RHSEG User's Manual:

Including the Core RHSEG Open Source Release, HSEGExtract, HSEGReader and HSEGViewer

Version 1.47
December 2, 2009

(Reserved for licensing information)

Table of Contents

Overview

The HSEG algorithm is an image segmentation approach based on iterative hierarchical step-wise region growing. The HSEG algorithm augments the usual region growing segmentation approach by (i) providing for the optional merging of non-adjacent regions (effectively classifying connected region objects into spatially disjoint region classes), and (ii) providing approaches for selecting region growing iterations from which segmentation results are saved to form a segmentation hierarchy. RHSEG is a fast, recursive approximation of HSEG. RHSEG is implemented in a software package that can optionally utilize parallel computing for increased processing speed. With a certain setting of parameters, RHSEG becomes identical to HSEG.

HSEGExtract is a program for extracting certain segmentation features (e.g., region mean, region standard deviation) from selected levels of the segmentation hierarchies produced by HSEG or RHSEG.

HSEGReader is a user interactive program for examining the region class and region object feature values of the regions in the segmentation hierarchies produced by HSEG or RHSEG.

HSEGViewer is a user interactive program for visualizing, manipulating and interacting with the segmentation hierarchies produced by HSEG or RHSEG.

This manual provides detailed instructions on how to install and use HSEG, RHSEG, HSEGExtract, HSEGReader and HSEGViewer.

Documentation Conventions

The following conventions are followed within this document:

- **Bold text** signifies command line text.
- *italicized text* signifies variable names and program parameter names.
- Unless otherwise specified, "clicking" the mouse button means pressing the left mouse button.

Additional Sources of Information

The following web sites can be consulted for additional and updated information:

- http://ipp.gsfc.nasa.gov/RHSEG/
- http://cisto.gsfc.nasa.gov/TILTON/

Chapter 1: What are HSEG, RHSEG, HSEGExtract, HSEGReader and HSEGViewer?

Overview

The HSEG algorithm is an image segmentation approach based on region growing. The HSEG algorithm augments the usual region growing segmentation approach by (i) providing for the optional merging of non-adjacent regions (effectively classifying connected region objects into spatially disjoint region classes), and (ii) providing approaches for selecting region growing iterations from which segmentation results are saved to form a segmentation hierarchy. RHSEG is a fast, recursive approximation of HSEG. RHSEG is implemented in a software package that can optionally utilize parallel computing for increased processing speed. With a certain setting of parameters, RHSEG becomes identical to HSEG.

HSEGExtract is a program for extracting certain segmentation features (e.g., region mean, region standard deviation) from selected levels of the segmentation hierarchies produced by HSEG or RHSEG.

HSEGReader is a user interactive program for examining the region class and region object feature values of the regions in the segmentation hierarchies produced by HSEG or RHSEG.

HSEGViewer is a user interactive program for visualizing, manipulating and interacting with the segmentation hierarchies produced by HSEG or RHSEG.

Since RHSEG is an approximation of HSEG, a basic understanding of image segmentation, segmentation hierarchies, and the HSEG approach for generating segmentation hierarchies is required before RHSEG can be described.

What is Image Segmentation?

Image segmentation is the partitioning of an image into related sections or regions. For remotely sensed images of the earth, an example of an image segmentation would be a labeled map that divides the image into areas covered by distinct earth surface covers such as water, snow, types of natural vegetation, types of rock formations, types of agricultural crops and types of other man created development. In unsupervised image segmentation, the labeled map may consist of generic labels such as region 1, region 2, etc., which may be converted to meaningful labels by a post-segmentation analysis.

What is a Segmentation Hierarchy?

A segmentation hierarchy is a set of several image segmentations of the same image at different levels of detail in which the segmentations at coarser levels of detail can be produced from simple merges of regions at finer levels of detail. This is useful for applications that require different levels of image segmentation detail depending on the characteristics of the particular image objects segmented. A unique feature of a segmentation hierarchy that distinguishes it from most other multilevel representations is

that the segment or region boundaries are maintained at the full image spatial resolution for all levels of the segmentation hierarchy.

In a segmentation hierarchy, an object of interest may be represented by multiple image segments in finer levels of detail in the segmentation hierarchy, and may be merged into a surrounding region at coarser levels of detail in the segmentation hierarchy. If the segmentation hierarchy has sufficient resolution, the object of interest will be represented as a single region segment at some intermediate level of segmentation detail. The segmentation hierarchy may be analyzed to identify the hierarchical level at which the object of interest is represented by a single region segment. The object may then be potentially identified through its spectral and spatial characteristics. Additional clues for object identification may be obtained from the behavior of the image segmentations at the hierarchical segmentation levels above and below the level(s) at which the object of interest is represented by a single region.

What is HSEG?

Hierarchical Step-Wise Optimization (HSWO) is a form of region growing segmentation that directly forms a segmentation hierarchy [1]. HSWO is an iterative process, in which the iterations consist of finding the best segmentation with one region less than the current segmentation. The HSWO approach can be summarized as follows:

1. Initialize the segmentation by assigning each image pixel a region label. If a pre-segmentation is provided, label each image pixel according to the pre-segmentation. Otherwise, label each image pixel as a separate region.
2. Calculate the dissimilarity criterion value between all pairs of spatially adjacent regions, find the pair of spatially adjacent regions with the smallest dissimilarity criterion value, and merge that pair of regions.
3. Stop if no more merges are required. Otherwise, return to step 2.

HSWO naturally produces a segmentation hierarchy consisting of the entire sequence of segmentations from initialization down to the final trivial one region segmentation (if allowed to proceed that far). For practical applications, however, a subset of segmentations needs to be selected out from this exhaustive segmentation hierarchy.

HSEG adds to HSWO approaches for selecting such a subset of segmentations. By default, the subset is selected that minimizes the number of hierarchical levels utilized to guarantee that each region is involved in no more than one merge from one hierarchical level to the next. The user may instead choose to explicitly a set of iterations based on the number of regions or merge thresholds at those iterations. Further, since segmentation results with a large number of regions are usually not interesting, the hierarchical segmentation results are not output until the number of regions reaches a user specified value, *chk_nregions*.

HSEG also optionally interjects between HSWO iterations merges of spatially non-adjacent regions (i.e., spectrally based merging or clustering) constrained by a threshold derived from the previous HSWO iteration [2]. The relative importance of region growing and spectral clustering merges is controlled by the parameter *spclust_wght*, which can vary from 0.0 to 1.0. When *spclust_wght* = 0.0, only merges between spatially

adjacent regions are allowed (no spectral clustering). With *spclust_wght* = 1.0, merges between spatially adjacent and spatially non-adjacent regions are given equal priority. For 0.0 < *spclust_wght* < 1.0, spatially adjacent merges are given priority over spatially non-adjacent merges by a factor of 1.0/*spclust_wght*. Thus for *spclust_wght* > 0.0, spatially connected region objects may be grouped or classified into spatially disjoint region classes.

While the addition of constrained spectral clustering significantly reduces the number of regions required to characterize an image, especially for larger highly varied images, it also significantly increases HSEG's computational requirements. This increase in computational requirements is counteracted by RHSEG, a computationally efficient recursive approximation of HSEG.

What is RHSEG?

RHSEG is a recursive, divide-and-conquer, approximation of HSEG. Following [3] and [4], it can be described for N_D spatial dimension image data as:

1. Given an input image *X*, specify the number of levels of recursion (*rnb_levels*) required and pad the input image, if necessary, so that for each spatial dimension the image can be evenly divided by $2^{(rnb\_levels-1)}$. (A good value for *rnb_levels* results in an image section at *level* = *rnb_levels* consisting of roughly 1000 to 4000 pixels.) Set *level* = 1.
2. Call rhseg*(level,X).*
3. Execute the HSEG algorithm on the image *X* using as a pre-segmentation the segmentation output by the call to rhseg( ) in step 2.

where rhseg(*level,X*) is as follows:

2.1. If *level* = *rnb_levels*, go to step 2.3. Otherwise, divide the image data into 2^{N_D} equal subsections and call rhseg(*level+1,* $X/2^{N_D}$) for each image section (represented as $X/2^{N_D}$).

2.2. After all 2^{N_D} calls to rhseg() from step 2.1 complete processing, reassemble the image segmentation results.

2.3. If *level* < *rnb_levels*, initialize the segmentation with the reassembled segmentation results from step 2.2. Otherwise, initialize the segmentation with one pixel per region. Execute the HSEG algorithm on the image *X* with the following modification: Terminate the algorithm when the number of regions reaches the preset value *min_nregions*.

Note that *rnb_levels* and *min_nregions* are user specified parameters (with default values available).

Under a number of circumstances, the segmentations produced by the RHSEG algorithm exhibit processing window artifacts. These artifacts are region boundaries that are along the processing window seams, even though the image pixels across the seams are very similar. Processing window artifacts are usually minor, but can be more noticeable, depending on the image. They tend to be more noticeable and prevalent in larger images.

However, the processing window artifacts can be completely eliminated by adding a 4th step to the definition of rhseg(*level*,*X*) given above (following [5] and [6]):

2.4. If *level* = *rnb_levels*, exit. Otherwise do the following (and then exit):

a. For each region, identify other regions that may contain pixels that are more similar to it than the region that they are currently in. These regions are placed in a *candidate_region_label* set for each region. This is done by:

   i. scanning the processing window seam between sections processed at the next deeper level of recursion for pixels that are more similar (by a factor of *seam_threshold_factor*) to the region existing across the processing window seam.

   ii. for, *spclust_wght* > 0.0, identifying regions that have a dissimilarity between each other less than *region_threshold_factor***max_threshold*).

b. For each region with a non-empty *candidate_region_label* set, identify pixels in the region that are more similar by a factor of *split_pixels_factor* to regions in the *candidate_region_label* set than to the region they are currently in. If *spclust_wght* = 1.0, simply switch the region assignment of these pixels to the more similar region. Otherwise, split these regions out of their current regions and remerge them through a restricted version of HSEG in which region growing is performed with these split-out pixels and merging is restricted to neighboring regions and regions in the *candidate_region_label* set from which the pixel came from.

Processing window artifact elimination as introduced here not only eliminates the processing window artifacts, but does so with minimal computational overhead. The computational overhead is no more than doubles for a wide range of image sizes [6]. The program defaults for the parameters values *seam_threshold_factor* = 1.3, *split_pixels_factor* = 1.4 work well for a wide range of images. (The default value for *region_threshold_factor* is 0.0, as this aspect of the processing window artifact elimination procedure is usually unnecessary.)

What is HSEGExtract?

HSEGExtract is a program written in C++ for extracting certain segmentation features from selected levels of the segmentation hierarchies produced by HSEG or RHSEG.

With HSEGExtract, an analyst can select a particular hierarchical segmentation level and then output region class or region objects features from the selected hierarchical level in the form of ENVI format images. The following region class or region objects features may be output: Region labels, region number of pixels, region mean, region standard deviation and region boundary pixel to number of pixels ratio.

What is HSEGReader?

HSEGReader is a graphical user interactive (GUI) program written in C++ (utilizing the gtkmm GUI library, see http://www.gtkmm.org/) that enables an analyst to examine the feature values of the region classes and region objects contained in the hierarchical segmentation results produced by HSEG or RHSEG.

With HSEGReader, an analyst can select a particular hierarchical segmentation level and then view the feature values of the region classes in the segmentation at that particular hierarchical level. The analyst can order the region classes by size, standard deviation or boundary pixel ratio feature values. Then for each region class the analyst can view the feature values of the region objects contained in that particular region class. These region objects can also be ordered by size, standard deviation or boundary pixel ratio feature value.

What is HSEGViewer?

HSEGViewer is a graphical user interactive (GUI) program written in C++ (utilizing the gtkmm GUI library, see http://www.gtkmm.org/) that enables an analyst to visualize, manipulate, and interact with, the hierarchical segmentation results produced by HSEG or RHSEG. It is based on an earlier version of HSEGViewer that was written in Java, which was, in turn, based on the "Region Labeling Tool" [7], an earlier GUI program written in C.

With HSEGViewer, an analyst can view pseudo-color (random color table) versions of the region class and region object segmentations at each hierarchical level saved by HSEG or RHSEG, as well as view a region mean image and hierarchical region boundary map image for each of these segmentations. An analyst can also select a particular region class or object from a particular hierarchical level and label it with a selected color and ASCII text string. With this region selection and labeling facility, an analyst can selectively create a tailored image labeling. HSEGViewer also displays certain region statistics for the selected region class or object over all hierarchical levels, including region number of pixels, region mean vector values, region boundary pixel ratio, and region standard deviation.

References

[1] J-M. Beaulieu and M. Goldberg, "Hierarchy in picture segmentation: A stepwise optimal approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 2, pp. 150-163, Feb. 1989.

[2] James C. Tilton, "Image Segmentation by Region Growing and Spectral Clustering with a Natural Convergence Criterion," *Proceedings of the 1998 International Geoscience and Remote Sensing Symposium*, Seattle, WA, pp. 1766-1768, July 6-10, 1998.

[3] J. C. Tilton, "D-dimensional formulation and implementation of recursive hierarchical segmentation," *Disclosure of Invention and New Technology: NASA Case No. GSC 15199-1*, May 26, 2006.

[4] J. C. Tilton, "Parallel Implementation of the Recursive Approximation of an Unsupervised Hierarchical Segmentation Algorithm," Chapter 7 of *High Performance Computing in Remote Sensing*, CRC Press, to be published in late 2007.

[5] James C. Tilton, "Method for recursive hierarchical segmentation which eliminates processing window artifacts," *Disclosure of Invention and New Technology: NASA Case No. GSC 14,681-1*, October 11, 2002 (revised January 24, 2003). NOTE: U. S. Patent Application Serial No. 10/845,419 was filed on this technology on May 11, 2004.

[6] James C. Tilton, "A split-remerge method for eliminating processing window artifacts in recursive hierarchical segmentation," *Disclosure of Invention and New Technology: NASA Case No. GSC 14,994-1*, February 9, 2005.). NOTE: U. S. Patent Application Serial No. 11/251,530 was filed on this technology on September 29, 2005. This patent application includes and supersedes the previous application Serial No. 10/845,419.

[7] James C. Tilton, "A Region Labeling Tool for use with Hierarchical Segmentation," *Disclosure of Invention and New Technology: NASA Case No. GSC 14,331-1*, February, 29, 2000. See also http://cisto.gsfc.nasa.gov/TILTON/publications/region_label_disclosure/NF1679.html.

Chapter 2: Installing the Programs

Overview

The HSEG/RHSEG package is available in four versions. Each version is installed differently. This chapter provides detailed information on how to install each version. The four versions are:

1. Core RHSEG Open Source Release Version
2. RHSEG Demonstration Version
3. RHSEG Licensed Serial Version
4. RHSEG Licensed Parallel Version

All versions also include HSEGExtract, HSEGReader and HSEGViewer.

NOTEs: The Core RHSEG Version provides a 2-Dimensional version of RHSEG – with the recursive subdivision of the input data, but without the processing window artifact elimination code. The RHSEG Demonstration Version provides a 2-Dimensional version of HSEG/RHSEG with the processing window artifact elimination code. The 3-Dimensional version is available only be special arrangement or with a licensed version of RHSEG.

Core RHSEG Open Source Release Version

The Core RHSEG open source release version is built and installed in the same manner that the RHSEG licensed serial version is built and installed. Just substitute "core_rhseg" for "rhseg" in the discussion for the RHSEG licensed serial version. The instructions concerning the "serialkey" and 3-dimensional processing do not apply to the Core RHSEG open source release version.

