EXTRACTING SPATIALLY AND SPECTRALLY COHERENT REGIONS FROM MULTISPECTRAL IMAGES

Farhana Bandukwala
BAE Systems - GXP
San Diego, CA
farhana.bandukwala@baesystems.com

Abstract

Extracting spectrally homogeneous regions as features from hyperspectral and multispectral raster data has unique challenges when accurate shape preservation is a priority. We tackle this task by representing neighborhoods that contain heterogeneously classified pixels as a graph. We then use graph-cut based combinatorial optimization to eliminate spuriously classified pixels. After the region of interest is uniformly classified, we use a vectorization step to extract it as a feature.

1. Introduction

In remote sensing, one critical function for image analysts is the capability to identify regions in imagery that correspond to a particular object or material. Automatic extraction of image areas that represent a feature of interest requires two specific steps. The first step is to accurately classify the pixels that represent the region while minimizing misclassified pixels. Secondly, a vectorization step extracts a contiguous boundary along each classified region which, when paired with its geo-location, can be inserted in a feature database independent of the image.

We have implemented an algorithm, to be used by our commercial customers, that extracts spatially coherent regions which have been classified as a particular object or material. We apply this algorithm to all types of imagery, including standard 3-band, multispectral images, and hyperspectral images which contain non-visible bands. Robust automatic classification and coherent region extraction is especially relevant for hyperspectral images which have numerous bands making simultaneous visualization and therefore manual intervention difficult.

While relying on traditional classification first to identify matched pixels, our contribution lies in the next step where we efficiently apply graph cuts based optimization to eliminate spuriously labeled pixels and extract spatially contiguous regions as intact features [1],[6]. In this paper we explain how we use the optimization method and also the feature extraction step.

2. Problem Description

The goal of classification algorithms is to label pixels which have spectral signatures that fall within a distribution defining a region of interest. A pixel belongs to a classification set when the distance, in feature space, between the pixel's spectral signature and the signature of a representative set of pixels is small. Classification algorithms vary in how the feature vector (and therefore feature space) is defined, how the distance metric is defined, how a representative set of pixels or distribution is determined and in the algorithm by which pixels matches are identified. Nevertheless, they all share the concept of goodness-of-fit, a per pixel score measuring how well a pixel actually fits the target spectral distribution. Examples of supervised and unsupervised classification algorithms include clustering algorithms, support-vector machines, matched filter algorithms and neural networks to name a few [5]. Purely relying on spectral signatures, may not lead to good spatial localization of the pixels of interest. Spurious pixels could easily fall on the wrong side of the classification criteria and become incorrectly classified. Multiple spuriously classified pixels will degrade coherent region extraction [6].

In the context of our software, users are allowed to determine class membership via supervised algorithms such as spectral angle metric and matched filters. Also, unsupervised classification algorithms are available to our users such as general clustering and constrained energy minimization. In addition, specialized algorithms such as NDVI (Normalized Difference Vegetation Index) or NDWI (Normalized Difference Water Index) are also implemented. Due to the varied contexts that determine which spectral classification is applied by our users, it is critical to allow a user to select the appropriate algorithm. Our goal is to provide a post-processing algorithm that outputs spatially consistent regions that can function with the goodness of fit metric of any classification algorithm.

Post-processing with filters or morphological operators is often used to condition the classification results output from spectral classification algorithms. These post-processing steps tend to adversely modify the details of the region boundary and eliminate fine features. In our application, users have a low tolerance for the loss of fine details in the shape. Figure 1, is an example of a multispectral image of a harbor. In this example, the docks and pier are just a few pixels across and could easily be removed in an erosion/dilation step in the presence of enough spuriously classified pixels.

3. Algorithm Summary

Our goal is to accurately identify the boundary of a spatially consistent set of pixels that belong to a region of interest, with the intent of extracting that region as a distinct feature. We aim to minimize spurious pixels, while maximizing spatial consistency. Currently, we focus on only a single classification set at a time. We use graph-cuts based combinatorial optimization to explicitly optimize the data and smoothing constraints [6]. Once spatially contiguous regions are identified, we delineate the boundary of the feature as a set of connected polylines using a custom vectorization algorithm.

