Semi-automated land/water segmentation of multi-spectral imagery

Benjamin Cook and Stephanie Graceffo
Areté Associates
Arlington, USA
bcook@arete.com, sgraceffo@arete.com

***Abstract*—Segmentation of land and water regions is necessary in many applications involving analysis of remote sensing imagery. Not only is manual segmentation of these regions prone to considerable subjective variability, but the large volume of imagery collected by modern platforms makes manual segmentation extremely tedious to perform, particularly in applications that require frequent re-measurement. This paper examines a robust, semi-automated approach that utilizes simple and efficient machine learning algorithms to perform supervised classification of multi-spectral image data into land and water regions. By combining the four wavelength bands widely available in imaging platforms such as IKONOS, QuickBird, and GeoEye-1 with basic texture metrics, high quality segmentation can be achieved. An efficient workflow was created by constructing a Graphical User Interface (GUI) to these machine learning algorithms.**

Keywords—Segmentation, machine learning, remote sensing

I. INTRODUCTION

Segmentation of water and land regions is a promising area for automation in the analysis of remote sensing imagery, with applications in mapping, monitoring flood risk, erosion and coastal change, the topography of inter-tidal zones, water resource assessment, and perhaps estimating river stage. In addition to manual segmentation, several other methods have been used in practice, but each presents significant disadvantages.

The simplest method of image segmentation is thresholding. In thresholding, pixels are allocated to categories according to the range of values in which a pixel lies. This method is extremely fast to implement, but it is limited in accuracy. Thresholding can only be accurate for relatively simple classification problems, and even then it is non-trivial to determine an appropriate threshold without resorting to trial and error.

Clustering techniques attempt to group together patterns that are similar in some sense. It finds partitions such that objects within each cluster are as close to each other as possible and as far from objects in other clusters as possible. When used for image segmentation, these methods group together pixels with high similarity, i.e., pixels with small distance in feature space, into the same cluster. While these methods are able to produce clusters of highly similar pixels, the number of clusters of the image must be known a priori, which may not always be feasible in real-world applications. Additionally, different initial seed placements for the initial clusters may produce different segmentation results.

Histogram based methods are similar to clustering methods These algorithms compute the histogram from all the pixels in the image then use the peaks and valleys in the histogram to locate the clusters in the image. Histogram-based methods fail, however, when it is difficult to identify significant peaks and valleys in the image histogram.

Edge based segmentation relies on detecting edges in an image which are assumed to represent object boundaries and used to identify the objects. All pixels are classified as either an edge or non-edge pixel depending on the output of the edge detection filter, and those pixels within the same boundary are all allocated to the same category. The greatest shortfall with edge-based segmentation is the variation that may occur in the strength, sharpness, and topology of edges in an image. Edge detection is often subject to false detections and missed detections which could adversely affect the accuracy of the segmentation. Poor edge detection nearly guarantees poor segmentation results.

Region based algorithms essentially take an opposite approach from the edge based algorithms. Instead of finding an object boundary and locating an object by filling the boundary in, region-based approaches start in the middle of an object and "grow" outward until it meets the object boundary. These algorithms operate iteratively by grouping together pixels which are neighbors with high similarity and splitting groups of pixels that are dissimilar. The disadvantages of this technique are that outputs are often either over-segmented (too many regions) or under-segmented (too few regions) and these methods cannot find objects that span multiple disconnected regions.

While several methods of segmentation have been discussed, they each present significant challenges from requiring additional pre- or post-processing steps to requiring a priori knowledge that may not realistically be available. Machine learning in general provide faster, more efficient means of performing complex tasks, making these types of algorithms good candidates for tasks such as segmentation.

This is a DRAFT. As such it may not be cited in other works.
The citable Proceedings of the Conference will be published in
IEEE Xplore shortly after the conclusion of the conference.

978-0-933957-43-5 ©2015 MTS

II. METHODOLOGY

Machine learning algorithms attempt to learn from data to make predictions about future data. These algorithms build a model from example input data, or training data, in order to make data-driven predictions or decisions. At its simplest, this model can be a statistical regression, and more complex models develop more intricate estimates of the probability density function (PDF) of the data.

Machine learning algorithms can be applied to the problem of image segmentation by treating each pixel (or small cluster of pixels) as a sample drawn from a probability distribution determined by the material class to which it belongs. Thus a trained classifier can make a prediction for any pixel (or cluster) as to its class, and computing this prediction over the entire image results in a segmentation image.