RHSEG Demonstration Version

Obtain a copy of RHSEG_install.exe and copy it to a convenient location on your computer. You can start the RHSEG installation by double clicking on the program icon. You can also start the installation by clicking on "start", then on "Run." After browsing for the RHSEG_install.exe program, run it by clicking on "OK."

By default, the RHSEG package is installed in the C:\Program Files\RHSEG directory. However, you can choose to have it installed in another directory if you so desire. You can then choose to have RHSEG installed in an existing Program Manager Group, or in its own group called "RHSEG." Following the simple installation instructions will complete the process.

What the installation does:

The RHSEG suite of executables (rhsegGUI.exe, rhseg.exe, hsegextract.exe, hsegreader.exe and hsegviewer.exe) and several other associated files are copied to the installation directory (by default, C:\Program Files\RHSEG). A subdirectory, named "Sample Data," is also created into which a sample image data set and parameter file are copied. You can use this sample image data set and parameter file to test your installation

of RHSEG. Finally the dlls subdirectory is also created. This subdirectory contains the necessary dll library files.

The directory paths to the RHSEG executables and the dll library files (by default C:\Program Files\RHSEG and C:\Program Files\RHSEG\dll\bin, respectively) are added to the system PATH environment variable.

An entry "RHSEG" is added to the Program List (accessed through the "start" button). Included under the RHSEG entry, are subentries "RHSEG User's Manual," "RHSEG," "HSEGExtract," "HSEGReader," and "HSEGViewer."

RHSEG Licensed Serial Version

The RHSEG suite of programs is written in C++. To install the programs, you need to compile and link the provided source code with a C++ compiler.

The instructions provided here assume you have a GNU C++ (gcc) compiler installed under a LINUX or UNIX (e.g., Sun Solaris) operating system, or under a Linux-type environment on Windows. Use cygwin (http://www.cygwin.com/), djgpp (http://www.delorie.com/djgpp) or MinGW-msys (http://www.mingw.org) to provide a Linux-type environment for Windows machines. Tests were performed using gcc versions 3.4.4, 3.4.5 and 4.1.2. Results are not guaranteed for other compilers and systems, though it is very likely that you will be successful compiling the code in other environments (particularly with the GNU C++ compiler).

The RHSEG program is available in three versions. Building the "rhseg_run" version requires nothing else besides a C++ compiler. However, the "rhseg" version *requires* that you have the Geospatial Data Abstraction Library (GDAL) installed on your computer. If GDAL is not already installed on your computer go to http://www.gdal.org/ and install GDAL before proceeding further in building the "rhseg" version of RHSEG. The "rhsegGUI" version of RHSEG (RHSEG with a graphical user interface) also requires that you have gtkmm (the C++ Interface for GTK+) installed on your system. GTK+ is a toolkit for creating graphical user interfaces. See http://www.gtkmm.org to download and install this software. The "rhsegGUI" of RHSEG also needs the pthreads library. The GNU Portable Threads version of pthreads can be obtained from http://www.gnu.org/software/pth/.

Please also see the note at the end of this section for advice on installing GDAL, gtkmm and pthreads.

Once your licensing agreement is finalized, you should be provided with copies of rhsegV1.47.tar.gz and CommonV1.47.tar.gz. As the first step in building the suite of RHSEG programs, place rhsegV1.47.tar.gz in an appropriate directory (e.g., $HOME/src/RHSEG) and uncompress and extract the files from this gzip'd tar file using gunzip and tar (or just tar with the "z" option) as follows:

gunzip rhsegV1.47.tar.gz

and

tar xf rhsegV1.47.tar

or

tar xzf rhsegV1.47.tar.gz

Upon completing the above, you will see a directory rhsegV1.47 with various subdirectories. In the following we will refer to this directory (with the suggested full path $HOME/src/RHSEG/rhsegV1.47) as RHSEG_DIR.

You will find four makefiles in the RHSEG_DIR directory. Makefile and Makefile_serialkey are used to build the "rhseg" version of RHSEG, with or without serial key dependency, respectively. (The serial key function is used to limit access to the RHSEG program executable.) Makefile_gtkmm and Makefile_serialkey_gtkmm are use to build the "rhsegGUI" version of RHSEG, with or without serial key dependency, repectively. There are several macros defined near the beginning of these makefiles. The first definitions listed are appropriate for most Linux operating system environments. The other definitions may be appropriate for Windows environments. Be sure to examine each makefile and make any changes in the macro definitions that might be appropriate for your operating system environment.

The define.h file in RHSEG_DIR contains definitions for compiler flags and program constants. Check to see whether or not the "WINDOWS" compiler flag is left defined or not. If you are on a Windows machine using djgpp or MinGW-msys, you need to make sure this compiler flag is left defined (for cygwin, leave the compiler flag CYGWIN defined instead). Look for the line:

//#undef WINDOWS // Leave defined if compiling under Windows

and make sure it is commented out (as above) if you are on a Windows machine using cygwin, djgpp or MinGW-msys. Otherwise, make sure this compiler flag is undefined by editing this line (if necessary) to make it look like the following:

#undef WINDOWS // Leave defined if compiling under Windows

Since you are a licensed user of RHSEG, you will also want to make sure the "SERIALKEY" compiler flag is undefined. Otherwise you will have to enter in a "Serial Key" to run "rhseg" or "rhseg_run." Make sure this compiler flag is undefined by editing the line

#undef SERIALKEY // Leave defined if using the Serial Key code for access control

(if necessary) to make it look as above.

You should not need to modify the definitions of any of the other compiler flags in define.h.

To build the "rhseg_run" version of RHSEG, go to the RHSEG_DIR/rhseg_run directory and execute the command:

make all

after making sure that the macro definitions in RHSEG/DIR/rhseg_run/Makefile are appropriate for your operating system environment. In some environments you might see warning like

warning: '__cur' might be used uninitialized in this function

that you should ignore. Besides this warning, this version of "rhseg_run" should build cleanly. Be sure to fix any problems encountered here before trying to build any other versions of RHSEG or any of the companion programs (HSEGExtract, HSEGReader or HSEGViewer). Contact the author of this User's Manual if you can't solve the problems on your own.

The "rhseg_run" version of RHSEG requires that your input data be put into a simple "raw" data format. You can use the "rhseg_setup" to do this for you. The "rhseg_setup" program requires GDAL, and can be built with or without a GUI.

Since "rhseg_setup" requires GDAL, it also needs CommonV1.47.tar.gz (which contains interfaces to various GDAL and gtkmm routines). Place this file in an appropriate directory (e.g., $HOME/src) and unpack it. You will now see the directory CommonV1.47 with various subdirectories. The path to this directory will be referred to as COMMON_DIR (e.g., $HOME/src/CommonV1.47) in the following.

To build the non-GUI, GDAL version of "rhseg_setup," undefine the GTKMM compiler flag in the RHSEG_DIR/rhseg_setup/main.h file. Look for the line:

#undef GTKMM // GTKMM may be undefined if a GUI is not desired as an option

and edit this line, if necessary, so that there is no "//" at the beginning of this line (as shown). Also make sure RHSEG_DIR and COMMON_DIR and the other macros are defined appropriately for your operating system environment in RHSEG_DIR/ rhseg_setup/Makefile. Then build and install this version of "rhseg_setup" by executing the command:

make all

in the RHSEG_DIR/rhseg_setup directory.

To build the GUI, GDAL version of "rhseg_setup," you will need to have gtkmm installed on your computer. Once gtkmm is installed, you may build this version of "rhseg_setup" as follows. Make sure that the GTKMM compiler flag is left defined in the RHSEG_DIR/rhseg_setup/main.h file. Look for the line:

//#undef GTKMM // GTKMM may be undefined if a GUI is not desired as an option

and edit this line, if necessary, so that there is a "//" at the beginning of this line (as shown). Also make sure RHSEG_DIR and COMMON_DIR and the other macros are defined correctly in RHSEG_DIR/rhseg_setup/Makefile_gtkmm. Then build and install this version of "rhseg_setup" by executing the command:

make –f Makefile_gtkmm all

in the RHSEG_DIR/rhseg_setup directory.

The "rhseg_setup" and "rhseg_run" programs are designed to work together. The "rhseg_setup" program converts the input image data from various popular image data formats to the plain "raw" data format required by "rhseg_run." It also creates the input parameter file "rhseg_run.params" for "rhseg_run." This arrangement is most useful when it is advantageous to run the "rhseg_run" program on a separate more powerful computer (such as a parallel cluster) that does not have GDAL or gtkmm installed on it.

You may also build and install a consolidated version of RHSEG, "rhseg." This version of RHSEG requires GDAL and may be built with or without a GUI. Follow the following steps to build the non-GUI version of "rhseg." First check to make sure that COMMON_DIR is defined correctly in RHSEG_DIR/Makefile. Then verify that the GTKMM compiler flag is undefined in the RHSEG_DIR/main.h file by editing the appropriate line (if necessary) to look like the following:

#undef GTKMM // GTKMM may be undefined if a GUI is not desired as an option

Then build and install the non-GUI, GDAL version of RHSEG ("rhseg") by executing the command:

make all

in the RHSEG_DIR directory.

Follow the following steps to build the GUI version of "rhseg." First check to make sure that COMMON_DIR is defined correctly in RHSEG_DIR/Makefile_gtkmm. Then verify that the GTKMM compiler flag is *not* undefined in the RHSEG_DIR/main.h file by editing the appropriate line (if necessary) to look like the following:

//#undef GTKMM // GTKMM may be undefined if a GUI is not desired as an option

Then build and install the GUI, GDAL version of RHSEG ("rhseg") by executing the command:

make –f Makefile_gtkmm all

in the RHSEG_DIR directory.

If you will be using RHSEG primarily to process 2-dimensional image data, you should use the 2d version of RHSEG. While the 3d version of RHSEG can process 2-dimensional (and 1-dimensional) data by setting *nslices* = 1 (and *nrows* = 1), the 2d version of RHSEG will process 2-dimensional (and 1-dimensional) data more efficiently. The RHSEG suite of programs is by default set to compile for 2-d processing.

If you do want process 3-d data, you will need to leave the THREEDIM compiler flag defined in the define.h file. Look for the line:

//#undef THREEDIM // Leave defined if three-dimensional processing is desired.

and edit this line, if necessary, so that "//" is at the beginning of this line (as shown).

RHSEG Licensed Parallel Version

TBD

HSEGExtract, HSEGReader, and HSEGViewer

You can build hsegextract, hsegreader and hsegviewer by going to the appropriate directory, RHSEG_DIR/rhseg_extract, RHSEG_DIR/rhseg_read and RHSEG_DIR/rhseg_view in turn (RHSEG_DIR/hsegextract, RHSEG_DIR/hsegreader and RHSEG_DIR/hsegviewer for the core RHSEG open source version) and following the same procedure you used to build rhseg. For hsegextract and hsegreader you have the option of building either a GUI or non-GUI version, and hsegreader does not require

GDAL. In building these programs, make sure COMMON_DIR and RHSEG_DIR and the other macros are defined correctly in the makefiles.

Advice on Installing GDAL, gtkmm and pthreads

Installation packages for GDAL and gtkmm binaries are available for most Linux operating systems and for the MinGW-msys environment under Windows. While pthread binaries don't appear to be generally available, this package should be easy to build from source under most Linux operating systems. Pre-built versions of the latest DLL, development library and include files for pthreads on Windows (including MinGW-msys) are available from ftp://sourceware.org/pub/pthreads-win32/dll-latest.

Unfortunately such installation packages and/or pre-built libraries for GDAL and gtkmm are not available for djgpp and cygwin under Windows. A build should be possible under djgpp, but this has not been tested by the author. The author has successfully used the following procedure to build GDAL, gtkmm and also pthreads (required for the GUI version of RHSEG) from source under cygwin. Source code for these three packages are available from http://www.gtkmm.org/, http://www.gdal.org/ and http://www.gnu.org/software/pth/, respectively.

GTK+ is required for gtkmm. The version of GTK+ available through cygwin is too old to work with the version gtkmm required by RHSEG (in particular, for hsegviewer). Building GTK+ and its dependencies from source is a big challenge.

I found the tutorial posted at http://kemovitra.blogspot.com/2009/06/cygwin-tutorial-compiling-gtk-2162-for.html to be invaluable for building the needed version of GTK+. I installed the latest version of each package described using the approach outlined in the tutorial:

glib-2.20.5, jpegsrc.V7, tiff-3.8.2, atk-1.27.90, cairo-1.8.8, pango-1.24.5, and gtk+-2.16.6

Additional prerequisites for gtkm are cairomm, glibmm, libsigc++, and pangomm.

I successfully built the following in a similar manner to the advice provided by the GTK+ tutorial:

cairomm-1.8.2, glibmm-2.20.1, libsigc++-2.2.4 and pangomm-2.24.0.

In cairomm/fontface.cc I had to make the following modification:

I changed:

```
cairo_font_face_set_user_data(cobj(),
&USER_DATA_KEY_DEFAULT_TEXT_TO_GLYPHS,
reinterpret_cast<void*>(true), NULL);
```

to:

```
int int_true = true;

cairo_font_face_set_user_data(cobj(),
&USER_DATA_KEY_DEFAULT_TEXT_TO_GLYPHS,
reinterpret_cast<void*>(int_true), NULL);
```

I had to hack (glibmm)/examples/compose/main.cc in order to get glibmm to build.

I commented out the lines:

```
//    << std::endl
//    << ustring::compose("a : b = [%1|%2]",
//                        ustring::format(std::setfill(L'a'), std::setw(i), ""),
//                        ustring::format(std::setfill(L'b'), std::setw(40 - i), ""))
```

as shown above. (Let me know if you have a better solution. The above really is a bad hack!)

I found that the install of GDAL is straightforward. Here you can use the latest version. It's probably best to use the built-in versions of libz, libtiff, libgeotiff, libpng, libgif and libjpeg, as these versions are probably newer than those provided by cygwin. I built GDAL version 1.6.2.

I found that the install of pthreads was straightforward. Here you also can use the latest version (2.0.7).

Chapter 3: Running the Programs

Overview

This chapter provides an overview of how to run the Core RHSEG, RHSEG, HSEGExtract, HSEGReader and HSEGViewer suite of programs.

Running the Core RHSEG program

The Core RHSEG open source release version is run in the same manner that the RHSEG licensed serial or RHSEG demonstration versions run are run. Just substitute "core_rhseg" for "rhseg" in the discussion provided for running RHSEG in the following section. The instructions concerning the "serialkey" and 3-dimensional processing do not apply to the Core RHSEG open source release version.

Running RHSEG

There are three versions of the RHSEG program provided with the demonstration version of RHSEG, all of which can also be built in the licensed version. The version that does not depend on gtkmm or GDAL is named "rhseg_run," the version that depends upon GDAL but not gtkmm is named "rhseg," and the version that depends on both gtkmm and GDAL is named "rhsegGUI." There is also a companion program to "rhseg_run" called "rhseg_setup," which depends on both gtkmm and GDAL.

The version of RHSEG most similar to previous versions is "rhseg_run." This is also the only version that can be used for three-dimensional image processing (licensed version only). To run "rhseg_run," create a parameter file with the appropriate entries, and run the program with the following command:

rhseg_run parameter_file_name

The parameter file consists of entries of the form:

-parameter_name parameter_values(s)

This parameter file may be constructed manually following the definitions provided in the on-line help, which may be obtained through the command:

rhseg_run –h or **rhseg_run -help**

and/or by referring Chapter 4 of this User's Manual.

The "rhseg_run" version of RHSEG requires that the input data be a headerless binary 1-, 2-, or 3-spatial dimension image or image-like data file in band sequential format. See Chapter 4 of this User's Manual for more details.