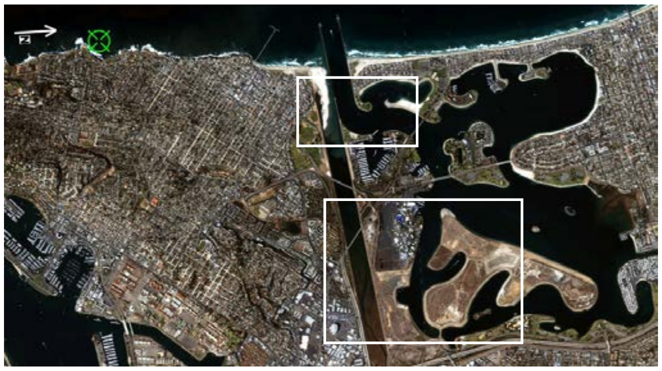


Figure 1: Multispectral image of a harbor. In this example, our goal is to extract the regions containing water, while preserving fine details such as bridges and docks. Some areas on land have bodies of water that we need to include; in the water there are vessels that we need to exclude. Subsequent figures show results of our algorithm in the rectangles delineated here.

The input to our module is a set of scores, one per pixel in an image. The score denotes how strongly a pixel matched the target spectrum. The score, output from any one of several classification algorithms available to our users, measures the goodness-of-fit of a particular pixel. Using the scores from a user selected classification algorithm we compute a heterogeneity metric at each pixel. Each point with a non-zero heterogeneity metric serves as the center point of a neighborhood at which a graph will be initialized and which will be optimized to be spatially coherent. Using graph-cuts combinatorial optimization, we identify the partitioning of that neighborhood into pixels that either belong to the interior of the region of interest or to the exterior.

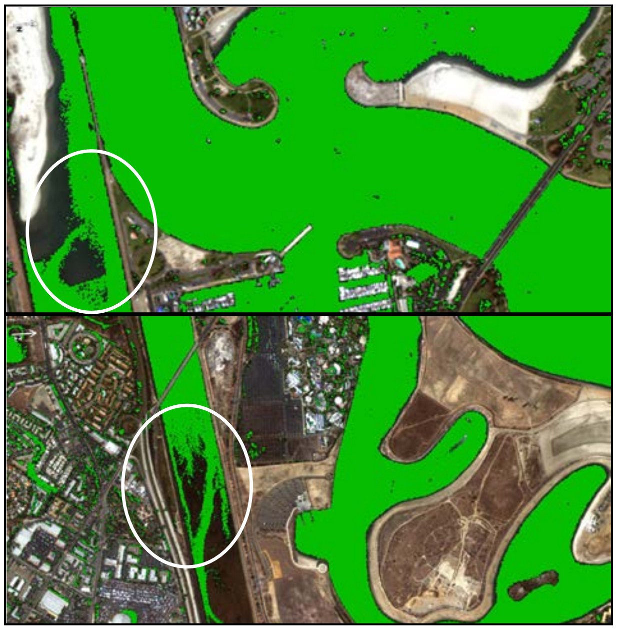


Figure 2: The above two areas show the result of simply classifying the water areas by spectral classification and then using a threshold to denote interior (green areas) vs exterior. The circles highlight areas containing many spurious classifications.

4. Algorithm Details

4.1. Identifying heterogeneous pixels

Once we have a goodness-of-fit score per pixel, the first step of our algorithm calculates how heterogeneous a particular pixel is with respect to its 8 neighbors. The heterogeneity measure identifies pixels that could potentially be classified differently from their neighbors. Typically, these pixels have scores at the middle of the distribution and could have easily flipped classification. Therefore, we do not simply compute a difference between the pixel's score and its neighbor's score. We first identify if a pixel is statistically close to the threshold that identifies region interior with region exterior. If the pixel is close, we then use the difference in the score between the pixel and its neighbors, normalized by local standard deviation, as the heterogeneity measure. Thus at a given pixel (i,j), the heterogeneity measure ($H_{(i,j)}$ below) is

$$H_{(i,j)} = (1/(8s_l))\sum_{m,n}|f_{(i,j)}-f_{(m,n)}|$$

where $f_{(i,j)}$ is the goodness-of-fit score of the pixel, $f_{(m,n)}$ is the score at each neighbor and s_l is the local standard deviation. If a pixel's score indicates that it either strongly matched or strongly did not match the region of interest, the heterogeneity measure will be zero.