Supervised learning requires training samples, in this context image locations that are "labeled" by the class (land, water) to which they belong. While every image we have seen contains at least some regions that can be easily identified as water and land by eye, recording such information in a machine-readable format is not straightforward. We found that producing labels with a reasonable amount of human effort requires a GUI that is able to display the image while allowing the user to select particular image locations with a mouse.

To this end, we implemented the classification algorithms within a GUI workflow. The resulting software tool ingests imagery from the standard NITF file format, and provides geo-referenced visualization using the Rapid Positioning Capability (RPC) metadata included in such imagery. With a simple mouse interface, the user selects known regions from the image for training data, labeling each as water or land. The corresponding image data is used to train a supervised classifier such as Linear Discriminant Analysis (LDA) or Support Vector Machine (SVM), which then provides an automated classification of the entire image. We find that with the addition of a simple morphological smoothing post-processor, this workflow provided accurate classification on a sample of 12 IKONOS images of estuaries in the continental United States.

A. Inputs and Training Data

The segmentation tool algorithm requires a 4-band, Multi-Spectral Image (MSI) encoded in a National Imagery Transmission Format (NITF) file. From this input, the user generates training data by using the tool to select regions throughout the image and then specifying those regions as either "land" or "water". Once these regions are selected and appropriately labeled, the features necessary to build the prediction model were computed.

B. Features

A subset of points from the selected training data is used to form 4x4 patches on which 12 features are computed: mean, standard deviation, and min-max (difference between the maximum intensity and minimum intensity within each patch), for the blue, green, red, and Near Infrared (NIR) bands. Each patch is treated as a data point in a 12 dimensional input space, determined by these features, and will ultimately support a single classification variable (water or land). Smaller patches would support a greater spatial density of classification, thus a higher resolution segmentation; larger patches can support a larger set of texture features which may result in greater discriminating power. The 16-pixel patch size was chosen to balance those concerns.

C. Learning Algorithm

D. Using these features, the algorithm trains a Linear Discriminant Analysis (LDA) classifier on the training data. As the name suggests, LDA classifiers are classifiers with a linear decision boundary generated by finding a linear combination of features that characterizes or separates two or more classes of objects. This is done by modeling class conditional densities of the data X for each class y using Bayes' rule in Equation (1):

$$P(y|X) = P(X|y) * \frac{P(y)}{P(X)} = P(X|y) * P(Y)/ \sum_{y'} P(X|y') * p(y') \quad (1)$$

In linear discriminant analysis, $P(X|y)$ is modelled as a Gaussian distribution, and the Gaussians for each class are assumed to have different means but share a common covariance matrix. This leads to a linear decision surface such as that depicted in Figure 1.

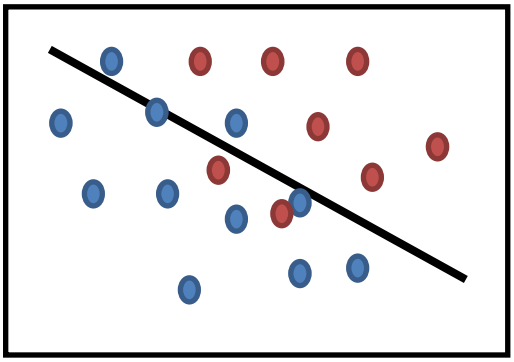

Figure 1: Linear Decision boundary for a binary classification problem

The benefits to using LDA classifiers over other classifiers are that they have closed-form solutions that can easily be computed, are inherently multi-class, and have proven to work well in practical applications. While other classifiers were investigated, LDA was chosen because of its simplicity, quick computation time, and results that were comparable to those yielded by more complex and computationally intensive classifiers.

After training the LDA classifier on labeled data, the classifier is then applied to the entire image to obtain masks representing the segmented regions. Due to the use of the 4x4 patches formed in the feature extraction phase, the resulting mask resolution is coarser by a factor of 4 from the original image.