The parameter file may also be constructed automatically using the "rhseg_setup" program with the command:

rhseg_setup

Invoking this command will display a GUI through which you can provide the input parameter information. In this case, the input image data must be in one of the image data

formats recognized by GDAL instead of the headerless binary data file expected by "rhseg_run." You must specify the "Input image data file," "the relative importance of spectral clustering vs. region growing," and the "Output log file" through this GUI. You may also optionally specify an "Input mask data file" and "Input region map data file" as well as specify a "Dissimilarity Criterion" other than the default "Square Root of Band Sum Mean Squared Error." Once the requirements of this GUI panel are satisfied, you may either run the program by selecting the "Program Action" "Run RHSEG," or specify additional output files by selecting the "Program Action" "Go to Next Panel." From the "RHSEG Output File Specification" panel you may also similarly choose to "Run RHSEG," or "Go to Next Panel." In this case, the next panel allows you to specify non-default values for other RHSEG parameters. From this "RHSEG Parameter Specification" panel you may also run the program by selecting the "Program Action" "Run RHSEG."

"Running" RHSEG from the "rhseg_setup" program does not actually run the RHSEG algorithm, but instead creates an input parameter file for "rhseg_run" (with the default name "rhseg_run.params"). It also creates the headerless binary input data files required by "rhseg_run."

You may also run RHSEG from a command line with a parameter file using the "rhseg" command. In contrast the "rhseg_run," in this case the input image data must be in one of the image data formats recognized by GDAL instead of the headerless binary data file expected by "rhseg_run." At a minimum you must also specify a value for the parameters *spclust_wght* and *log* (see Chapter 4 of this User's Manual).

Invoking "rhsegGUI" (without the parameter file name) will bring up the GUI version of RHSEG. The GUIs are exactly the same as described in the discussion of "rhseg_setup" above. In this case, though, when you request "Run RHSEG" you will actually run the RHSEG algorithm! If you like, you can create a shortcut for this program and place it on your desktop.

When RHSEG is run using the "rhsegGUI" version, upon completion of the RHSEG program, the user is given the opportunity to run HSEGReader, HSEGViewer and/or display the log file by selecting buttons on a GUI.

Again, for help on the parameter file entries, type

rhseg –h or **rhseg -help**

To find out the version of your copy of RHSEG type

rhseg –v or **rhseg -version**

Special notes for the RHSEG demonstration version:

Notes for the demonstration version:

(i) For the demonstration version, the first time you run the "rhseg" or "rhseg_run" version of the RHSEG program, you will be prompted for your user name and Serial Key, which should have been provided to you with RHSEG_setup.exe. This information is written to a file in the system TEMP directory. Subsequent runs of RHSEG read the user name and Serial Key information from this file, and you will not be prompted at all. However, if this file gets corrupted or deleted - then you will again be prompted for the

information. In this case, reenter your original Serial Key, or obtain a new Serial Key from the distributor of RHSEG_setup.exe. When your time allotment expires you will again be prompted to enter your user name and Serial Key. In this case, you will have to contact the distributor of RHSEG_setup.exe for terms under which a new Serial Key can be obtained, or for arranging the procurement of a licensed version of RHSEG.

(ii) For the initial run of RHSEG, serialkey information must be entered via the command line (non-GUI) version of RHSEG, called from a DOS command window or a LINUX terminal window. Subsequent runs of RHSEG (within the demonstration period) may be run, if desired, through the GUI version.

For the demonstration version, the Sample Data folder in the RHSEG installation directory (C:\Program Files\RHSEG by default) contains the following sample files:

girl.bmp	Sample Image Data File
rhseg.params	Sample Parameter File

Using these files, you can test RHSEG by bringing up a DOS window, setting your directory location to this Sample Data folder and typing the command:

rhseg rhseg.params

The program should take a little over 1 minute to run on a 2 GHz clock machine. You may also use this sample data set to test the GUI version of RHSEG.

Running HSEGExtract

In running the HSEGExtract program, you will find it most convenient to set your directory location to the directory where the output files from a run of the RHSEG reside, but this is not necessary. You may run HSEGExtract with either of the following commands:

hsegextract

or

hsegextract parameter file name

The first choice brings up a parameter input GUI. This version of HSEGExtract can also be called from "RHSEG" group in "All Programs" in the "start" menu in Windows. If you like, you can create a shortcut for this program and place it on your desktop.

The second choice uses an input parameter file. A description of the contents of this file can be found using the command:

hsegextract –h or **hsegextract -help**

To find out the version of your copy of HSEGExtract type

hsegextract –v or **hsegextract -version**

The HSEGExtract program is designed to directly use the outputs from the RHSEG program as its inputs. In particular, the output parameter file from RHSEG (with the suffix "oparam") provides most of the needed inputs for the HSEGExtract program.

If you run HSEGExtract with a parameter file and all the required parameters are not specified properly in the parameter file, the program will display the parameter input GUI panel (as if run with the "hsegextract" command). Once all appropriate parameters are entered in the HSEGExtract parameter input GUI panel, select the "Run Program" option in the "Program Actions" menu. The HSEGExtract program will now run producing the selected outputs.

For the demonstration version, the Samples folder in the RHSEG installation directory (C:\Program Files\RHSEG by default) contains the following sample files:

girl.bmp	Sample Image Data File
rhseg.params	Sample Parameter File

As noted earlier in the section "Running RHSEG," RHSEG can be tested by bringing up a DOS window, setting your directory location to this Samples folder and typing the command:

rhseg rhseg.params

Once RHSEG completes, you will find that RHSEG produced the following files:

girl.log	RHSEG log file
girl.oparam	RHSEG output parameter file
girl.class_labels_map	RHSEG region class label map for hierarchical level 0
girl.region_classes	RHSEG region class information (all levels)
girl.boundary_map	RHSEG hierarchical boundary map
girl.object_labels_map	RHSEG region object label map for hierarchical level 0
girl.region_objects	RHSEG region object information (all levels)

HSEGExtract uses these files as input (except for girl.log). The girl.oparam file contains the names of all of the input files for HSEGExtract, plus other required information such as image number of rows and columns, number of regions at hierarchical level 0, and the number of hierarchical levels.

An input parameter file for HSEGExtract is also provided: hsegextract.params.

Thus, HSEGExtract can be run by simply providing it with the name of the HSEGExtract input parameter file:

hsegextract hsegextract.params

An alternate way to run HSEGExtract is with the command:

hsegextract

As noted earlier, this version of HSEGExtract can also be called from "RHSEG" group in "All Programs" in the "start" menu in Windows. In this case you will need to enter "girl.oparam" as the "RHSEG/HSEG Output Parameter File" (as an input to HSEGExtract).

When run with the "hsegextract" command, all the other required parameter value entries will be read from the "RHSEG/HSEG Output Parameter File." Once you have selected values for the optional parameters (or left them at their default values) you run the program by selecting "Run Program," under the "Program Actions" menu (upper left

corner of the panel). The HSEGExtract program will now run producing the selected outputs.

Running HSEGReader

In running the HSEGReader program, you will find it most convenient to set your directory location to the directory where the output files from a run of the RHSEG reside, but this is not necessary. You may run HSEGReader with either of the following commands:

hsegreader

or

hsegreader parameter file name

The first choice brings up a parameter input GUI. This version of HSEGReader can also be called from "RHSEG" group in "All Programs" in the "start" menu in Windows. If you like, you can create a shortcut for this program and place it on your desktop.

The second choice uses an input parameter file. A description of the contents of this file can be found using the command:

hsegreader –h or **hsegreader -help**

To find out the version of your copy of HSEGReader type

hsegreader –v or **hsegreader -version**

The HSEGReader program is designed to directly use the outputs from the RHSEG program as its inputs. In particular, the output parameter file from RHSEG (with the suffix "oparam") provides most of the needed inputs for the HSEGReader program.

If you run HSEGReader with a parameter file and all the required parameters are not specified properly in the parameter file, the program will display the parameter input GUI panel (as if run with the "hsegreader" command). Once all appropriate parameters are entered in the HSEGReader parameter input GUI panel, select the "Run Program" option in the "Program Actions" menu. The main panel for the HSEGReader program will now appear.

For the demonstration version, the Samples folder in the RHSEG installation directory (C:\Program Files\RHSEG by default) contains the following sample files:

girl.bmp	Sample Image Data File
rhseg.params	Sample Parameter File

As noted earlier in the section "Running RHSEG," RHSEG can be tested by bringing up a DOS window, setting your directory location to this Samples folder and typing the command:

rhseg rhseg.params

Once RHSEG completes, you will find that RHSEG produced the following files:

girl.log	RHSEG log file
girl.oparam	RHSEG output parameter file

girl.class_labels_map	RHSEG region class label map for hierarchical level 0
girl.region_classes	RHSEG region class information (all levels)
girl.boundary_map	RHSEG hierarchical boundary map
girl.object_labels_map	RHSEG region object label map for hierarchical level 0
girl.region_objects	RHSEG region object information (all levels)

HSEGReader uses these files as input (except for girl.log). The girl.oparam file contains the names of all of the input files for HSEGReader, plus other required information such as image number of rows and columns, number of regions at hierarchical level 0, and the number of hierarchical levels.

An input parameter file for HSEGReader is also provided: hsegreader.params.

Thus, HSEGReader can be run by simply providing it with the name of the HSEGReader input parameter file:

hsegreader hsegreader.params

An alternate way to run HSEGReader is with the command:

hsegreader

As noted earlier, this version of HSEGReader can also be called from "RHSEG" group in "All Programs" in the "start" menu in Windows. In this case you will need to enter "girl.oparam" as the "RHSEG/HSEG Output Parameter File" (as an input to HSEGReader).

When run with the "hsegreader" command, all the other required parameter value entries will be read from the "RHSEG/HSEG Output Parameter File." Once you have selected values for the optional parameters (or left them at their default values) you run the program by selecting "Run Program," under the "Program Actions" menu (upper left corner of the panel). The main "Hierarchical Segmentation Results Reader" panel will then appear.

Once the "Hierarchical Segmentation Results Reader" panel is displayed, you may select a particular hierarchical segmentation level. Once you do so, you can choose to order the region classes by size, standard deviation or by boundary pixel ratio feature value. The feature values for the appropriate region class are now displayed. You can then examine the next largest region class, or choose to order the region objects contained in the selected region class by size, standard deviation or by boundary pixel ratio feature value. The feature values for the appropriate region object are now displayed.

Running HSEGViewer

In running the HSEGViewer program, you will find it most convenient to set your directory location to the directory where the output files from a run of the RHSEG reside, but this is not necessary. You may run HSEGViewer with either of the following commands:

hsegviewer

or

hsegviewer parameter file name

The first choice brings up a parameter input GUI. This version of HSEGViewer can also be called from "RHSEG" group in "All Programs" in the "start" menu in Windows. If you like, you can create a shortcut for this program and place it on your desktop.

The second choice uses an input parameter file. A description of the contents of this file can be found using the command:

hsegviewer –h or **hsegviewer -help**

To find out the version of your copy of HSEGViewer type

hsegviewer –v or **hsegviewer -version**

The HSEGViewer program is designed to directly use the outputs from the RHSEG program as its inputs. In particular, the output parameter file from RHSEG (with the suffix "oparam") provides most of the needed inputs for the HSEGViewer program.

If you run HSEGViewer with a parameter file and all the required parameters are not specified properly in the parameter file, the program will display the parameter input GUI panel (as if run with the "hsegviewer" command). Once all appropriate parameters are entered in the HSEGViewer parameter input GUI panel, select the "Run Program" option in the "Program Actions" menu. The main panel for the HSEGViewer program will now appear.

Note: The "Output Selected Class Label Map File" and the optional "Output ASCII Class Label Names List File" can be used to store intermediate results from one session of the HSEGViewer program. These files can be used as the "Input Class Label Map File" and "Input ASCII Class Label Names List File", respectively, to start up where you left off in a previous session. (The ASCII Class Label Names List File also includes color map information.)

For the demonstration version, the Samples folder in the RHSEG installation directory (C:\Program Files\RHSEG by default) contains the following sample files:

girl.bmp	Sample Image Data File
rhseg.params	Sample Parameter File

As noted earlier in the section "Running RHSEG," RHSEG can be tested by bringing up a DOS window, setting your directory location to this Samples folder and typing the command:

rhseg rhseg.params

Once RHSEG completes, you will find that RHSEG produced the following files:

girl.log	RHSEG log file
girl.oparam	RHSEG output parameter file
girl.class_labels_map	RHSEG region class label map for hierarchical level 0
girl.region_classes	RHSEG region class information (all levels)
girl.boundary_map	RHSEG hierarchical boundary map
girl.object_labels_map	RHSEG region object label map for hierarchical level 0
girl.region_objects	RHSEG region object information (all levels)

HSEGViewer uses these files as input (except for girl.log), along with the original data file (in this case, girl.bmp). The girl.oparam file contains the names of all of the input files for HSEGViewer, plus other required information such as image number of rows and columns, number of regions at hierarchical level 0, and the number of hierarchical levels.

An input parameter file for HSEGViewer is also provided: hsegviewer.params.

Thus, HSEGViewer can be run by simply providing it with the name of the HSEGViewer input parameter file:

hsegviewer hsegviewer.params

An alternate way to run HSEGViewer is with the command:

hsegviewer

As noted earlier, this version of HSEGViewer can also be called from "RHSEG" group in "All Programs" in the "start" menu in Windows. In this case you will need to enter "girl.oparam" as the "RHSEG/HSEG Output Parameter File" (as an input to HSEGViewer).

When run with the "hsegviewer" command, with the exception of the Red, Green and Blue Display Band values, all the other required parameter value entries will be read from the "RHSEG/HSEG Output Parameter File." Since the example girl image is a RGB image with bands stored in the order blue, green then red, enter "0" for the "Red Display Band," "1" for the "Green Display Band," and "2" for the "Blue Display Band." Then, under the "Program Actions" menu (upper left corner of the panel), select "Run Program." The main "Hierarchical Segmentation Results Viewer" panel will then appear.

Details on using the HSEGViewer main panel for viewing and interacting with the hierarchical segmentation results, and more details on configuring and starting the HSEGViewer program are provided in Chapter 5: "HSEGViewer Tutorial."

Chapter 4: Guide to HSEG/RHSEG Parameters and Parameter Settings

Overview

This chapter provides information on the HSEG/RHSEG program parameters and provides some guidance as to appropriate parameter settings.

HSEG/RHSEG Program Parameters

HISTORY: * Version 1.47 is a minor upgrade/bug fix over Version 1.46. Besides some bug fixes, one change is the renaming of the "Extract Region of Interest" option in HSEGViewer to "Circle Region of Interest." Another change is the addition of an option to display the Region Classes in grey scale rather than pseudo color in HSEGViewer.
* Version 1.46 is a minor upgrade/bug fix over Version 1.45. The most significant change is the restoration of the "region of interest" capability to HSEGViewer that was lost when HSEGViewer was converted to C++ from JAVA for Version 1.32. Another important change is that the reading of parameter values from parameter files was made much more robust. With this version there can be any number of spaces or tabs between the parameter name and parameter value.
* Version 1.45 of RHSEG brought with it some significant operational changes: The capability was added to input float format input image data, the *dtype* parameter definitions were changed, a complete GUI was added as an option, and a capability to input image data in a wide variety of popular image data formats was (optionally) provided.
* A "percent complete" tracking facility was added to both the command line and GUI versions of RHSEG with version 1.42.
* With version 1.41, the *chk_mn_std_dev* and *conv_mn_std_dev* input parameters were added in version 1.32 were removed and the behavior of the *region_std_dev* parameter was changed. The optional *hseg_out_nregions* and *hseg_out_thresholds* parameters were added.
* A option for a rudimentary GUI is provided with version 1.40. This GUI option makes it possible to make RHSEG a desktop function in Windows.
* A large number of changes were made to the input parameters with version 1.32. Most of these changes are related to the consolidation of the output region class and region object information into the *region_classes* and *region_objects* files. In addition, the *rlblmap* output was renamed to *class_labels_map*, the *conn_rlblmap* output was renamed to *object_labels_map*, and the *rlblmap_in* input was renamed to *region_map_in*. The *chk_mn_std_dev* and *conv_mn_std_dev* input parameters were also added.
* With version 1.31, the *conv_criterion*, *conv_factor*, *gdissim_crit*, and *gstd_dev_crit* parameters were eliminated, the *rconv_critlist* parameter was renamed to *rthreshlist*, and the *gdissim* parameter was added.