4.2. Graph initialized at heterogeneous neighborhoods

At each point on an image where the heterogeneity measure is non-zero, we initialize a graph representing the data constraints and the homogeneity constraints on the local pixel distribution. Processing local neighborhoods is vital in our operational context for performance reasons. Our users deal with large, tiled imagery that is impractical to handle in its entirety. The spatial coherency constraint is inherently local since we are considering areas, typically at boundaries, with spuriously classified pixels. We justify optimizing over local neighborhoods for this reason. In our current implementation, the local neighborhood is empirically set to 10% of the tile size. We are investigating setting the neighborhood size (bounded by a maximum) to include all locally connected, heterogeneous pixels.

The nodes of the graph represent the pixels within the neighborhood to be processed. Each node is connected, via an arc, to a source node and a sink node. The arc connecting the node to the source is weighted by the normalized goodness-of-fit score of the pixel. The arc connecting the node to the sink is the complement of the normalized goodness-of-fit score. In addition, arcs connect each node with another node that represents an immediate neighboring pixel. The arc connecting two pixel nodes (not the source nor sink) is weighted by the complement of the pair-wise heterogeneity score between these two pixels. Thus, pixels that have similar goodness of fit scores will have arcs with large weights.

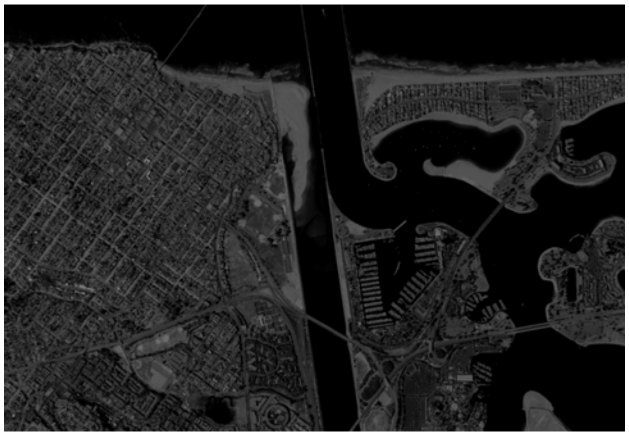
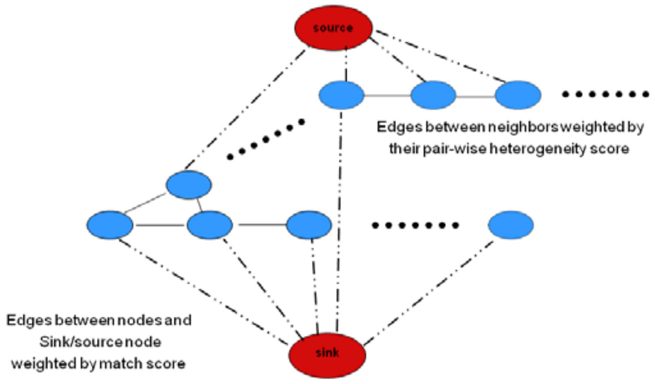


Figure 3: For the harbor data, this image depicts the difference at each point between the spectral signature of the pixel and the spectral signature of the seed. Low intensity values signify proximity in spectral feature space to the seed pixel.


source
Edges between neighbors weighted by their pair-wise heterogeneity score
Edges between nodes and Sink/source node weighted by match score
sink
Figure 4: Diagram of graph initialized at each heterogeneous neighborhood.

4.3. Graph Cut Optimization

Once the graph is initialized for a particular heterogeneous neighborhood, we use the maximum flow algorithm to eliminate arcs. The initial flow magnitude is the minimum residual capacity of any one arc. The flow into and out of a node is the weight of its incoming or outgoing arc respectively. Our incoming and outgoing arcs have the initial capacity. The graph cut optimization method has been widely used to solve energy minimization problems in computer vision [1][2][6].