III. GUI WORKFLOW

To make supervised learning an efficient process, a GUI was constructed providing the user with an interface in which to select training data, quickly view and analyze the results,

and post-process the results as needed. The user interface was developed in Python using the QT framework, and uses the PyQtGraph library (http://pyqtgraph.org) for enhanced image display (pan, zoom, histogram-based color-map stretching). An advantage of the QT framework is the low-level support for affine geometric transformation. We have used this capability to display the image in approximate ortho-rectified geometry, using a linearization of the RPC metadata, without the slow step of re-sampling the image. The GUI application also supports export of the binary classification image to a GeoTIFF file. This section walks through the GUI step by step as a user would when generating the land and water masks using an example IKONOS image of a region of the Chesapeake.

A. Input the Image

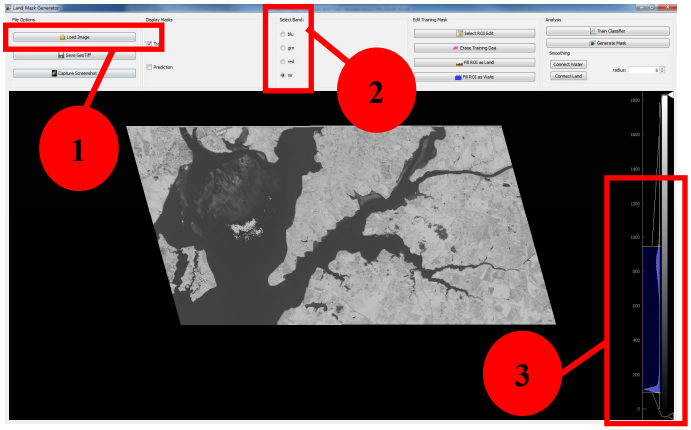

Figure 2: Example image loaded into GUI

First, the user inputs a 4-band multi-spectral image in NITF format into the land mask generation tool by selecting the "Load Input" button (1). (This technique is not specific to 4-band imagery, that is simply the form of the IKONOS data available for this study.) The image loaded into the view window is mapped to the default ground elevation contained in the RPC data in the NITF file. The default band displayed is NIR because it consistently has the highest land/water contrast making it easier for the viewer to distinguish between land and water regions in this band than any other band. The user can, however, switch between the four bands using the radio buttons (2) and adjust the histogram of the image using the histogram tool (3) as well as zoom and pan using the mouse. Altering any of the default settings for the image view will not affect the processing performed.

B. Select Land and Water Training Data

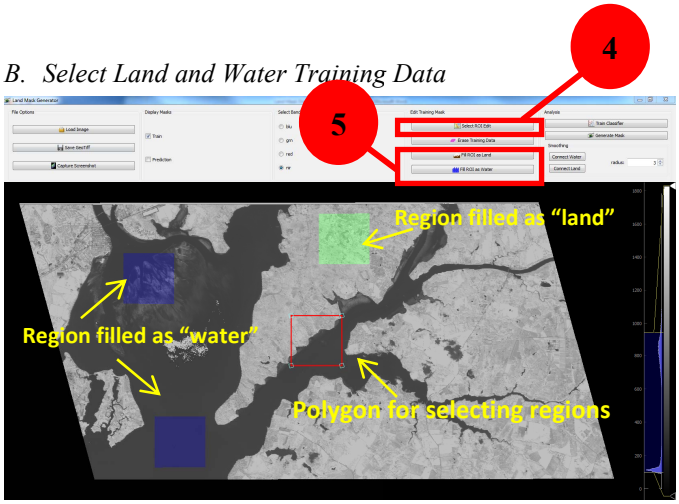

Figure 3: Land and water regions selected for training classifier

Next, the user pushes the "Select ROI Edit" button (4) which drops a red polygon into the image. The user then drags the square to the desired position over either the land or the water. The shape of the polygon can be adjusted by clicking and dragging the vertices; to add another vertex, double click on an edge, and to remove a vertex, simply right click the vertex and select "remove" from the menu.

Once the polygon is over the desired region, the user pushes the "Fill ROI as Land" or "Fill ROI as Water" buttons (5) to label the points appropriately. This process is then repeated until the user has selected and labeled all of the desired regions. At least one water region and one land region must be selected to train the classifier, but multiple regions of both types may be selected. The algorithm then selects a subset of points from all of the land regions and all of the water regions to be used as training data.

When selecting regions in the image, the user should select regions that represent both the majority texture and unique textures throughout the image. In other words, the user should select textures that are most prevalent in the image because those best represent the land or water class for that case, but should also include unique textures that are not characteristic of how those regions normally appear throughout the image.