* With version 1.30, the *min_npixels_pct* parameter was replaced with *min_npixels*. Also, the default values if *split_pixels_factor*, *seam_threshold_factor* and *region_threshold_factor* were changed and a new dissimilarity criterion (SAR Speckle Noise) was added. Some other changes were made to improve processing efficiency.
* New parameter added with version 1.28: *init_threshold*.
* New parameters added with version 1.20: *nslices, scale,* and *offset* (*rlblmap_mask_value* was removed).
* Parameters revised with version 1.20: *spclust_wght*, *dissim_crit*, *conn_type*, *normind*, *conv_criterion*, and *gdissim_crit*. The parameter *spatial_wght* was renamed to be *std_dev_wght*.
* New parameter added with version 1.10: *ionb_levels*.

NOTE: In the following discussion, most parameters are valid for both HSEG and RHSEG. However some parameters are not valid for HSEG. See the on-line help for HSEG (hseg –h) to determine whether or not a parameter is valid for HSEG

The input image data file name must be specified in the input parameter file:

input_image (string) Input image data file name

The input image data file from which hierarchical image segmentation is to be produced.

For "rhseg_run," this image data file is assumed to be a headerless binary 1-, 2-, or 3-spatial dimension image or image-like data file in band sequential format. This means that the column index increments that fastest, followed by the row index, followed by the slice index, followed by the spectral band index. The number of columns, rows, slices, spectral bands and the data type are specified by other required parameters (see below). Data types "unsigned char (byte)," "short unsigned int," and "float" are supported (see *dtype* below).

For "rhseg_setup," "rhseg" and "hseg" this image data file is assumed to be in one of the wide variety of image data formats supported by GDAL (Geospatial Data Abstraction Library – see http://www.gdal.org/).

For "rhseg_run" (but not for "rhseg_setup," "rhseg" or "hseg") the following parameters must also be specified (no defaults):

ncols (int) Number of columns in the input image data ($0 < ncols < 65535$)

nrows (int) Number of rows in the input image data ($0 < nrows < 65535$)

nslices (int) Number of slices in the input image data (Only for the 3-D version) ($0 < nslices < 65535$)

nbands (int) Number of spectral bands in input image data ($0 < nbands < 65535$)

dtype (string) Data type of input image data:

dtype = UInt8 designates "unsigned char (byte)"
dtype = UInt16 designates "short unsigned int"
dtype = Float32 designates "float"
(otherwise undefined)

The following input image data files may also be specified in the input parameter file:

mask (string) Input data mask file name (default = {none})

The optional input data mask must match the input image data in number of columns and rows. Even if the input image data has more than one spectral band, the input data mask need only have one spectral band. If the input data mask has more than one spectral band, only the first spectral band is used and is assumed to apply to all spectral bands for the input image data. If the data value of the input data mask is not equal to *mask_value* (see the next parameter definition), the corresponding value of the input image data object is taken to be a valid data value. If the data value of the input data mask object is equal to *mask_value*, the corresponding value of the input image data object is taken to be invalid and a region label of "0" is assigned to that spatial location in the output region label map data. For "rhseg_run," the input data mask data type is assumed to be "unsigned char (byte)." Otherwise, the GDAL supported format input data mask is converted, as necessary, to "unsigned char (byte)."

mask_value (int) If input data mask file is provided, this is the value in the mask file that designates bad data. Otherwise this is the value in the input data that designates bad data. (If mask file provided, default = 0 for. "rhseg_run," and the input image data format specified value otherwise. No default if no mask file is provided.)

region_map_in (string) Input region label map file name. (default = {none})

The optional region label map must match the input image data in number of columns and rows (and slices for 3-D). If provided, the image segmentation is initialized according to the input region label map instead of the default of each pixel as a separate region. Wherever a region label of "0" is given by the input region label map, the region labeling is assumed to be unknown and the region label map is initialized to one-pixel regions at those locations (except see *mask_value* above). For "rhseg_run," the input region label map data type is assumed to be "short unsigned int." Otherwise, the GDAL supported format input region label map is converted, as necessary, to "short unsigned int."

The following parameters must also be specified:

spclust_wght (float) Relative importance of spectral clustering versus region growing

($0.0 \leq$ *spclust_wght* ≤ 1.0, no default)

dissim_crit (int) Dissimilarity criterion

1. "1-Norm,"
2. "2-Norm,"
3. "Infinity Norm,"
4. "Spectral Angle Mapper,"
5. "Spectral Information Divergence,"
6. "Square Root of Band Sum Mean Squared Error,"
7. "Square Root of Band Maximum Mean Squared Error,"
8. "Normalized Vector Distance,"
9. "Entropy,"
10. "SAR Speckle Noise."

(default: 6 "Square Root of Band Sum Mean Squared Error")

Criterion for evaluating the dissimilarity of one region versus another.

Dissimilarity criteria 1, 2 and 3 are based on vector norms. The 1-Norm of the difference between the region mean vectors, u_i and u_j, of regions X_i and X_j, each with B spectral bands, is:

$$\left\| u_i - u_j \right\|_1 = \sum_{b=1}^{B} \left| \mu_{ib} - \mu_{jb} \right|, \tag{4-1a}$$

where μ_{ib} and μ_{jb} are the mean values for regions i and j, respectively, in spectral band b, i.e., $u_i = (\mu_{i1}, \mu_{i2}, \ldots, \mu_{iB})^T$ and $u_j = (\mu_{j1}, \mu_{j2}, \ldots, \mu_{jB})^T$. The dissimilarity function for regions X_i and X_j, based on the vector 1-Norm, is given by:

$$\mathrm{d}_{\text{1-Norm}}\left(X_i, X_j\right) = \left\| u_i - u_j \right\|_1. \tag{4-1b}$$

The vector 2-Norm of the difference between the region mean vectors, u_i and u_j, of regions X_i and X_j is:

$$\left\| u_i - u_j \right\|_2 = \left[\sum_{b=1}^{B} \left(\mu_{ib} - \mu_{jb} \right)^2 \right]^{1/2}, \tag{4-2a}$$

The dissimilarity function for regions X_i and X_j, based on the vector 2-Norm, is given by:

$$\mathrm{d}_{\text{2-Norm}}\left(X_i, X_j\right) = \left\| u_i - u_j \right\|_2. \tag{4-2b}$$

The vector ∞-Norm of the difference between the region mean vectors, u_i and u_j, of regions X_i and X_j is:

$$\left\| u_i - u_j \right\|_\infty = \max\left(\left| \mu_{ib} - \mu_{jb} \right|, b = 1,2,\cdots,B \right) \tag{4-3a}$$

The dissimilarity function for regions X_i and X_j, based on the vector ∞-Norm, is given by:

$$d_{\infty\text{-Norm}}(X_i, X_j) = \left\| u_i - u_j \right\|_\infty . \tag{4-3b}$$

Dissimilarity criterion 4 is the Spectral Angle Mapper (SAM) criterion, which is widely used in hyperspectral image analysis [1]. This criterion determines the spectral similarity between two spectral vectors by calculating the "angle" between the two spectral vectors. An important property of the SAM criterion is that poorly illuminated and more brightly illuminated pixels of the same color will be mapped to the same spectral angle despite the difference in illumination. The spectral angle θ between the region mean vectors, u_i and u_j, of regions X_i and X_j is given by:

$$\theta(u_i, u_j) = \arccos\left(\frac{u_i \circ u_j}{\left\| u_i \right\|_2 \left\| u_j \right\|_2} \right) =$$

$$\arccos\left(\frac{\sum_{b=1}^{B} \mu_{ib} \mu_{jb}}{\left(\sum_{b=1}^{B} \mu_{ib}^2 \right)^{1/2} \left(\sum_{b=1}^{B} \mu_{jb}^2 \right)^{1/2}} \right). \tag{4-4a}$$

The dissimilarity function for regions X_i and X_j, based on the SAM distance vector measure, is given by:

$$d_{SAM}(X_i, X_j) = \theta(u_i, u_j) \tag{4-4b}$$

Note that the value of d_{SAM} ranges from 0.0 for similar vectors up to $\pi/2$ for the most dissimilar vectors.

Dissimilarity criterion 5 is the Spectral Information Divergence (SID) criterion, which is derived from the concept of divergence in information theory, and measures the discrepancy of probabilistic behaviors between two spectral signatures [2, 3]. It is based on a process that models the region mean vector, u_i, of region X_i as a random variable. Although the assumption of this model do not necessarily hold true for most images, the effect of the violation is negligible [4]. Noting that, for image data, the elements of u_i are nonnegative, a probability measure for u_i can be defined as

$$q_b(u_i) = \frac{\mu_{ib}}{\sum_{b=1}^{B} \mu_{ib}}, \tag{4-5a}$$

where $u_i = (\mu_{i1}, \mu_{i2}, \ldots, \mu_{iB})^T$ as before. This being the case, the entropy of the region mean vector, u_i, of region X_i is

$$H(u_i) = -\sum_{b=1}^{B} q_b(u_i) \log\left[q_b(u_i) \right] \tag{4-5b}$$

The relative entropy of the region mean vector, u_j, of region X_j with respect to the region mean vector, u_i, of region X_i with can be defined by

$$K(u_i \| u_j) = -\sum_{b=1}^{B} q_b(u_i)\{\log[q_b(u_j)] - \log[q_b(u_i)]\} =$$

$$\sum_{b=1}^{B} q_b(u_i)\log\left[\frac{q_b(u_i)}{q_b(u_j)}\right]. \qquad (4\text{-}5c)$$

$K(u_i\|u_j)$ in (4-5c) is also known as the Kullback-Leibler information measure [5]. The symmetric hyperspectral measure, SID, can be defined using (4-5c) by

$$SID(u_i, u_j) = K(u_i \| u_j) + K(u_j \| u_i) = \sum_{b=1}^{B}\left\{q_b(u_i)\log\left[\frac{q_b(u_i)}{q_b(u_j)}\right] + q_b(u_j)\log\left[\frac{q_b(u_j)}{q_b(u_i)}\right]\right\}. \qquad (4\text{-}5d)$$

The dissimilarity function for regions X_i and X_j, based on the SID vector measure, is given by:

$$d_{SID}(X_i, X_j) = SID(u_i, u_j) \qquad (4\text{-}5e)$$

Dissimilarity criteria 6 and 7 are based on minimizing the increase of mean squared error between the region mean image and the original image data. The sample estimate of the mean squared error for the segmentation of band b of the image X into R disjoint subsets $X_1, X_2, \ldots, X_R$ is given by:

$$MSE_b(X) = \frac{1}{N-1}\sum_{i=1}^{R} MSE_b(X_i), \qquad (4\text{-}6a)$$

where N is the total number of pixels in the image data and

$$MSE_b(X_i) = \sum_{x_p \in X_i}(\chi_{pb} - \mu_{ib})^2 \qquad (4\text{-}6b)$$

is the mean squared error contribution for band b from segment X_i. Here, x_p is a pixel vector (in this case, a pixel vector in data subset X_i), and χ_{pb} is the image data value for the b^{th} spectral band of the pixel vector, x_p. A dissimilarity function based on a measure of the increase in mean squared error due to the merge of regions X_i and X_j is given by:

$$d_{BSMSE}(X_i, X_j) = \sum_{b=1}^{B} \Delta MSE_b(X_i, X_j), \qquad (4\text{-}7a)$$

where

$$\Delta MSE_b(X_i, X_j) = MSE_b(X_i \cup X_j) - MSE_b(X_i) - MSE_b(X_j)\ . \qquad (4\text{-}7b)$$

BSMSE refers to "band sum *MSE*." Instead of summing over the bands in (4-7a) one could take the maximum over the spectral bands, resulting in a "band maximum *MSE*:"

$$d_{BMMSE}(X_i, X_j) = \max\{\Delta MSE_b(X_i, X_j)\, b = 1,2\ldots, B\} \tag{4-7c}$$

Using (4-6b) and exchanging the order of summation, (4-7b) can be manipulated to produce an efficient dissimilarity function based on aggregated region features:

$$\Delta MSE_b(X_i, X_j) = \left(\sum_{x_p \in X_{ij}} [\chi_{pb} - \mu_{ijb}]^2 - \sum_{x_p \in X_i} [\chi_{pb} - \mu_{ib}]^2 - \sum_{x_p \in X_j} [\chi_{pb} - \mu_{jb}]^2 \right) =$$

$$\left(\sum_{x_p \in X_i} [(\chi_{pb} - \mu_{ijb})^2 - (\chi_{pb} - \mu_{ib})^2] + \sum_{x_p \in X_j} [(\chi_{pb} - \mu_{ijb})^2 - (\chi_{pb} - \mu_{jb})^2] \right) =$$

$$\left(\begin{array}{l} \sum_{x_p \in X_i} [\chi_{pb}^2 - 2\chi_{pb}\mu_{ijb} + \mu_{ijb}^2 - \chi_{pb}^2 + 2\chi_{pb}\mu_{ib} - \mu_{ib}^2] + \\ \sum_{x_p \in X_j} [\chi_{pb}^2 - 2\chi_{pb}\mu_{ijb} + \mu_{ijb}^2 - \chi_{pb}^2 + 2\chi_{pb}\mu_{jb} - \mu_{jb}^2] \end{array} \right) =$$

$$\left(\begin{array}{l} -2\mu_{ijb} \sum_{x_p \in X_i} \chi_{pb} + n_i\mu_{ijb}^2 + 2\mu_{ib} \sum_{x_p \in X_i} \chi_{pb} - n_i\mu_{ib}^2 \\ -2\mu_{ijb} \sum_{x_p \in X_j} \chi_{pb} + n_j\mu_{ijb}^2 + 2\mu_{jb} \sum_{x_p \in X_j} \chi_{pb} - n_j\mu_{jb}^2 \end{array} \right) =$$

$$\left(\begin{array}{l} -2n_i\mu_{ib}\mu_{ijb} + n_i\mu_{ijb}^2 + 2n_i\mu_{ib}^2 - n_i\mu_{ib}^2 \\ -2n_j\mu_{jb}\mu_{ijb} + n_j\mu_{ijb}^2 + 2n_j\mu_{jb}^2 - n_j\mu_{jb}^2 \end{array} \right) =$$

$$n_i(\mu_{ib}^2 - 2\mu_{ib}\mu_{ijb} + \mu_{ijb}^2) + n_j(\mu_{jb}^2 - 2\mu_{jb}\mu_{ijb} + \mu_{ijb}^2) =$$

$$n_i(\mu_{ib} - \mu_{ijb})^2 + n_j(\mu_{jb} - \mu_{ijb})^2. \tag{4-8a}$$

where μ_{ijb} is the mean value for the b^{th} spectral band of the mean vector, u_{ij}, of region represented by $X_{ij} = X_i \bigcup X_j$.