At the start of each flow cycle a node connected to either the source or sink propagates the initial flow to all nodes that are either similarly labeled or unassigned and are connected to it via arcs that have a residual capacity greater than the flow magnitude. As the flow is propagated to connected nodes, these nodes form either a source tree or a sink tree. The residual capacity of an arc is reduced by the flow magnitude. If the flow magnitude equals the residual capacity of an arc, that arc is saturated and therefore cut [1]. The residual capacity of a saturated arc is added to the total cost of the cut. At each iteration, the total cost of the cut is minimized. When all nodes are separated from either the source or the sink node via a cut, the optimization terminates. The nodes that are connected to the source node are identified as the interior nodes of the region.

4.4. Classified region extracted as a feature

The core of the region extraction algorithm depends on the ability to automatically extract the boundary pixels robustly. We maintain shape accuracy while minimizing the shape complexity wherever possible. To achieve this we divide the task into three steps. The first step extracts simply connected boundary segments that can be unambiguously labeled as being on the positive side of the region or on the negative side. (Positive/negative is relative to the vector defined by the start and end point of the segment, using the right-hand rule.) The second step consists of a greedy algorithm that connects segments with the same orientation that are close and satisfy certain quality metrics. The third step simplifies the contour such that only vertices that are relevant to shape preservation within a user-specified tolerance are maintained.

To extract simply connected segments, we identify the nodes of each instantiated graph which has a cut in the arc connecting it to its neighbor. Simple boundary segments are formed by following the points which are disconnected from its neighbors. The positive/negative designation is based on whether other interior nodes are along the positive side of the vector connecting boundary nodes or the negative side.

Next, these simple segments are input to a greedy, stitching algorithm that iteratively takes an available segment and finds the closest simple segment that has the same positive/negative designation and fulfills edge consistency metrics. The edge consistency metrics discourage joining of edges that self-intersect or have a sharp angle at the join. For each region, the stitching algorithm terminates when the start and end segments are the best two remaining candidates to connect, resulting in a closed contour.

Once the contour is closed, thus defining the extents of a region, we simplify the contour by removing vertices that have minimal significance to the overall shape of the region [3]. The significance of each vertex is determined by how much the local shape deviates if the vertex is removed. If the shape deviation is within a user specified tolerance, the vertex may be removed. Once these regions are extracted, insert them into a feature database.

4.5. Implementation details

Since local regions are optimized separately, disjoint regions may be processed in separate threads. The graph-cut optimization only requires that memory for the graph is allocated once per thread. At each new neighborhood the graph is merely re-initialized with the appropriate scores.

We are able to extract multiple regions that are classified as disparate distributions. Currently we process one classification set at a time. We are expanding of our graph-cut algorithm to jointly optimize multiple classifications.

The back-end architecture of our software allows seamless interaction with neighborhoods that span tiles. Therefore, tiling does not hamper our algorithm. Typically we process large images, which consist of multiple 1024x1024 tiles, within seconds on regular PCs. The actual processing time is highly dependent on the complexity of the region.

5. Results

Results of our spatially coherent region extraction algorithm are presented in the following figures. We extracted the water regions from the harbor image. The initial classification uses a user-specified seed pixel located in the water region. The seed pixel merely serves to identify the spectral signature of the region of interest. This manual seed identification is the only user intervention in the entire workflow. The characteristic spectral signature for the region of interest may also be input from an external process. Next, the entire image is processed by computing the spectral angle between the spectral signature of the seed pixel and the spectral signature of the each image pixel. The input to our module is the set of goodness-of-fit scores at each image. We use these scores to compute the heterogeneity measure for each pixel. Once the regions are clearly delineated, we extract them as features using the vectorization algorithm explained above. These extracted features are shown in the following figures.

Figures 5, 6 and 7, illustrate that the boundaries of a region are non-trivial to extract accurately. With the graph-cut optimization, we can effectively remove spuriously classified pixels and, therefore, accurately represent the underlying shape of the region.