For instance, if an image contained water and mainly a cityscape, but also contained smaller regions of open field, the user would want to include both the cityscape and the field in the training data. This reduces the risk of having entire land or water regions misclassified for being characteristically different, while still putting more emphasis on what the algorithm is most likely to find in the image. An example of what is meant by unique and majority texture regions is provided in Figure 4.

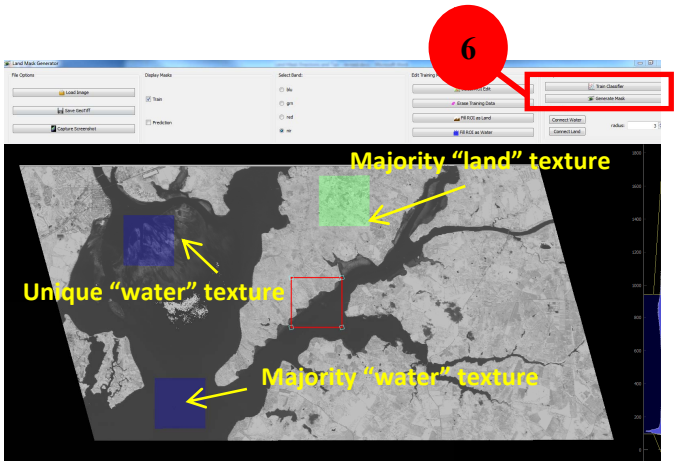

Figure 4: Examples of majority and unique textures

In this image, the user selected one land region that best exemplified the land in the image, but selected two water regions. The bottom region represents how most of the water looks throughout the image while the top region captures water of a different texture possibly due to sediment. Without including this region, the algorithm may classify the region as land rather than water due to the suspended sediment.

C. *Train Classifier*

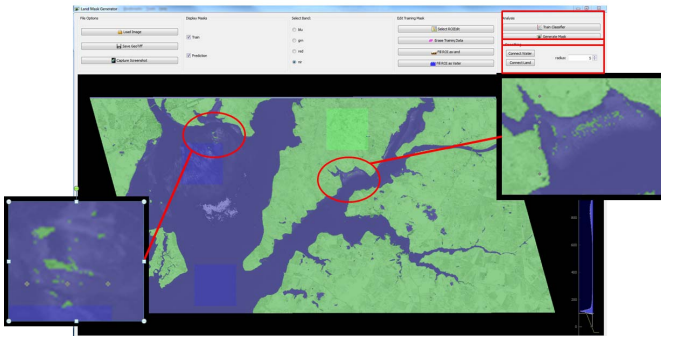

Figure 5: Mask generation

After the user has selected and labeled the training data regions, the user pushes the "Train Classifier" button (6). At this point, the algorithm extracts the tile data for the labeled regions, calculates the feature vectors for each tile, and passes these to the LDA classifier to generate a prediction model which can then be used to classify the remaining points in the image.

D. *Generate and Refine Mask*

Once this is complete, the user pushes the "Generate Mask" button and the resulting mask is overlaid in the image. While the algorithm predicts the classification of each pixel reasonably well, it may still be necessary to refine the mask using morphological opening and closing to connect the land and water regions, respectively. This is accomplished by pushing the "Connect Water" button and/or the "Connect Land" button.

The land regions are refined using the opening morphological operation, which is an erosion followed by dilation, Equation (2).

$$A \circ B = (A \ominus B) \oplus B, \qquad (2)$$

This removes some of the foreground, or water, pixels from the edges of the foreground regions and is similar to but less destructive than erosion alone. To perform morphological opening, a structuring element is slid around *inside* each foreground region. All pixels that can be covered by the structuring element with the structuring element being entirely within the foreground region will be preserved as foreground. An example of this is shown in Figure 6.

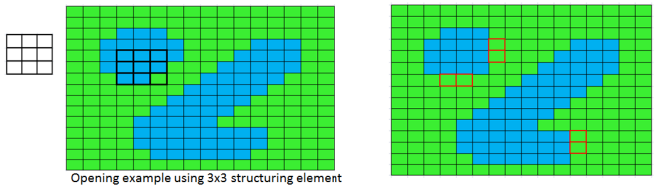

Figure 6 (left to right): 3x3 structuring element, input image, image with morphological opening applied. The green pixels outlined in red show which foreground regions were not kept.