Since

$$\mu_{ijb} = \frac{n_i\mu_{ib} + n_j\mu_{jb}}{n_i + n_j}, \tag{4-8b}$$

an alternate form for Equation (4-8a) is:

$\Delta MSE_b(X_i,X_j) =$

$$n_i(\mu_{ib} - \mu_{ijb})^2 + n_j(\mu_{jb} - \mu_{ijb})^2 =$$

$$n_i\mu_{ijb}^2 - 2n_i\mu_{ib}\mu_{ijb} + n_i\mu_{ib}^2 + n_j\mu_{ijb}^2 - 2n_j\mu_{jb}\mu_{ijb} + n_j\mu_{jb}^2 =$$

$$n_i\mu_{ib}^2 + n_j\mu_{jb}^2 - 2(n_i\mu_{ib} + n_j\mu_{jb})\mu_{ijb} + (n_i + n_j)\mu_{ijb}^2$$

$$\frac{1}{(n_i + n_j)}[(n_i + n_j)(n_i\mu_{ib}^2 + n_j\mu_{jb}^2) - 2(n_i\mu_{ib} + n_j\mu_{jb})^2 + (n_i\mu_{ib} + n_j\mu_{jb})^2] =$$

$$\frac{1}{(n_i + n_j)}[(n_i + n_j)(n_i\mu_{ib}^2 + n_j\mu_{jb}^2) - (n_i\mu_{ib} + n_j\mu_{jb})^2] =$$

$$\frac{1}{(n_i + n_j)}\left(n_i^2\mu_{ib}^2 + n_i n_j \mu_{jb}^2 + n_i n_j \mu_{ib}^2 + n_j^2 \mu_{jb}^2 - n_i^2 \mu_{ib}^2 - 2n_i n_j \mu_{ib}\mu_{jb} - n_j^2 \mu_{jb}^2\right) =$$

$$\frac{n_i n_j}{(n_i + n_j)}\left(\mu_{jb}^2 + \mu_{ib}^2 - 2\mu_{ib}\mu_{jb}\right) =$$

$$\frac{n_i n_j}{(n_i + n_j)}\left(\mu_{ib} - \mu_{jb}\right)^2. \qquad (4\text{-}8c)$$

Combining Equations (4-7a) and (4-8c),

$$d_{BSMSE}(X_i, X_j) = \frac{n_i n_j}{(n_i + n_j)}\sum_{b=1}^{B}\left(\mu_{ib} - \mu_{jb}\right)^2. \qquad (4\text{-}9a)$$

Similarly combining Equations (4-7c) and (4-8c),

$$d_{BMMSE}(X_i, X_j) = \frac{n_i n_j}{(n_i + n_j)}\max\left\{\left(\mu_{ib} - \mu_{jb}\right)^2 : b = 1,2,\cdots,B\right\} \qquad (4\text{-}9b)$$

The dimensionality of the d_{BSMSE} and the d_{BMMSE} dissimilarity criteria is equal to the square of the dimensionality of the image pixel values, while the dimensionality of the vector norm based dissimilarity criteria is equal to the dimensionality of the image pixel values. To keep the dissimilarity criteria dimensionalities consistent, HSEG uses the square root of these dissimilarity criteria. The "Square Root of Band Sum Mean Squared Error" criterion is:

$$d_{BSMSE}^{1/2}(X_i, X_j) = \left[\frac{n_i n_j}{(n_i + n_j)}\sum_{b=1}^{B}\left(\mu_{ib} - \mu_{jb}\right)^2\right]^{1/2}, \qquad (4\text{-}10a)$$

and the "Square Root of Band Sum Maximum Squared Error" criterion is:

$$d_{BMMSE}^{1/2}(X_i, X_j) = \left[\frac{n_i n_j}{(n_i + n_j)}\max\left\{\left(\mu_{ib} - \mu_{jb}\right)^2 : b = 1,2,\cdots,B\right\}\right]^{1/2}. \qquad (4\text{-}10b)$$

Dissimilarity criterion 8, the Normalized Vector Distance (NVD), is taken from papers by Baraldi and Parmiggiani [6, 7]. The NVD is based on a combination of a vector modulus measure (such as the 2-norm of the vector) with the previously defined SAM criterion (4-4a). Under this criterion, two vectors are considered to be equal if they have the same modulus (i.e., 2-norm) and the spectral angle between them is zero.

As before, let u_i and u_j be the mean vectors of regions X_i and X_j, respectively. Define

$$\sigma_1(u_i, u_j) = \min\left\{\frac{\|u_i\|_2}{\|u_j\|_2}, \frac{\|u_j\|_2}{\|u_i\|_2}\right\} =$$

$$\min\left\{\frac{\left(\sum_{b=1}^{B}\mu_{ib}^2\right)^{1/2}}{\left(\sum_{b=1}^{B}\mu_{jb}^2\right)^{1/2}},\frac{\left(\sum_{b=1}^{B}\mu_{jb}^2\right)^{1/2}}{\left(\sum_{b=1}^{B}\mu_{ib}^2\right)^{1/2}}\right\}. \tag{4-11a}$$

Note that $0.0 \le \sigma_1(u_i,u_j) \le 1.0$ and 0.0/0.0 is defined to equal 1.0. Here, similar length vectors will have σ_1 close to 1.0 and dissimilar vectors will have σ_1 close to 0.0.

The spectral angle θ between the region mean vectors, u_i and u_j, of regions X_i and X_j was defined earlier in (4-4a). Define

$$\sigma_2(u_i,u_j)=\frac{(\pi/2-\theta(u_i,u_j))}{\pi/2} \tag{4-11b}$$

as the normalized spectral angle between the vectors u_i and u_j. Note that $0.0 \le \sigma_2(u_i,u_j) \le 1.0$ and that similar length vectors will have σ_2 close to 1.0 and dissimilar vectors will have σ_2 close to 0.0. The NVD dissimilarity criterion is then defined as:

$$\mathrm{d}_{NVD}(X_i,X_j)=1.0-\sigma_1(u_i,u_j)\sigma_2(u_i,u_j) \tag{4-11c}$$

Note that $0.0 \le \mathrm{d}_{NVD} \le 1.0$ and that similar length vectors will have d_{NVD} close to 0.0 and dissimilar vectors will have d_{NVD} close to 1.0.

The Entropy criterion, dissimilarity criterion 9, was first defined by Tilton [8]. The basic idea behind the Entropy criterion is to minimize the change of entropy between the existing region mean image and the region mean image created after a pair of regions merge. For the previously defined Spectral Information Divergence criterion, we defined a probability measure for a pixel element by normalizing the pixel element value by the sum of the pixels at that location over all spectral bands. However, for the Entropy criterion, we define a probability measure for a pixel element, χ_{pb}, by normalizing the pixel element value by the sum over all pixels for a particular spectral band over all image pixels:

$$q_b'(x_p)=\frac{\chi_{pb}}{\sum_{p=1}^{N}\chi_{pb}}, \tag{4-12}$$

where $x_p = (\chi_{p1}, \chi_{p2}, \ldots, \chi_{pB})^{\mathrm{T}}$. Then, entropy of band b (out of B spectral bands) of the image X is then given by

$$\mathrm{H}_b(X)=-\sum_{p=1}^{N}q_b'(x_p)\log\left[q_b'(x_p)\right]=-\sum_{p=1}^{N}\left(\frac{\chi_{pb}}{NM_b}\right)\log\left(\frac{\chi_{pb}}{NM_b}\right)=$$

$$-\sum_{p=1}^{N}\left(\frac{\chi_{pb}}{NM_b}\right)\left[\log(\chi_{pb})-\log(NM_b)\right]=\log(NM_b)-\frac{1}{NM_b}\sum_{p=1}^{N}\chi_{pb}\log(\chi_{pb}) \tag{4-13a}$$

where M_b is the mean value of spectral band b over all N image pixels. The total multispectral entropy is taken to be

$$H(X) = \sum_{b=1}^{B} H_b(X) \tag{4-13b}$$

This summation is strictly true only if all spectral bands are uncorrelated, which is generally *not* the case. Notwithstanding this statistical technicality, this summation still leads to a useful dissimilarity criterion for multispectral (and hyperspectral) data.

For a particular pair of regions X_i and X_j, with mean vectors, u_i and u_j, respectively, let $\Delta H(u_i,u_j)$ be the change in $H(X)$ for the multispectral region mean image formed after the pair of regions is merged as compared to the region mean image before the merge:

$$\Delta H(u_i,u_j) = -\frac{1}{N}\sum_{b=1}^{B}\frac{1}{M_b}\left[\sum_{x_p\in X_{ij}}\mu_{ijb}\log(\mu_{ijb}) - \sum_{x_p\in X_i}\mu_{ib}\log(\mu_{ib}) - \sum_{x_p\in X_j}\mu_{jb}\log(\mu_{jb})\right] =$$

$$\frac{1}{N}\sum_{b=1}^{B}\frac{1}{M_b}\left[n_i\mu_{ib}\log(\mu_{ib}) + n_j\mu_{jb}\log(\mu_{jb}) - (n_i+n_j)\mu_{ijb}\log(\mu_{ijb})\right] \tag{4-14a}$$

where the μ_{ijb} are the elements of the mean vector $u_{ij} = (\mu_{ij1}, \mu_{ij2}, \ldots, \mu_{ijB})^T$ and n_i (n_j) is the number of pixels in region X_i (X_j). Noting that the factor N has no effect on dissimilarity comparisons, the Entropy criterion is defined as:

$$d_{ENT}(X_i, X_j) = N\Delta H(u_i,u_j) =$$

$$\sum_{b=1}^{B}\frac{1}{M_b}\left[n_i\mu_{ib}\log(\mu_{ib}) + n_j\mu_{jb}\log(\mu_{jb}) - (n_i+n_j)\mu_{ijb}\log(\mu_{ijb})\right] \tag{4-14b}$$

If the data is normalized so as to have equal mean values across the bands (see the discussion for the *normind* parameter below), the M_b factor can also be dropped:

$$d'_{ENT}(X_i, X_j) = \sum_{b=1}^{B}\left[n_i\mu_{ib}\log(\mu_{ib}) + n_j\mu_{jb}\log(\mu_{jb}) - (n_i+n_j)\mu_{ijb}\log(\mu_{ijb})\right] \tag{4-14c}$$

Note that μ_{ijb} can easily be calculated using (4-8b) above.

Dissimilarity criterion 10 is based on the "SAR Speckle Noise Criterion" from a paper by J.-M. Beaulieu [9].The criterion is:

$$d_{SAR}(X_i, X_j) = \left[\frac{n_i n_j}{(n_i+n_j)}\right]^{1/2}\sum_{b=1}^{B}\frac{|\mu_{ib}-\mu_{jb}|*(n_i+n_j)}{(n_i\mu_{ib}+n_j\mu_{jb})}$$

$$= (n_i n_j (n_i+n_j))^{1/2}\sum_{b=1}^{B}\frac{|\mu_{ib}-\mu_{jb}|}{(n_i\mu_{ib}+n_j\mu_{jb})}, \tag{4-15}$$

NOTE: Other dissimilarity criterion can be included as additional options without changing the nature of the RHSEG implementation.

log (string) Output log file (no default)

At a minimum (for *debug* = 1), the output log file records program parameters and the number of regions and maximum merge ratio value for each level of the region segmentation hierarchy.

The following optional parameters specify the scaling of the input image data:

scale (double) Comma delimited list of input data scale factors (specify one value per band, default = 1.0 for each band)

offset (double) Comma delimited list of input data offset factors (specify one value per band, default = 0.0 for each band)

The optional scale and offset values were added to accommodate the input of MODIS data into RHSEG. The MODIS multispectral data are normally stored in scaled short integer format, with scale and offset factors provided to rescale the data into calibrated reflectance or radiance values. These scale and offset values are used in the following manner to scale the input image data (*input_image*) for each band:

scaled_input_image[*band*] = *scale*[*band*]*(*input_image*[*band*] – *offset*[*band*])

The following parameters specify output files (with default names):

class_labels_map (string) Output region class labels map data file name (default = '*input_image*'_class_labels_map)

The region class labels map at the finest level of segmentation detail (hierarchical level 0). Together with *region_classes* (see below), this forms the main output of RHSEG. Region class label values of "0" correspond to invalid input data values in the input image data. Valid region label values range from 1 through 4,294,967,295. The data is of data type "unsigned int," and the rows and columns (and slices for 3-D) of *class_labels_map* correspond to the rows and columns (and slices for 3-D) of the input image data.

boundary_map (string) Output hierarchical boundary map file name (default = {none})

The hierarchical boundary map is an optional output of RHSEG. The data values of this map are (of type unsigned char (byte)), and the rows and columns (and slices for 3-D) of *boundary_map* correspond to the rows and columns (and slices for 3-D) of the input image data. The data values of the boundary map correspond to the last hierarchical level (plus one) at which the image pixel was last on the boundary of a region.

region_classes (string) Output region classes file name (default = '*input_image*'_region_classes)

The region classes file contains selected information about each region class at each hierarchical level. The information includes the "region merges list" and "region number of pixels" feature values, plus various region features as selected by the *region_sum*, *region_std_dev*, *region_boundary_npix*, *region_threshold*, *region_nb_objects*, and *region_objects_list* parameters (see below).

The region merges list feature consists of the renumberings of the region class labels map required to obtain the region class labels map for the second most detailed level (hierarchical level 1) through the coarsest (last) level of the segmentation hierarchy from the *class_labels_map* (see above). The data is stored as rows of values, with the column location (with counting starting at 1) corresponding to the region class labels value in the *class_labels_map* (the region class labels map at the finest level of detail in the segmentation hierarchy) and the row location corresponding to the segmentation hierarchy level (the l^{th} row contains the renumberings required to obtain the $(l+1)^{th}$ level of the segmentation hierarchy).

The region number of pixels feature consists of the number of pixels in each region class stored as rows of values, with the column location (with counting starting at 1) corresponding to the region class label value and the row location corresponding to the segmentation hierarchy level (with counting starting at 0).

oparam (string) Output parameter file name (default = '*input_image*'.oparam)

The output parameter file contains (in ascii form) all the output parameters from RHSEG. This parameter file is formatted in the same way as the input parameter file for RHSEG and contains most of the same parameters. Additional parameters are the number of hierarchical segmentation levels (*nb_levels*) in the hierarchical segmentation output and the number of regions (*level0_nregions*) in the hierarchical segmentation with the finest segmentation detail. These additional parameter values are required to interpret the *region_classes* output file and the optional *region_objects* output file (see below).

When *spclust_wght* > 0.0, the following optional parameters may be used to output information on the region objects contained in each region class (no defaults, and ignored if *spclust_wght* = 0.0 or if both of these optional parameters are not specified):

object_labels_map (string) Output region object labels map data file name (optional)

The region object labels map at the finest level of segmentation detail (hierarchical level 0). Region object label values of "0" correspond to invalid input data values in the input image data. Valid region label values range from 1 through 4,294,967,295. The data is of data type "unsigned int," and the rows and columns (and slices for 3-D) of *object_labels_map* correspond to the rows and columns (and slices for 3-D) of the input image data.

region_objects (string) Output region objects file name (optional)

The region objects file contains selected information about each region object at each hierarchical level. The information includes the "region merges list" and "region number of pixels" feature values, plus various region features as selected by the *region_sum*, *region_std_dev* and *region_boundary_npix* parameters (see below).

The region merges list feature consists of the renumberings of the region object labels map required to obtain the region object labels map for the second most

detailed level (hierarchical level 1) through the coarsest (last) level of the segmentation hierarchy from the *object_labels_map* (see above). The data is stored as rows of values, with the column location (with counting starting at 1) corresponding to the region object labels value in the *object_labels_map* (the region object labels map at the finest level of detail in the segmentation hierarchy) and the row location corresponding to the segmentation hierarchy level (the l^{th} row contains the renumberings required to obtain the $(l+1)^{th}$ level of the segmentation hierarchy).