The next set of results depict a different workflow that contains less manual intervention. The example image is a hyper-spectral image which contains mostly non-visible IR bands. In Figure 8, we have visualized 3 of the 189 bands by inserting them into the standard red/green/blue channels. When dealing with this genre of imagery, it is critical that the software automate as much of the processing as possible since interacting with the sheer amount of data overwhelms most PCs. Thus we automate the selection of spectral signatures that represent disparate regions of interest. The user only intervenes to select a distance in spectral feature space that denotes the distribution span that will be considered the interior. Figure 9 shows the results of unsupervised k-means classification on this data. Note that this step identifies three spectral distributions which segment the image. Figure 10-11 shows our multi-class results on this hyper-spectral image. One region (green) has a spectral peak around 1.2 micron wavelengths. The next region (orange) has a spectral peak around 2.3 micron wavelength and the third region (yellow) has a multi-modal distribution that peaks in the .5 micron and 1 micron wavelength range. In Figure 10, we have taken the representative spectral signature from the unsupervised clustering algorithm for each region but we have modified the distance threshold to limit the extent of some of the regions. We do this to allow visualization of distinct areas and spatially contiguous features. Figure 11 shows the end result of our graph optimization and vectorization steps on each of the classified sets.

References

[1] Y. Boykov, O. Veksler and R Zabih, “Fast Approximate Energy Minimization via Graph Cuts”, *IEEE – PAMI*, vol 23, no. 11, pp. 1222-1239, March 2001.

[2] Y. Boykov and V. Kolmogorov, “An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision”, *IEEE – PAMI*, vol 26, no. 9, pp. 1124-1137, 2004.

[3] D. Dori and W Liu, “Sparse Pixel Vectorization: An algorithm and its performance evaluation”, *IEEE – PAMI*, vol 21, no. 23, pp. 202-215, March 1999.

[4] X. Hillaire and K Tombre, “Robust and accurate vectorization of line drawings”, *IEEE –PAMI*, vol 28 no. 6, pp 890-904, June 2006.

[5] D. Lu and Q. Weng, “A survey of image classification methods and techniques for improving classification performance, *International Journal of Remote Sensing*, vol 28, no.56, pp 823-870, March 2007.

[6] R. Zabih and V. Kolmogorov, “Spatially Coherent Clustering Using Graph Cuts”, Proceedings of IEEE CVPR, 2004.

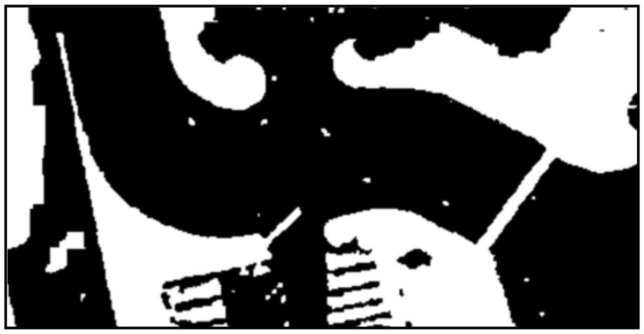


Figure 5: Spatially coherent regions extracted after applying graph-cut optimization. Black regions denote water areas. The docks and water vehicles are preserved, while spuriously classified pixels are accurately incorporated into either the land or water regions.

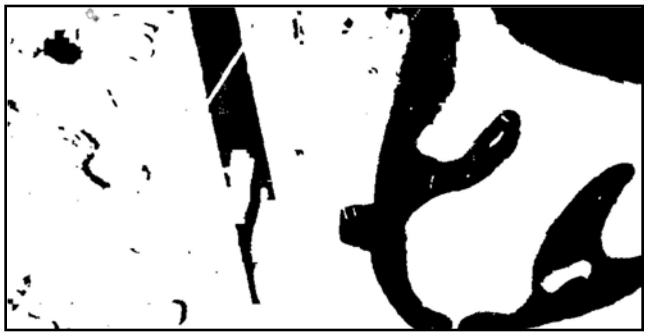


Figure 6: Result of graph-cut optimization to produce spatially coherent regions. Notice how the pixels that would have been spuriously labeled near the sandbar and on the land areas are cleaned. The details of the bridge, docks and water vehicles are preserved.