Similar to morphological opening is morphological closing which is used to refine the water regions. Morphological closing is a dilation followed by erosion, Equation (3).

$$A \bullet B = (A \oplus B) \ominus B, \qquad (3)$$

It enlarges the boundaries of the foreground, or water, regions in an image and shrinks the background, or land, holes in such regions. Closing is similar to dilation, but is less destructive. To perform morphological closing, the structuring element is slid around *outside* each foreground region. Any pixel that can touch the structuring element with the structuring element being entirely within the background region will be preserved as background. An example of this is shown in Figure 7.

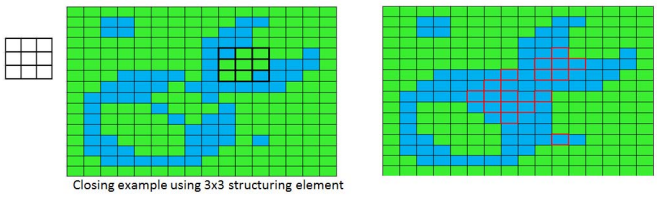

Figure 7 (left to right): 3x3 structuring element, input image, image with morphological closing applied. The blue pixels outlined in red in the output image represent the pixels that were added to the foreground region, i.e., the holes in the foreground region that were filled.

Figure 8 shows the resulting mask after these operations are applied.

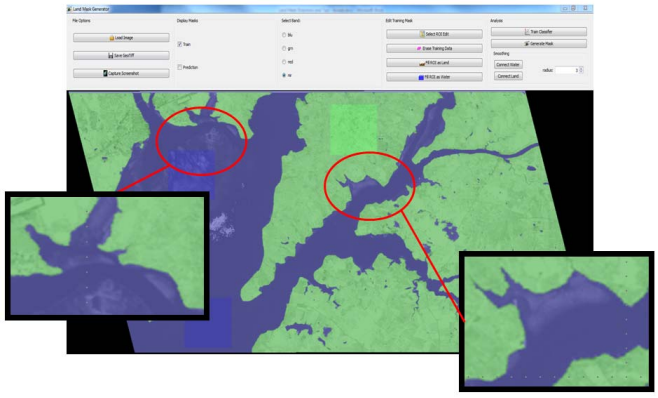

Figure 8: Refined Mask

IV. Parameters and Error Metrics

A. Parameters

Several parameters were considered in the design of this algorithm. Among these include the features used, as well as the tile size for computing the features. While the algorithm uses the mean, standard deviation, and difference between the minimum and maximum values for each of the four bands, entropy was also considered. The results, however, did not improve significantly enough to justify the additional extensive computational time. Additionally, when determining a tile size for computing the features, the trade-off between computational speed, classification accuracy, and output resolution was considered. While a larger tile size sped up the algorithm, the resolution was degraded. Additionally, tile sizes that were too big resulted in a number of patches that contained both land and water making it difficult to compute the features for just one class.

Different classifiers were also investigated, however the influence on classification accuracy was minor. The LDA classifier was chosen mainly due to its simplicity. The benefits of more complex classifiers typically require larger training samples to achieve, which is not the case in our expected usage. While an image can have as many as 10^8 pixels, and a training set may have up to 10^4 -10^5 tiles, these training samples are not independent: to minimize human effort an operator will select a highly localized training sample, which will be more similar by the relevant spectral and textural metrics than a random sample from a single class across the entire image. Thus from a machine-learning perspective this problem is relatively data-poor and cannot support a complex classifier. All classifiers were implemented through the standard Python machine learning package, scikit-learn (http://scikit-learn.org/stable/).

As discussed previously, the algorithm selects a predetermined number of samples from each region that the user selects so that not all pixels that are selected are used. This reduces the amount of redundant information contained in large regions selected by the user and improves the speed of the algorithm. The number of these points used was experimented with to achieve a reasonable balance between computation time, accuracy, and good representation of the selected region.

The last parameter considered in the design for this algorithm is the size of the structure element used for smoothing in the post-processing step. The idea was to determine a default size that would generally provide good smoothing results without being too destructive. In practice the desired smoothing radius varies considerably from image to image, and it was deemed preferable to retain this as a user-adjustable parameter.