The region number of pixels feature consists of the number of pixels in each region object stored as rows of values, with the column location (with counting starting at 1) corresponding to the region object label value and the row location corresponding to the segmentation hierarchy level (with counting starting at 0).

The following parameters select the optional contents of the required output *region_classes* file and the optional output *region_objects* file (above):

region_sum (bool) Region sum feature inclusion flag (*true* (1) or *false* (0), default = *true* if *nbands* < 20, default = *false* otherwise)

When this flag is *true*, the region sum feature values for each spectral band are stored in the *region_classes* file (and *region_objects* file, if specified). When available, the region sum squared values and the sum of the product of the region values times the log of the region values are also stored.

region_std_dev (bool) Region standard deviation inclusion flag (*true* (1) or *false* (0), default = *false*. User provided value ignored and set to *false* if *std_dev_wght* = 0.0)

When this flag is *true*, the region standard deviation feature values are stored in the *region_classes* file (and *region_objects* file, if specified). Here the region standard deviation feature is defined as the maximum over spectral bands of the region mean normalized standard deviation for each region. See the discussion of the *std_dev_wght* parameter (below) for more information on this feature.

region_boundary_npix (bool) Region boundary number of pixels inclusion flag (*true* (1) or *false* (0), default = *false*)

When this flag is *true*, the region number of boundary pixels feature values are stored in the *region_classes* file (and *region_objects* file, if specified).

region_threshold (bool) Inclusion flag for the merge threshold for the most recent merge for each region class (*true* (1) or *false* (0), default = *false*)

When this flag is *true*, the merge threshold for the most recent merge for each region class is stored in the *region_classes* file.

region_nb_objects (bool) Inclusion flag for the number of region objects contained in each region class (*true* (1) or *false* (0), default = *false*. User provided value ignored and set to *false* if *std_dev_wght* = 0.0)

When this flag is *true*, the number of region objects contained in each region class is stored in the *region_classes* file.

region_objects_list (bool) Inclusion flag for the list of the labels of the region objects contained in each region class (*true* (1) or *false* (0), default = *false*. User provided value ignored and set to *false* if *std_dev_wght* = 0.0)

When this flag is *true*, the list of the labels of the region objects contained in each region class is stored in the *region_classes* file.

The following optional parameters are recommended for variation by all users (defaults provided):

conn_type (int) Neighbor connectivity type:

One-dimensional case:

1. "Two Nearest Neighbors,"
2. "Four Nearest Neighbors,"
3. "Six Nearest Neighbors,"
4. "Eight Nearest Neighbors,"

(default: 1. "Two Nearest Neighbors")

based on the following neighborhood chart, where the focal pixel is marked "X":

7	5	3	1	X	2	4	6	8

Using this chart, *n* Nearest Neighbors include pixels 1, 2, … *n*.

Two-dimensional case:

1. "Four Nearest Neighbors,"
2. "Eight Nearest Neighbors,"
3. "Twelve Nearest Neighbors,"
4. "Twenty Nearest Neighbors,"
5. "Twenty-Four Nearest Neighbors,"

(default: 2. "Eight Nearest Neighbors")

based on the following neighborhood chart, where the focal pixel is marked "X":

21	15	11	17	23
13	5	3	7	19
9	1	X	2	10
20	8	4	6	14
24	18	12	16	22

Using this chart, *n* Nearest Neighbors include pixels 1, 2, … *n*.

Three-dimensional case:
1. "Six Nearest Neighbors,"
2. "Eighteen Nearest Neighbors,"
3. "Twenty-Six Nearest Neighbors,"

(default: 3. "Twenty-Six Nearest Neighbors")

based on the following neighborhood chart, where the focal pixel is marked "X":

slice-1

19	11	23
15	5	17
25	13	21

Slice

7	3	9
1	X	2
10	4	8

slice+1

22	14	26
18	6	16
24	12	20

Using this chart, *n* Nearest Neighbors include pixels 1, 2, … *n*.

chk_nregions (unsigned int) Number of region classes at which segmentation hierarchy output is initiated ($2 \leq$ *chk_nregions* < 65535, default = 64 if *hseg_out_nregions* and *hseg_out_thresholds* not specified)

hseg_out_nregions (unsigned int) The set of number of regions at which hierarchical segmentation output are made (a comma delimited list, default = {none})

hseg_out_thresholds (float) The set of merge thresholds at which hierarchical segmentation output are made (a comma delimited list, default = {none})

NOTE: *chk_nregions*, *hseg_out_nregions*, and *hseg_out_thresholds* are mutually exclusive. If more than one of these is specified, the last one specified controls and the previous specifications are ignored. However, *hseg_out_nregions* and *hseg_out_thresholds* may not be specified for *rnb_levels* > 1.

conv_nregions (short unsigned int) Number of regions for final convergence (the iteration at which HSEG or RHSEG is terminated) ($0 <$ *conv_nregions* < 65535, default = 2)

gdissim (boolean) Flag specifying whether or not the global dissimilarity value is output for each level of the output segmentation hierarchy (1 (true) or 0 (false), default = 0)

The dissimilarity criterion utilized is specified by the *dissim_crit* parameter (above). The global dissimilarity is a measure of the quality of the image segmentations based on the global dissimilarity of the region mean image versus the original image data.

The global dissimilarity criteria 1, 2 and 3 are based on vector norms. The global dissimilarity function, based on the vector 1-Norm, for the R region segmentation of the N pixel data set X is given by:

$$\mathrm{D}_{\text{1-Norm}}(X) = \frac{1}{N}\sum_{i=1}^{R}\sum_{x_p \in X_i} \left\| x_p - u_i \right\|_1 . \tag{4-16}$$

where x_p is the p^{th} image pixel and u_i is the region mean vector for region X_i.

The global dissimilarity function, based on the vector 2-Norm, for the R region segmentation of the N pixel data set X is given by:

$$\mathrm{D}_{\text{2-Norm}}(X) = \frac{1}{N}\sum_{i=1}^{R}\sum_{x_p \in X_i} \left\| x_p - u_i \right\|_2 . \tag{4-17}$$

The global dissimilarity function, based on the vector ∞-Norm, for the R region segmentation of the N pixel data set X is given by:

$$\mathrm{D}_{\infty\text{-Norm}}(X) = \frac{1}{N}\sum_{i=1}^{R}\sum_{x_p \in X_i} \left\| x_p - u_i \right\|_\infty . \tag{4-18}$$

The global dissimilarity criterion 4, based on the Spectral Angle Mapper (SAM) criterion introduced previously, is given by:

$$\mathrm{D}_{\text{SAM}}(X) = \frac{1}{N}\sum_{i=1}^{R}\sum_{x_p \in X_i} \theta(x_p, u_i) = \frac{1}{N}\sum_{i=1}^{R}\sum_{x_p \in X_i} \arccos\left(\frac{x_p \circ u_i}{\left\| x_p \right\|_2 \left\| u_i \right\|_2} \right). \tag{4-22}$$

where $\theta(x_p,u_i)$ is the spectral angle between x_p, the p^{th} image pixel, and u_i, the region mean vector for region X_i.

The global dissimilarity criterion 5 is based on the Spectral Information Divergence (SID) criterion introduced previously. The entropy of the p^{th} image pixel, x_p, and the entropy of the region mean vector, u_i, for region X_i are defined as

$$q_b(x_p) = \frac{\chi_{pb}}{\sum_{b=1}^{B}\chi_{pb}} \text{ and } q_b(u_i) = \frac{\mu_{ib}}{\sum_{b=1}^{B}\mu_{ib}},$$

respectively, where $x_p = (\chi_{p1}, \chi_{p2}, \ldots, \chi_{pB})^{\mathrm{T}}$ and $u_i = (\mu_{i1}, \mu_{i2}, \ldots, \mu_{iB})^{\mathrm{T}}$. Then

$$D_{SID}(X) = \frac{1}{N}\sum_{i=1}^{R}\sum_{x_p \in X_i}\sum_{b=1}^{B}\left\{ q_b(x_p)\log\left[\frac{q_b(x_p)}{q_b(u_i)}\right] + q_b(u_i)\log\left[\frac{q_b(u_i)}{q_b(x_p)}\right]\right\}. \qquad (4\text{-}19)$$

The global dissimilarity criteria 6 and 7 are based on the square root of the mean squared error between the region mean image and the original image data. With the mean square error for spectral band *b* as given in (4-6a) and (4-6b), the global dissimilarity criterion "Square Root of Band Sum Mean Squared Error" is:

$$D_{BSMSE}^{1/2}(X) = \left[\sum_{b=1}^{B}\frac{1}{(N-1)}\sum_{i=1}^{R}\sum_{x_p \in X_i}(\chi_{pb} - \mu_{ib})^2\right]^{1/2}$$

$$= \left[\frac{1}{(N-1)}\sum_{i=1}^{R}\sum_{b=1}^{B}\left\{\left(\sum_{x_p \in X_i}\chi_{pb}^2\right) - n_i\mu_{ib}^2\right\}\right]^{1/2}. \qquad (4\text{-}20)$$

Similarly, the global dissimilarity criterion "Square Root of Band Maximum Mean Squared Error" is:

$$D_{BMMSE}^{1/2} = \left[\frac{1}{(N-1)}\sum_{i=1}^{R}\max_{b=1}^{B}\left\{\left(\sum_{x_p \in X_i}\chi_{pb}^2\right) - n_i\mu_{ib}^2\right\}\right]^{1/2}. \qquad (4\text{-}21)$$

Global dissimilarity criterion 8 is based on the Normalized Vector Distance (NVD) introduced previously. Let x_p be the p^{th} image pixel and u_i be the region mean vector for region X_i. Then define

$$\sigma_1(x_p, u_i) = \min\left\{\frac{\|x_p\|_2}{\|u_i\|_2}, \frac{\|u_i\|_2}{\|x_p\|_2}\right\} \quad \text{and} \quad \sigma_2(x_p, u_i) = \frac{(\pi/2 - \theta(x_p, u_i))}{\pi/2}, \qquad (4\text{-}22a)$$

where $\theta(x_p,u_i)$ is the spectral angle between x_p, the p^{th} image pixel, and u_i, the region mean vector for region X_i (see (4-4a) and (4-22)). Then the NVD global dissimilarity criterion is given by

$$D_{NVD}(X) = \frac{1}{N}\sum_{i=1}^{R}\sum_{x_p \in X_i}[1.0 - \sigma_1(x_p, u_i)\sigma_2(x_p, u_i)] \qquad (4\text{-}22b)$$

Global dissimilarity criterion 9 is a measure of how much the entropy of the region mean image differs from the original image data. Using the notation defined previously, the total multispectral entropy of the image, *X*, is given by

$$H(X) = \sum_{b=1}^{B}\left[\log(NM_b) - \frac{1}{NM_b}\sum_{p=1}^{N}\chi_{pb}\log(\chi_{pb})\right] =$$

$$\sum_{b=1}^{B}\left[\log(NM_b) - \frac{1}{NM_b}\sum_{i=1}^{R}\sum_{x_p \in X_i}\chi_{pb}\log(\chi_{pb})\right] \qquad (4\text{-}23a)$$

where the summation over the N image pixels is reordered to sum over the groups of pixels in each of the regions in an R region segmentation. Similarly, the total multispectral entropy of the region mean image of an R region segmentation of the image, X, is given by

$$\mathrm{H}_R(X) = \sum_{b=1}^{B}\left[\log(NM_b) - \frac{1}{NM_b}\sum_{i=1}^{R} n_i \mu_{ib} \log(\mu_{ib})\right] \qquad (4\text{-}23b)$$

The increase in image entropy of the R region mean image over that of the original data is then $\mathrm{D}_{ENT}(X) = \mathrm{H}_R(X) - \mathrm{H}(X)$, or (after changing the order of summation)

$$\mathrm{D}_{ENT}(X) = \frac{1}{N}\sum_{i=1}^{R}\left[\sum_{b=1}^{B}\left\{\frac{1}{M_b}\sum_{x_p \in X_i}\left[\chi_{pb}\log(\chi_{pb})\right] - n_i \mu_{ib}\log(\mu_{ib})\right\}\right]. \qquad (4\text{-}24)$$

The global dissimilarity criterion 10 is based on the SAR Speckle Noise criterion. The global dissimilarity function, based on the SAR Speckle Noise criterion, for the R region segmentation of the data set X is given by:

$$\mathrm{D}_{\mathrm{SAR}}(X) = \sum_{i=1}^{R}\left[\left(n_i n_j (n_i + n_j)\right)^{1/2} \sum_{x_p \in X_i}\sum_{b=1}^{B}\frac{\left|\chi_{pb} - \mu_{jb}\right|}{\left(\chi_{pb} + n_j \mu_{jb}\right)}\right]. \qquad (4\text{-}25)$$

where x_p is the p^{th} image pixel and u_i is the region mean vector for region X_i.

The default values should be used for the following optional parameters, except in special circumstances (defaults provided):

debug (int) Debug option (*debug* ≥ 0, default = 1)

normind (short unsigned int) Image normalization type

1. "No Normalization,"
2. "Normalize Across Bands,"
3. "Normalize Bands Separately"

(default: 2. "Normalize Across Bands")

Let χ_{pb} be the original value for the p^{th} pixel (out of N pixels) in the b^{th} band (out of B bands). The sample mean and sample variance of the b^{th} band are

$$\mu_b = \frac{1}{N}\sum_{p=1}^{N}\chi_{pb} \text{ and } \sigma_b^2 = \frac{1}{N-1}\sum_{p=1}^{N}\left(\chi_{pb} - \mu_b\right)^2, \qquad (4\text{-}26)$$

respectively. The following transformation of the data, χ_{pb}, will produce image data, ξ_{pb}, with mean, M, and standard deviation, Σ:

$$\xi_{pb} = \left[\frac{\Sigma}{\sigma_b}\left(\chi_{pb} - \mu_b\right)\right] + \mathrm{M} = \Sigma_b'\left(\chi_{pb} - \mathrm{M}_b'\right) \qquad (4\text{-}27a)$$

where

$$\Sigma_b^{'} = \frac{\Sigma}{\sigma_b} \text{ and } \mathrm{M}_b^{'} = \mu_b - \mathrm{M}\frac{\sigma_b}{\Sigma}. \quad \text{(4-27b)}$$

For convenience, for most dissimilarity criteria, the data is normalized by default such that $\Sigma^2(=\Sigma)=1$, and M=0. However the Spectral Angle Mapper, Spectral Information Divergence, Normalized Vector Distance and Entropy assume that all data values are nonnegative. Moreover, to avoid the singularity at log(0.0) for the Spectral Information Divergence and Entropy criteria, all data values should be strictly positive (i.e., all greater than zero) in these cases. Due to these considerations, the default value of M is set such that the overall normalized minimum value is 0.0 for the Spectral Angle Mapper and Normalized Vector Distance criteria, and the default value of M is be set such that the overall normalized minimum value is 1.0 for the Spectral Information Divergence and Entropy criteria.

As written above, the normalization is applied to each spectral band separately. It can also be defined to apply equally across all spectral bands. For this case, use $\sigma = \max\{\sigma_b : b = 1,2,\cdots,B\}$ in (4-27a) and (4-27b). However, this choice of normalization will produce the same hierarchical segmentation result as no normalization at all.

init_threshold (float) Threshold for initial fast region merging by a region oriented first merge process adapted from an algorithm proposed by Muerle and Allen [10]. (default = 0.0)

In this region scan version of first merge region growing, unmerged pixels are visited in random order and designated as a new single pixel region. This new region is grown by adding individual (unmerged) neighboring image pixels that are similar enough to the growing region. After no more pixels can be added to a particular region, a region is similarly grown from the next randomly selected unmerged pixel. This process continues until no unvisited or unmerged pixels remain.

