Roger N. Clark, Gregg A. Swayze. U.S. Geological Survey. Sept, 16, 2002 [Online] .http://speclab.cr.usgs.gov/PAPERS.calibration.tutorial. Accessed May 20th, 2015

[26] “OpenCV”. 2015. [Online] Available. http://opencv.org Accessed February 7thth, 2015

[27] Marcus Borengasser, William S. Hungate. CRC Press, Dec 13, 2007. “Hyperspectral Remote Sensing Principles and Applications.” pp 1-3.

[28] Prasad S. Thenkabail, John G. Lyon. CRC Press. October 25, 2011. “Hyperspectral Remote Sensing of Vegetation”. pp 83- 85

[29] Exelis Visualization Information Services. 2015. “Classification”. [Online] http://www.exelisvis.com/docs/classification.html#ClassUnsupervised Accessed February 27th, 2015.

[30] Heesung Kwon, Dalton Rosario. Army Research Lab. 2005. “Hyperspectral Imaging and Obstacle Detection for Robotics Navigation”. [Online]. http://www.arl.army.mil/arlreports/2005/ARL-TR-3639.pdf . pp ii-2. Accessed May 7th, 2015.

[31] Mac Developer Library. “New Features in Xcode 4.2.”[Online]. https://developer.apple.com/library/mac/documentation/DeveloperTools/Conceptual/WhatsNewXcode/Articles/xcode_4_2.html Accessed May 7th, 2015.

[32] U.S Department of the Interior U.S Geological Survey. “Sensors - Hyperion”. [Online]. http://eo1.usgs.gov/sensors/hyperion Accessed May 7th, 2015.

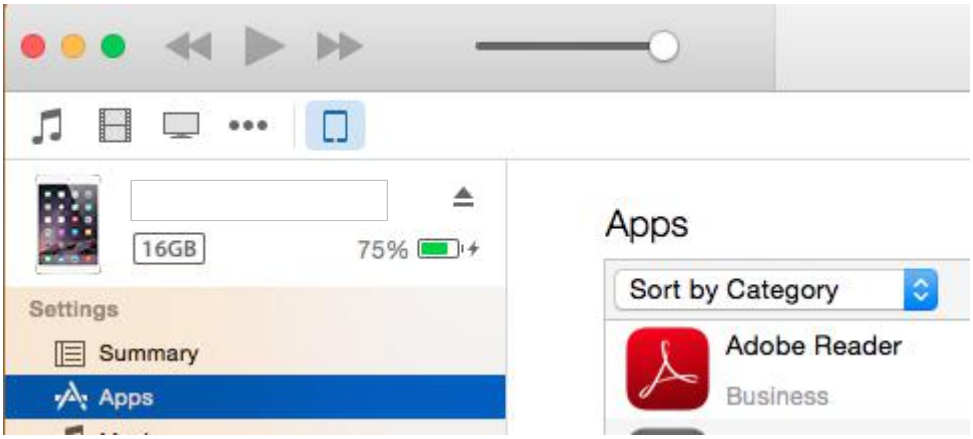
[33] Purdue Resaerch Foundation. September 28, 2011. “MultiSpec A Freeware Multispectral Image Data Analysis System.” [Online] https://engineering.purdue.edu/~biehl/MultiSpec/. Accessed May 7th, 2015.

Appendix I – Installation of Spectral Motion

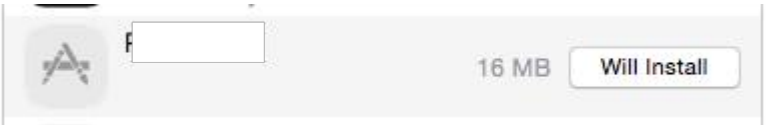
Spectral Motion can be installed and run on iPad 3rd generation or later devices and currently is available as a compiled ipa file, which is an iOS executable.

To install the Spectral Motion ipa file onto an iPad, please follow below instructions:

1. iTunes is the Apple desktop application used for syncing images, videos, and applications to a users iOS device and can be used to import new applications onto an iOS device. iTunes can be downloaded for free onto your Windows or Mac computer from https://www.apple.com/itunes/download/
2. Launch the iTunes application and import the ipa into iTunes by dragging and dropping the ipa into the "Apps" tab in iTunes.
3. Connect your iPad to the computer with an Apple 30-Pin to USB connector or Lightning to USB connector depending on your version of iPad.
4. After connecting your device to your computer, it should appear in iTunes. After it appears, select your device.
5. Select the "Apps" tab under the settings header.



6. Find the app that you would like to install on your iPad and click install. The button title will change to "Will Install".



7. Click on the "Apply" button in the right corner. The app will begin installation onto your iPad and you will see a progress bar under the app icon on your iOS device. After installation has completed, click on the Spectral Motion app icon to launch the application.

Appendix II – Source Code

The source code printout for this project is provided as a separate document. As the code prints on over 150 pages, in agreement with the project advisor, the appendix is made available only in electronic form.