B. Error Metrics

The standard metric of accuracy for machine learning algorithms is cross-validation on a labeled sample distinct from the training data. Typically the training and validation samples are drawn from a random partition of a single dataset, and in that case the difference in classification accuracy between the training and cross-validation sets measures the ability of the classifier to generalize to the real-world source from which the sample was drawn.

The case of pixels (or super-pixels) in an image differs from the normal cross-validation scenario because the training sample is not drawn randomly from the underlying dataset. On the contrary, the pixels of the training sample are selected by user input, in this case entering a small number of polygonal regions with the mouse, and subject to a practical need to accomplish this data entry task quickly and with minimal effort. The resulting training sample is thus highly localized within the image, and will usually be more homogeneous than the image overall. More precisely each class will be more homogeneous in the training sample than in the whole image. The accuracy at segmenting the full image therefore measures the difference between the training samples and the full class, relative to the difference between classes.

In our evaluation of accuracy, we considered both visual inspection (against the intuitively clear regions of water and land) and quantitative comparison with a manually generated land mask. In the qualitative evaluation the algorithm-generated mask was overlaid on the image and the accuracy and resolution were visually analyzed. Doing this overlay allowed diagnosis of which region types proved problematic to the algorithm and it helped determine how coarse of a resolution was allowed before the shorelines started degrading.

Each parameter experiment was also timed and the computational time as a function of the segmentation accuracy was greatly considered. For example, segmentation results for two different parameter setups may have yielded similar results, but if one setup had a significantly higher computational time then that combination of parameters was eliminated.

A quantitative error metric (number of incorrectly classified pixels) requires a "truth" (hand segmented) mask. In practice this hand segmented mask was slightly different from the segmentation output in requiring three classes: land, water, and not classified. This third class represented regions that were either too intricate to accurately segment by hand or regions that could not be identified with certainty as being land or water. These regions types most often occurred when the

water was shallow and it could not be determined whether it was actual water surface or the ground beneath the water that was being imaged, and also when there were high-sediment or muddy regions. The error metric thus compared only those pixels for which a hand-segmented label was available.

V. Results and Analysis

A. Results

This section contains several examples of images from the IKONOS dataset. Continuing with the Chesapeake example, the algorithm accurately classified the land and water regions. The zoomed insert of Figure 9 shows how thin the water lines can get before they are misclassified, due to an insufficient number of pixels being available for the calculations. Additionally, circled in red are clouds over land but were classified as water. One possible remedy to this problem is to introduce a third class for clouds, at least in the case of fully opaque clouds.

Figure 10 shows the four bands and how much each feature was weighted in the classification. The mean of the NIR band clearly dominates the features, and in this particular case may have been sufficient alone for classification with an appropriate choice of threshold. (Note that the LDA classifier reduces to a threshold in the case of a single input feature, e.g. a panchromatic image with no additional texture features.)

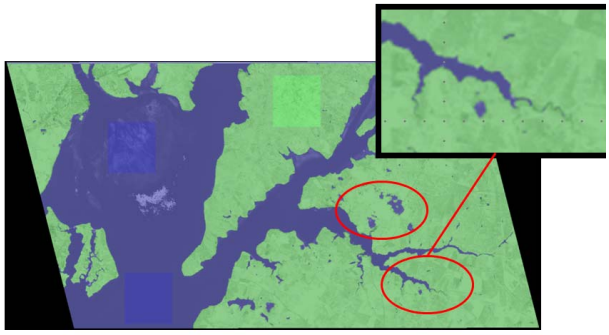

Figure 9: Chesapeake land and water segmentation results

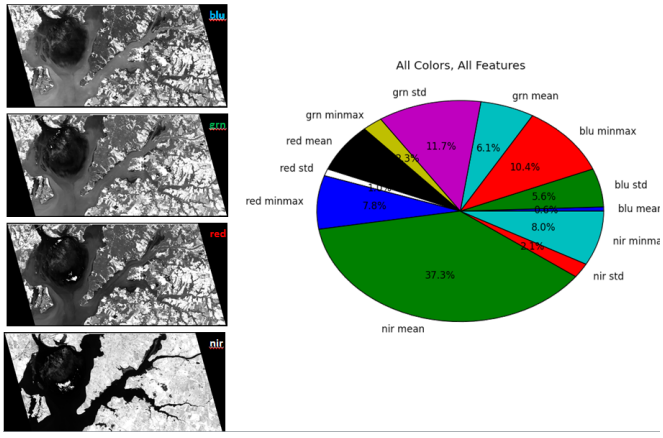

Figure 10: Chesapeake bands and weight plot for all features

Figure 11 shows an example of an IKONOS image taken of Philadelphia. Again, the zoom inserts show the limitations of the algorithm when it comes to the thin water lines; when there is an insufficient number of pixels representing a particular region, the algorithm may misclassify those pixels.