The seminal first merge region growing approach of Muerle and Allen [10], hereafter called MARG, utilizes a left to right, top to bottom scan to select the next unmerged pixel from which to start growing a region. However, in the adaptation of their algorithm that is utilized for initialization of HSEG and RHSEG, a random scan order is used to select the next pixel. The adaptation of MARG utilized herein is as follows:

1) Give all image pixels, x_p, in image X (p = 1 to N_P) region label 0, and compute a random ordering, p' = Rand(p), over the N_P pixels. Set T as the value of the merge threshold, p = 0, r = 0, and continue to step 2.

2) Set $p = p + 1$. If $p > N_P$, exit (the segmentation result contains $N_R = r$ regions). Otherwise, continue to step 3.

3) Select image pixel $x_{p'}$, where p' = Rand(p). If the image pixel $x_{p'}$ has already been merged into a region (i.e, it has a region label other than 0),

return to step 2. Otherwise, set $r = r + 1$ and create region object o_r with region label r, feature values computed from pixel $x_{p'}$, and a pixel neighbor list specifying the pixel index of unmerged neighboring pixels (i.e., neighboring pixels having region label 0). If the new region object o_r has no unmerged neighboring pixels, give region label r to pixel $x_{p'}$ and return to step 2. Otherwise, randomly shuffle the ordering of the pixel indices in the pixel neighbor list, give region label r to pixel $x_{p'}$, and continue to step 4.

4) Successively compute the dissimilarity, $d(o_r,x_k)$, between region object o_r and the unmerged neighboring pixels, x_k, in the randomly shuffled pixel neighbor list until a pixel is found that has $d(o_r,x_k) \le T$, or all neighboring pixels are checked. If $d(o_r,x_k) > T$ for all unmerged neighboring pixels, x_k, return to step 2. Otherwise, continue to step 5.

5) Merge the first found neighboring unmerged neighboring pixel, x_k, with $d(o_r,x_k) \le T$ into region object o_r by updating the region feature values and neighbor pixel index list for region object o_r, and giving pixel x_k region label r. If the new region object o_r has no neighboring unmerge pixels, return to step 2. Otherwise, randomly shuffle the ordering of the pixel indices in the new pixel neighbor list, and return to step 4.

Besides the random order of seed pixel selection, the main difference between the above algorithm and Muerle and Allen's approach is in step 5 where the first found unmerged neighboring pixel with dissimilarity less than or equal to T is selected for merging from a randomly shuffled list of unmerged neighboring pixels. It is not clear what scanning order Muerle and Allen used to select this next found unmerged neighboring pixel, but it is unlikely that it was a random ordering. Another difference is that Muerle and Allen initially aggregate the image pixels into blocks sized anywhere from 2×2 to 8×8 before performing region growing, whereas the initial regions in the above algorithm are single pixel in size.

std_dev_wght (float) Weight for standard deviation spatial feature (*std_dev_wght* ≥ 0.0, default = 0.0)

The parameter *std_dev_wght* sets the weighting for the standard deviation feature. The mean normalized standard deviation is used here instead of the usual standard deviation feature. If D is the dissimilarity function value before combination with the spatial feature value, the combined dissimilarity function value (comparing regions i and j), D^c, is:

$$D^c = D * \left[1.0 + \frac{\left| sdf_i - sdf_j \right|}{\left(sdf_i + sdf_j \right)} * std\_dev\_wght \right], \qquad (4\text{-}28)$$

where sdf_i and sdf_j are the standard deviation feature values for regions i and j, respectively.

The standard deviation feature employed here is the spectral band maximum, mean normalized region standard deviation. For regions consisting of 2 or more pixels, the mean normalized region standard deviation for spectral band *b* of region *i* is:

$$\sigma_{ib} = \frac{\sqrt{\frac{1}{n_i - 1}\sum_{x_p \in X_i}\left(\chi_{pb} - \mu_{ib}\right)^2}}{\mu_{ib}} = \frac{\sqrt{\frac{1}{n_i - 1}\left[\sum_{x_p \in X_i}\left(\chi_{pb}\right)^2 - n_i\left(\mu_{ib}\right)^2\right]}}{\mu_{ib}}, \qquad (4\text{-}29a)$$

where n_i is the number of pixels in the region and μ_{ib} is the region mean for spectral band *b* of region *i*:

$$\mu_{ib} = \frac{1}{n_i}\sum_{x_p \in \chi_i}\chi_{pb}\,.$$

The standard deviation feature value for region *i* is then defined as:

$$sdf_i = \sigma_i = \max\left\{\sigma_{ib} : b = 1,2,\ldots,B\right\} \qquad (4\text{-}29b)$$

where *B* is the number of spectral bands.

The region standard deviation is not defined for regions consisting of only one pixel. Further, the mean normalize region standard deviation as calculated by equation (4-17a) can only be considered a rough estimate for small regions (say, regions less than 9 pixels in size). Thus, if one of the regions being compared consists of less than 9 pixels, the *std_dev_wght* factor is modified by a *std_dev_factor* as follows:

*std_dev_wght' = std_dev_factor*std_dev_wght,* (4-30a)

where

std_dev_factor =(*min_npix*-1.0)/8.0, (4-30b)

and *min_npix* is the number of pixels in the smaller of the two regions being compared. Note that for *min_npix* = 1, *std_dev_factor* = 0.0. Thus, *std_dev_factor* serves to gradually phase in the standard deviation spatial feature as the regions get larger.

split_pixels_factor (float) Pixel splitting factor. A pixel will be split out from its current region if it is this factor more similar to another region than it is to its current region. (0.0 ≤= *split_pixels_factor*, default = 1.4. No pixel splitting is performed if *split_pixels_factor* < 1.0.)

For each region with a non-empty "candidate region label" set, compute the dissimilarity of each pixel in that region to its current region (*own_region_dissim*) and to each region in the region's "candidate region label" set (*other_region_dissim*). If a pixel is found to have *own_region_dissim* > *split_pixels_factor*other_region_dissim*, the pixel is split out from its current

region. NOTE: The **lower** the value of *split_pixels_factor*, the **more** pixels are split out from their regions (for *split_pixels_factor* ≥ 1.0).

seam_threshold_factor (float) This threshold factor is used in determining whether a region found across a processing window seam is to be considered in determining whether a pixel is to be split out of its current region. (1.0 <= *seam_threshold_factor*, default = 1.3. If *threshold_factor* = 1.0, no regions are selected by this method)

During the processing window elimination process, a "candidate region label" set is accumulated for use in considering whether or not a pixel is to be split out of its current region. Consider the data points that are in the pairs of rows and columns along the seam between the data quadrants reassembled in step 2 of the RHSEG algorithm. For each of these pixels calculate the dissimilarity between the pixel and its current region (*own_region_dissim*), and calculate the dissimilarity between the pixel and the region of the pixel across the seam (*other_region_dissim*). If *own_region_dissim* > *seam_threshold_factor** *other_region_dissim*, add the region label of the region of the pixel across the seam to the "candidate region label" set of the region the pixel belongs to.

NOTE: The **lower** the value of *seam_threshold_factor*, the **more** regions are included in the "candidate region label" set.

region_threshold_factor (float) This threshold factor is used in determining which regions are to be considered in determining whether a pixel is to be split out of its current region. (0.0 < *threshold_factor*, default = 0.0. If *region_threshold_factor* = 0.0, no regions are selected by this method)

During the processing window elimination process, a "candidate region label" set is accumulated for use in considering whether or not a pixel is to be split out of its current region. Compare each region to every other region. If the dissimilarity between a pair of regions is less than *region_threshold_factor** *max_threshold*, add each region label to the "candidate region label" set for the other region. NOTE: *max_threshold* is the maximum merging threshold encountered in the previous merging iterations. This factor is ignored for *spclust_wght* = 0.0. Also, the **higher** the value of *region_threshold_factor*, the **more** regions are included in the "candidate region label" set.

min_npixels (unsigned int) For regions smaller than this minimum number of pixels, the dissimilarity function is adjusted to favor merging. (0 <= *min_npixels*; for *dissim_crit* = 6, 7, 9 or 10: default = 0; for *dissim_crit* = 1, 2, 3, 4, 5 or 8: default = 200)

The value of *min_npixels* is used to calculate a merge acceleration factor, *factor*, which is multiplied times the dissimilarity criterion value. For two regions of size

(number of pixels) n_1 and n_2, let N_{min} = *min_npixels* and let N_i = min(n_i, N_{min}) for *i* = 1,2. Then

$$factor = \frac{\left(\frac{N_1 * N_2}{(N_1 + N_2)}\right)^{1/2}}{\left(\frac{N_{min} * N_{min}}{(N_{min} + N_{min})}\right)^{1/2}} = \left(\frac{2 * N_1 * N_2}{N_{min} * (N_1 + N_2)}\right)^{1/2}. \quad (4\text{-}31)$$

Note that if both n_1 and $n_2 \geq N_{min}$ (=*min_npixels*), *factor* = 1.0.

rnb_levels (short unsigned int) Number of recursive levels

(1 ≤ *rnb_levels* < 255, default calculated)

The number of recursive levels. The default is calculated such that the number of data points in the subsections of data processed at recursion level *rnb_levels* is no more than 4000 data points. The number of columns, rows and slices at recursion level *rnb_levels* is *sub_ncols* = *ncols*/$2^{rnb\_levels-1}$, *sub_nrows* = max(1, *nrows*/$2^{rnb\_levels-1}$, and *sub_nslices* = max(1, *nslices*/$2^{rnb\_levels-1}$).

ionb_levels (short unsigned int) Recursive level at which data I/O is performed

(1 ≤ *ionb_levels* ≤ *rnb_levels*, default calculated)

The recursive level at which data I/O is performed and pixel oriented data is maintained (sequential version only). Temporary data files are used to store the pixel oriented data for each section of data that the image is divided into at this recursive levels. The default value is *ionb_levels* = 1, unless the number of data points exceeds 9,437,184 (=262,144*36), where the default is calculated such that the number of data points in the subsections of data processed at recursion level *ionb_levels* is no more than 262,144 (=512^2) data points. The number of columns, rows and slices at recursion level *ionb_levels* is *ionb_ncols* = *ncols*/$2^{ionb\_levels-1}$, *ionb_nrows* = max(1, *nrows*/$2^{ionb\_levels-1}$), and *ionb_nslices* = max(1, *nslices*/$2^{ionb\_levels-1}$).

Important note: RHSEG uses environmental variables to determine in what directory the temporary files should be stored in. RHSEG first looks for the TMP environmental variable, and if this does not exist, it looks for the TEMP environmental variable, and if this does not exist, it looks for the TMPDIR environmental variable, and if this does not exist, it assumes the temporary directory is /tmp. To be sure RHSEG works as it should you should, set one of the listed environmental variables to a directory that has sufficient disk space to hold the temporary files. The space required will vary with data set characteristics and parameter settings. You should monitor the free space available in the temporary directory during your initial runs if your image contains more than 9,437,184 pixels (with the default parameter settings).

min_nregions (unsigned int) Number of regions for convergence in recursive stages
(0 < *min_nregions* < 65,535, default calculated)

If not specified, the default is calculated to be *min_nregions* = *sub_ncols***sub_nrows***sub_nslices* / 2^{N_D} , where N_D is the number of spatial dimensions (for *sub_ncols*, *sub_nrows* and *sub_nslices* see the *rnb_levels* parameter).

spclust_start (short unsigned int) Number of regions at and below which spectral clustering is utilized. Otherwise, the spectral clustering step is skipped.
(0 ≤ *spectral_start* < 65,535, default calculated)

The default for *spclust_start* is calculated such that spectral clustering is utilized only part of the time when *spclust_wght* > 0.0. If *spclust_wght* = 0.0, *spclust_start* = 0. Otherwise, *spclust_start* = *spclust_wght**(*max_nregions-min_nregions*) + *min_negions*, where *max_nregions* = max(2^{N_D} **min_negions*, *sub_ncols** *sub_nrows***sub_nslices*).

Guidance on HSEG/RHSEG Program Parameter Settings

The parameters that have the most effect on the nature of the segmentation results are *spclust_wght*, *dissim_crit* and *chk_nregions*. The default values are recommended for the other optional parameters for routine use of HSEG and RHSEG, with the exception that specification of the output file name parameter *boundary_map* is also recommended. Of course, if some input data elements are invalid, the some method of data masking should also be employed.

The following paragraphs give some guidance on the setting of the *spclust_wght*, *dissim_crit* and *chk_nregions* parameters:

spclust_wght: The user may want to vary the value of *spclust_wght* to modify the overall nature of the segmentation results. For *spclust_wght* = 0.0, you will obtain relatively coherent closed connected regions. For *spclust_wght* = 1.0, you will obtain relatively variated regions consisting of possibly several spatially disjoint subsections. For other values of *spclust_wght* you will obtain results intermediate the *spclust_wght* = 0.0 and *spclust_wght* = 1.0 results.

dissim_crit: The user may also want to vary the value of *dissim_crit* to modify the overall nature of the segmentation results. The different dissimilarity criterion will result in different merge ordering.

chk_nregions: The user may want to vary the value of *chk_nregions* to vary the level of segmentation detail in the most detailed level of the segmentation hierarchy. Higher values will increase the detail (the segmentation will have more regions) and lower values will decrease the detail (the segmentation will have fewer regions) and the most detailed level of the segmentation hierarchy.

Varying the other optional parameter values away from the default values requires a thorough understanding of the inner workings of the HSEG and RHSEG programs.

References

[1] F. A. Kruse, A. B. Lefkoff, J. W. Boardman, K. B. Heidebrecht, A. T. Shapiro, P. J. Barloon, and A. F. H. Goetz, "The Spectral Image Processing System (SIPS) – Interactive Visualization and Analysis of Imaging Spectrometer Data," *Remote Sensing of Environment*, Vol. 44, Nos. 2-3, pp. 145-163, May-June 1993.

[2] Chein-I Chang, "An Information-Theoretic Approach to Spectral Variability, Similarity, and Discrimination for Hyperspectral Image Analysis," *IEEE Transactions on Information Theory*, Vol. 46, No. 5, pp.1927-1932, August 2000.

[3] J. C. Tilton,W. T. Lawrence, and A. J. Plaza, "Utilizing Hierarchical Segmentation to Generate Water and Snow Masks to Facilitate Monitoring Change with Remotely Sensed Image Data," *GIScience and Remote Sensing*, Vol. 43, No. 1, pp. 39-66, 2006.

[4] Chein-I Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*, Kluwer Academic/Plenum Publishers: New York, 2003.

[5] Peter J. Bickel and Kjell A. Doksum, *Mathematical Statistics: Basic Ideas and Selected Topics*, Holden-Dya, Inc.: San Francisco, 1977.

[6] A. Baraldi and F. Parmiggiani, "A Neural Network for Unsupervised Categorization of Multivalued Input Patterns: An Application to Satellite Image Clustering," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 33, No. 2, pp. 305-316, March 1995.

[7] A. Baraldi and F. Parmiggiani, "Single Linkage Region Growing Algorithms Based on the Vector Degree of Match," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 34, No. 1, pp.137-147, January 1996.

[8] J. C. Tilton, "Experiences using TAE-Plus Command Language for an Image Segmentation Program Interface," *Proceedings of the TAE Ninth Users' Conference*, New Carrollton, MD, pp. 297-312, Nov. 5-7, 1991.