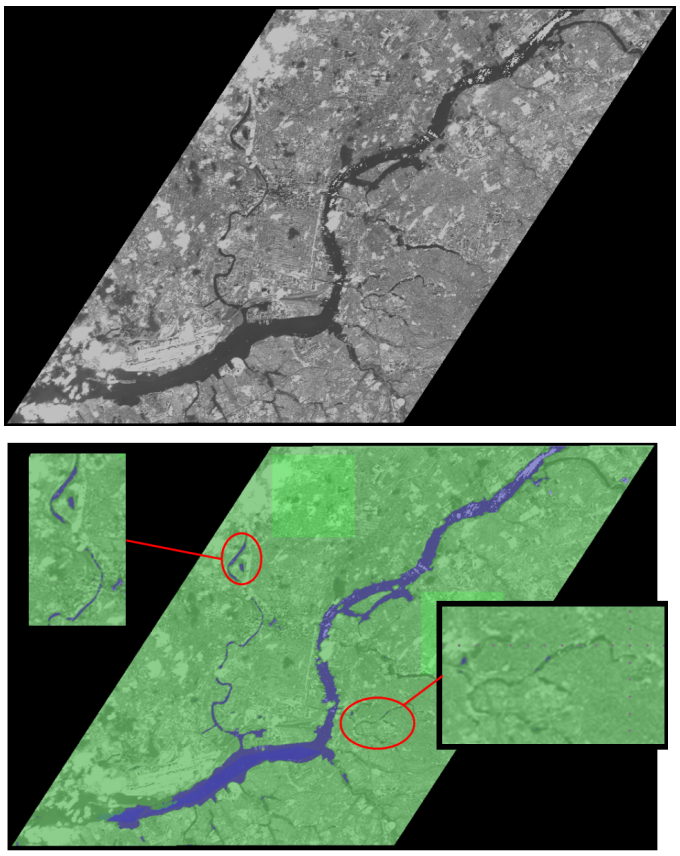

Figure 11 (top to bottom): Philadelphia input image, Philadelphia land and water segmentation results

The weights for each of the computed features can be seen in Figure 12 below. Again, note the dominance of the mean of the NIR band in this case as well.

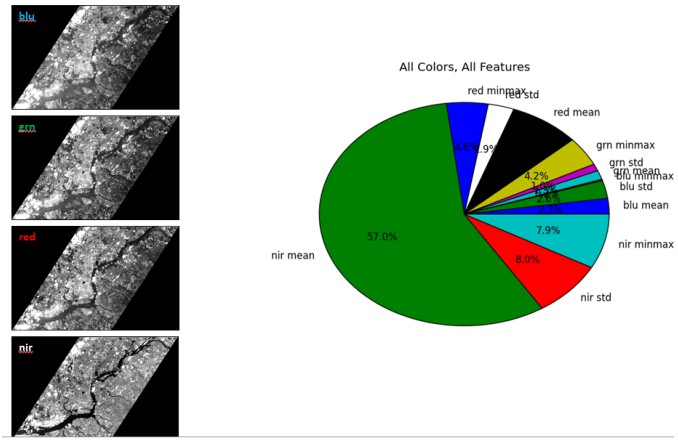

Figure 12: Philadelphia bands and weight plot for all features.

Figure 13 shows an example IKONOS image taken at Cape Fear. The individual bands and weight plot is shown below in Figure 14.