[9] J.-M. Beaulieu, "Utilization of contour criteria in micro-segmentation of SAR images," *International Journal of Remote Sensing*, Vol. 25, No. 17, pp. 3497-3512, Sept. 10, 2004.

[10] J. L. Muerle, D. C. Allen, "Experimental Evaluation of Techniques for Automatic Segmentation of Objects in a Complex Scene," in G. C. Cheng, et al. (Eds.), *Pictorial Pattern Recognition*, Thompson, Washington, DC, pp. 3-13, 1968.

Chapter 5: HSEGViewer Tutorial

Overview

This chapter provides a tutorial on the HSEGViewer program.

HSEGViewer Tutorial

The demonstration version of RHSEG includes a sample data set which is by default installed in the C:\Program Files\RHSEG\Sample Data directory. To provide inputs for the HSEGViewer program, run the RHSEG program on the "girl.bmp" sample image provided, using the provided "rhseg.params" input parameter file:

rhseg rhseg.params

With the specified set of parameters, the rhseg program should take a little over 1 minute to run on a 2 GHz clock machine. When rhseg completes processing, you may run HSEGViewer by entering the command:

hsegviewer

You may also run this version of HSEGViewer from "RHSEG" group in "Run Program" in the "start" menu in Windows. If you like, you can create a shortcut for this program and place it on your desktop.

The "Hierarchical Segmentation Results Viewer Parameter Input" GUI panel will now appear. You will need to enter the HSEG/RHSEG output parameter file through this panel. The easiest way to do this is to click on the file input box under the label "Input HSEG Parameter File (oparam) for Input to HSEGViewer," and then select the "girl.oparam" file through the file selector. You will then have to enter the appropriate values for Red, Green and Blue Display Bands. Since the girl image is an RGB image, enter these values by typing the number 0 in the box to the right of the "Red Display Band" label, the number 1 in the box to the right of the "Green Display Band" label, and the number 2 in the box to the right of the "Blue Display Band" label.

(NOTE: If you are viewing a single-band image, enter the number 0 for all three "Display Band" values – Red, Green and Blue.)

At this point you may also specify if you want to view the region class map in pseudo color or in grey scale by selecting either "Display Region Classes in Psuedo Color" or "Display Region Classes in Grey Scale" from a "Combo Box." For this tutorial, use the default "Display Region Classes in Psuedo Color."

Now you are ready to run the HSEGViewer program. Click on the "Program Actions" menu button at the top left of the panel and select "Run Program." The main "Hierarchical Segmentation Results Viewer" panel will then appear. You may resize this panel as desired.

You may also run HSEGViewer by entering the command (using the provided "hsegviewer.params" file):

hsegviewer hsegviewer.params

When running HSEGViewer in this manner, the main "Hierarchical Segmentation Results Viewer" panel will then appear without further user input.

The main panel holds several buttons and value entry fields with a large table at the bottom. Click on the "RGB Image" button. You will see an RGB rendition of the girl image, with a main viewing panel and a reduced sized image in the upper left. For large images, this reduced sized image will help you navigate to desired locations in your large image. As with all other HSEGViewer panels, this panel has an "Actions" menu button in the upper left corner. Click on this menu button and select "Zoom In." The image data in the "RGB Image" panel will now be displayed zoomed by a factor of two. You may view the entire image by resizing the panel. Return to the original display resolution by selecting "Zoom Out" from the "Actions" menu.

Now click on the "Current Region Labels" button on the main panel. You will now see the "Current Region Labels" display panel, which is initially blank. Now enter the value "1" in the text field to the right of the label "Select Pixels with Segmentation Level 0 Region Class Label" (you need to press the "Enter" key after typing "1" in the text field). This will cause region class 1 at hierarchical level 0 to be highlighted in white in the "Current Region Labels" panel. Region class 1 consists of several dark (mainly shadowed) areas. You may highlight any specific region class in this way by entering in the region class label value in this text box.

You can also highlight other region classes by clicking on any pixel in the RGB Image panel, and then clicking the "Select Pixels with Segmentation Level 0 Region Class Label" button on the main panel. Use this facility now by clicking on a bright yellow pixel in the yellow flower and then clicking on "Select Region Class at Location of Last Left Mouse Click" button on the main panel. If you clicked on the pixel I clicked on (at column 170 and row 235), bright yellow portions of the yellow flower will be highlighted. Looking at the table on bottom portion of the main panel, you should see region class 63 listed with 171 pixels at hierarchical level 0. We can explore how this region class changes at coarser levels of the segmentation hierarchy by clicking on the "Select Next Coarser Segmentation" button on the main panel. Click on this button once now.

You should now see that a larger portion of the yellow flower is added to the region at hierarchical level three (you can tell that this region at hierarchical level three is being highlighted by noting that the value "3" is being displayed in the text box between the "Select Next Finer Segmentation" and "Select Next Coarser Segmentation" buttons). Click on the "Select Next Coarser Segmentation" button once again, and you will see that the pink flower is added to the yellow flower region at hierarchical level five. Clicking on the "Select Next Coarser Segmentation" button a third time adds a number of non-flower pixels, including areas in the window and the girl's hair at hierarchical level eight.

You can also highlight region objects in the same way. Enter the value "1" in the text field to the right of the label "Select Pixels with Segmentation Level 0 Region Object Label." This will cause region object 1 at hierarchical level 0 to be highlighted in white in the "Current Region Labels" panel. Region object 1 is a single dark region in the lower left corner of the image. You may highlight any specific region object in this way by entering in the region object label value in this text box.

You can also highlight other region objects by clicking on any pixel in the RGB Image panel, and then clicking the "Select Pixels with Segmentation Level 0 Region Object Label" button on the main panel. Use this facility now by clicking on a bright yellow pixel in the yellow flower and then clicking on "Select Region Object at Location of Last Left Mouse Click" button on the main panel. If you clicked on the pixel I clicked on (at column 170 and row 235), a small bright yellow portion of the yellow flower will be highlighted. Looking at the table on bottom portion of the main panel, you should see region object 4265 listed with 46 pixels at hierarchical level 0.

We can explore how this region object changes at coarser levels of the segmentation hierarchy by clicking on the "Select Next Coarser Segmentation" button on the main panel. Click on this button once now. You should now see that more of the yellow flower is added to the region object at hierarchical level three. Click on the "Select Next Coarser Segmentation" button once again, and you will see that the pink flower is added to the yellow flower region at hierarchical level five. Clicking on the "Select Next Coarser Segmentation" button a third time adds only five more flower pixels at hierarchical level eight. However, clicking on the "Select Next Coarser Segmentation" button a fourth time adds all of the rest of the yellow flower to the region at hierarchical level eleven.

Let's go ahead and label this area "yellow flower" by clicking on the "Label Region" button in the upper right corner of the main "Hierarchical Segmentation Results Viewer" panel (we will correct the mislabeling of the pink flower in the next step).

You will now see the "Label Region Panel." With this panel you can label a highlighted region with a desired color, and associate that color with a text label. You can even modify one of the pre-configured colors by clicking on one of the colors. Let's go ahead and do that by clicking on the bright yellow color button labeled "60:" towards the bottom right of the panel.

You will now see a "Pick a color" panel. For example, you can select a different shade of yellow by clicking somewhere on the triangle in the left part of the panel. Alternatively you can provide specific HSB or RGB values in the provide text fields. Let's change the Blue value to 100 (to lighten the color a bit). Save this new color and exit this panel by clicking on the "OK" button.

To label the highlighted area with your chosen color type in a label, such as "yellow flower," in the text box to the right of your chosen color. Hitting the "Enter" key while in that text box will record your text label and label the highlighted region with your chosen color. Now close the "Label Region Panel" by clicking on the "X" at the upper left corner of the panel or by selecting "Close" from the "Action" menu.

Now go the RGB Image panel and select a pixel in the pink flower for highlighting. Do this by performing a left mouse click in the middle of the pink flower in the "RBG Image" and click on the "Select Region Class at the Location of Last Left Mouse Click" button on the main panel. You should now have either region 59 or region 62 highlighted. Looking at the bottom of the table on the main panel, you should see either region 59 listed with 120 pixels at hierarchical level 0 or region 62 listed with 120 pixels at hierarchical level 0. Regions 59 and 62 combine at hierarchical level three to nicely cover the pink flower. (Press the "Select Next Coarser Segmentation button once to get to

hierarchical level three.) Give this region a pink color and the label "pink flower" using the "Label Region Panel."

Now select an area on the girl's red shawl for highlighting. Do this by clicking on a pixel in the girl's shawl and then clicking on the "Select Region Class at the Location of Last Left Mouse Click" button. If you clicked on the pixel I clicked on (at column 90 and row 200), you should see that portions of the red shawl, plus part of the girl's red lips are highlighted. Click on the "Select Next Coarser Segmentation" button, and you will see that portions of the red flower are added at hierarchical level three Clicking on the "Select Next Coarser Segmentation" button again adds more red shawl and lip pixels at hierarchical level four. Clicking this button a third time adds most of the rest of the red shawl to the highlighted area at hierarchical level seven.

We can separate the girl's red lips from the rest of the highlighted region by doing the following. Select "Circle Region of Interest" from the "Program Control" menu of the "Current Region Labels" panel, and draw a line surrounding the girl's lips on the panel. Now only the girl's lips will be highlighted. Use the "Label Region" panel to label this area "girl's lips" with a shade of red.

Now perform a left mouse click in the middle of the green colored area on the girl's right shoulder (which is to the left in the image) in the "RBG Image" and click on the button "Select Region Class at the Location of Last Left Mouse Click." If you clicked on the pixel I clicked on (at column 45 and row 235), you will now see portions of the girl's green blouse on both shoulders highlighted in white. Looking at the bottom of the table on the main panel, you should see that this is region class 24 consisting of 897 pixels. (If statistics for another region are displayed at the bottom of the table, you can select region class 24 by entering the number "24" in the text box to the right of the "Select Pixels with Segmentation Level 0 Region Class Label" label and pressing the "Enter" button on your keyboard.) Now click on the "Select Next Coarser Segmentation" button. You will see that region class 24 has grown to 1402 pixels at hierarchical level one with the addition of more pixels from the girl's green blouse. When you click on the "Select Next Coarser Segmentation" button one more time you will see that region class 24 merges into region class 25 at hierarchical level four with 4823 pixels with the addition of background pixels to the region. Click on the "Select Next Finer Segmentation" button to display region class 24 at hierarchical level three.

Let's label the currently highlighted portion of the girl's green blouse shirt by clicking on the "Label Region" button. In the "Label Region Panel" that now appears, type "green blouse" in the text box to the right of dark green color button towards the lower left of the panel and then hit "Enter" on your keyboard. You will now see this region colored dark green in the "Current Class Labels Image" panel. You can now close the "Label Region Panel."

We can include more of the girl's green blouse in this region by selecting a pixel in the dark green area on the girl's left shoulder (to the right in the image). This time select the region object by clicking on the button "Select Region Object at Location of the Last Mouse Click." If you clicked on the pixel I clicked on (at column 200 and row 220), you should see region object 436 highlighted, which contains 853 pixels at hierarchical level zero. Clicking on "Select Next Coarser Segmentation" you will see that region object 436

merges into region object 292 at hierarchical level one. This 1357 pixel region adds more of the green blouse to the region at hierarchical level five. Clicking twice more on the "Select Next Coarser Segmentation" button again adds some of the girl's red shawl to the region object at hierarchical level seven. Go back to hierarchical level six (which displays the same region object as at hierarchical level five) by clicking on the "Select Next Finer Segmentation" button. Add this region to the "green blouse" region by clicking on the "Label Region" button, placing the cursor in the text box where you previously entered "green blouse" and pressing the "Enter" button on your keyboard. Close the "Label Region" panel again.

You have now exercised most of the features of HSEGViewer for labeling an image. However, there are other features of HSEGViewer we have not visited yet. The "Initial Segmentation Level" text box allows you to set the initial segmentation level that is displayed (defaulted to 0) when you select a new region class or object for highlighting. The "Refocus on Selected Region" button centers all image panels on the pixel selected with the "Select Region Class at Location of Last Left Mouse Click" button or "Select Region Object at Location of Last Left Mouse Click" button. The text box between the "Select Next Finer Segmentation" and "Select Next Coarser Segmentation" buttons not only displays the currently highlighted hierarchical segmentation level, but entering a valid hierarchical segmentation level into this text box will jump you to that hierarchical level.

We now come to the set of buttons under the "Display Options" label. We have already visited the "RGB Image" and "Current Class Labels" buttons. The "Segmentation Classes Slice" button provides a pseudo colored rendition of the region class segmentation at the currently selected hierarchical level. Similarly, the "Segmentation Objects Slice" button provides a pseudo colored rendition of the region object segmentation at the currently selected hierarchical level. The "Region Mean Image" button provides a view of the region mean image. The "Hierarchical Boundary Map" button provides a boundary map of the image segmentation, where the darker image boundaries correspond to boundaries that persist up to the highest hierarchical levels, while the lighter image boundaries correspond to boundaries that disappear at lower hierarchical levels. Finally, the "Region Class (Object) Boundary Pixel Ratio Image" displays this ratio for each region class (object).

At the bottom of the display panels the cursor location (column, row) is displayed. The pixel value is also displayed on the "Segmentation Slice View Image," "Current Class Labels Image," and "Hierarchical Boundary Map" display panels. The pixel value on the "Hierarchical Boundary Map" display panel corresponds to the last hierarchical level at which the boundary still exists.

You may save whatever is displayed in any of the image display panels (except for the reference file displays) to a PNG format file by selecting "Save PNG Image" from the "Actions" menu. You will be prompted to specify an output file name with a file chooser.

Finally, the large table at the bottom of the HSEGViewer main panel gives the available information about the selected region, at all hierarchical levels.

To exit the HSEGViewer program, click on the "Program Action" menu on the HSEGViewer main panel and select "Quit" from the menu. (You could also click on the red X in the top right corner of the panel.)

You can exit HSEGViewer and restart it where you left off by renaming the "label_out.raw" and "ascii_out.txt" (default names) files (I often use the file names "label_in.raw" and "ascii_in.txt"), and selecting them as the "Input Class Label Map File" and the "Input ASCII Class Labels Name File," respectively, in the parameter input file.

If you have reference files (such as ground truth) in "PNG" format files, you can use them as a reference files (Input Reference File 1 or Input Reference File 2) by specifying them as "Input Reference" files in the parameter input panel on startup.

Notes on viewing 3-D data with HSEGViewer

HSEGViewer cannot currently render 3-D data in three dimensions. One can display and interact only with selected 2-D planes of 3-D hierarchical segmentation results.

Assume that you have a single band 3-D image with 256 columns, 256 rows and 172 slices (this corresponds to an actual 3-D brain scan image that has been processed with rhseg_3d). By default, HSEGViewer looks at the 256 column by 256 row 2-D image plane at the 0th slice. You can change the slice viewed by changing the value in the text box to the right of the label "For 3-D data, view the 2-D Representation of ". For example, you can change from viewing the 0th slice to viewing the 86th slice.

Once you specify the desired slice for viewing, click on the "Program Actions" pull-down menu at the top left of the panel and select "Run Program." You can now do everything you learned in the 2-D tutorial on this 2-D plane of the 3-D image data.

To view from a different perspective you can select "Quit Program" from the "Program Control" pull-down menu at the top left of the Viewer panel and then select "OK" on the "Confirm Quit" panel. The "Parameter Input" panel then reappears. To the right of the Label "For 3-D data, view the 2-D Representation of" you can select "row" or "column" instead of "slice". For example, select "row" and then specify row index 128 (for the 2-D rendition along the middle row). Again select "Run Program" from the "Program Control" menu. You can now see and interact with the data and hierarchical segmentation results from this new perspective.