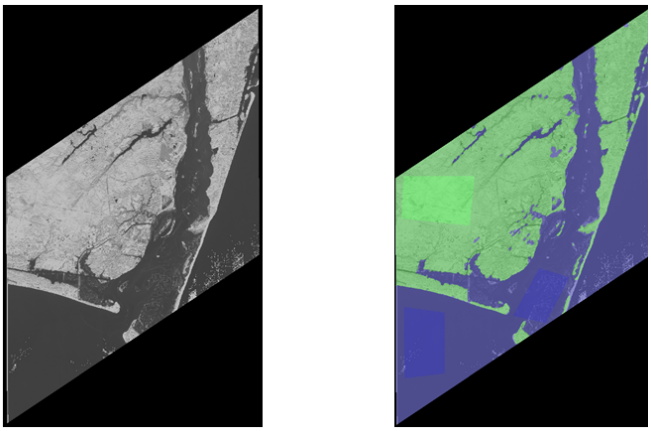

Figure 13 (left to right): Cape Fear input image, Cape Fear land and water segmentation results

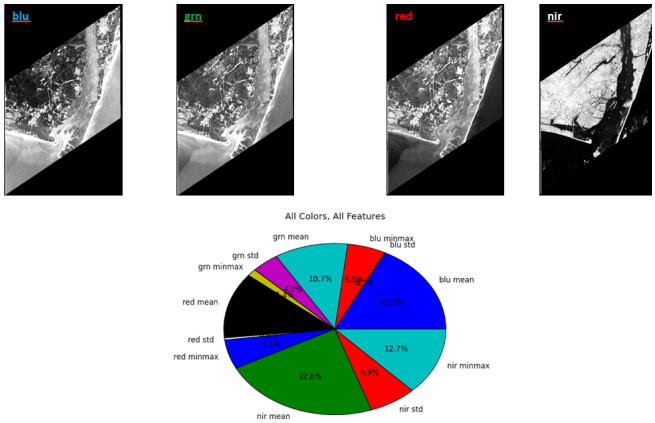

Figure 14: Cape Fear bands and weight plot for all features.

Figure 14 shows again, that the most prevalent feature is the NIR band mean. It is, however, less dominant than in the previous two cases indicating that this one feature alone may not be sufficient for classification. Just as before, the algorithm yielded good classification results with the only discrepancies being among the thinner water lines where there may have been an insufficient number of pixels representing the region. Also, note the two different water texture regions that were selected, one representing the smooth water and the other representing the sedimentary water. The inclusion of both types aided in a more accurate classification.

B. *Points of Failure*

These examples show the accuracy of the classification algorithm, but they also show a common trend in what types of regions were misclassified. The Chesapeake case showed that clouds over land were often misclassified as water; although this was not as problematic in the other two cases. Another area in which the algorithm failed to perform accurately, was in classifying the thin waterlines; although this is may be a reflection more on the data itself than the algorithm. The algorithm can only be as good as the input data, so if there is insufficient information in the input data, the algorithm may not be as successful.

Smoothing the boundaries was a necessary post-processing step to remove extraneous pixels and to fill holes, but over smoothing of the boundaries often removed more of the thin water lines than desired, as in the Philadelphia case.

VI. Summary and Future Work

A. *Summary*

The proposed algorithm behind the segmentation tool works by inputing a 4-band MSI in NITF format into the tool. The user then selects the training regions making sure to include both majority textures and unique textures for each region type in the image. From the user-selected regions, the algorithm selects a random subset of these selected regions. This speeds up the algorithm by reducing the redundant information contained in large regions. The mean, standard deviation, and min-max (difference between minimum value and maximum value within the patch) are computed from the training samples for each of the four bands (blue, green, red, nearinfrared) for a total of 12 features. LDA classifier uses the computed features of the training data to model the class-conditional densities using Bayes rule. From these models, a linear decision boundary is formed and used to determine the classes of the remaining pixels in the image. Finally, the user uses morphological opening and closing to refine the mask boundary generated by the classifier to remove isolated pixels and fill holes.

B. *Future Work*

Future work will improve the robustness of the technique by addressing the challenge of clouds. Including clouds as a third class may improve the classification results in the case of opaque clouds, but thin clouds cause the more complex phenomenology of mixed pixels. A more systematic study of various texture features, for example Haralick features, would be useful to ensure that the algorithm is optimized to its full potential by using the best, or most distinguishing, features. Also of interest is expanding the algorithm to work on other imaging modalities. With panchromatic imagery there is no color information but the higher resolution typically achieved could support more complex texture metrics. Finally, a larger sample of training images would allow the investigation of how well such classifiers generalize to different images and geographical regions. If successful, that would allow a pre-trained classifier to operate fully automated.

All imagery © DigitalGlobe.