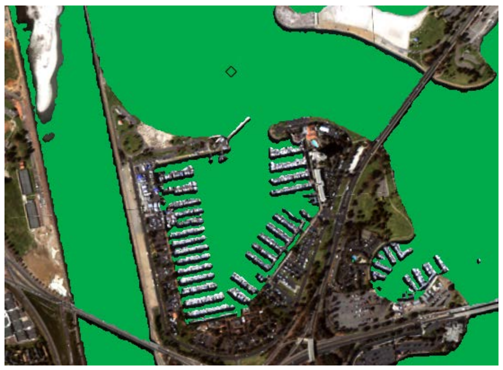


Figure 7: Water features extracted from the harbor image (in green) overlaid over the original harbor image. Notice that spurious regions are removed while small details are preserved, especially along the bridges and docks.

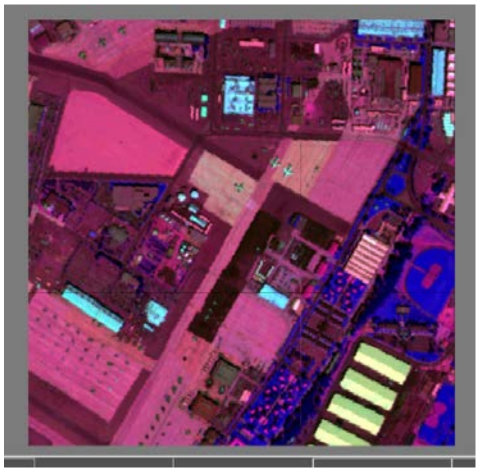


Figure 8: Visualization of some IR bands of a hyper-spectral image. We have put the visible red band in the green channel, a near IR band in the blue channel and the far IR band in the red channel. Different materials are prominent in different bands and can be classified accordingly.

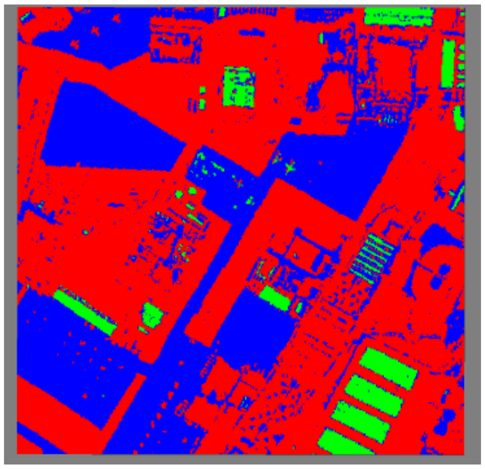


Figure 9: Three distinct distributions are extracted using an unsupervised k-means clustering algorithm. In this image, the blue regions have a single peak in the far IR range, the red regions have a single peak in the near IR range and the green regions have a mixed distribution with a peak in the visible and near IR range.

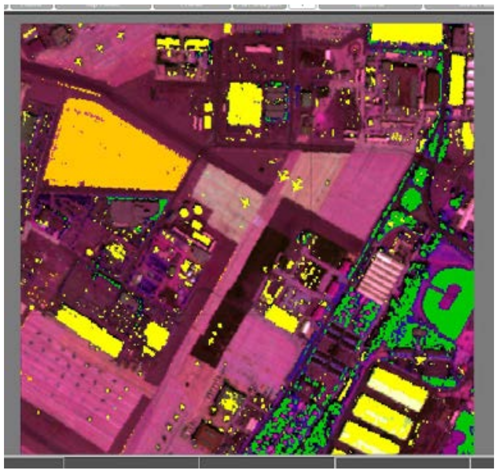


Figure 10: In this image, we have superimposed an overlay on the original mage showing the results of classification and thresholding using three distinct spectral signatures. Green overlay pixels denote pixels that are close to a spectral signature for strong near IR bands. Yellow pixels denote image locations that are strong in the visible and near IR. Orange pixels denote locations that are strong in the far IR range. Notice how spuriously classified pixels are present across the whole image. Taken as is, coherent region extraction would be difficult.

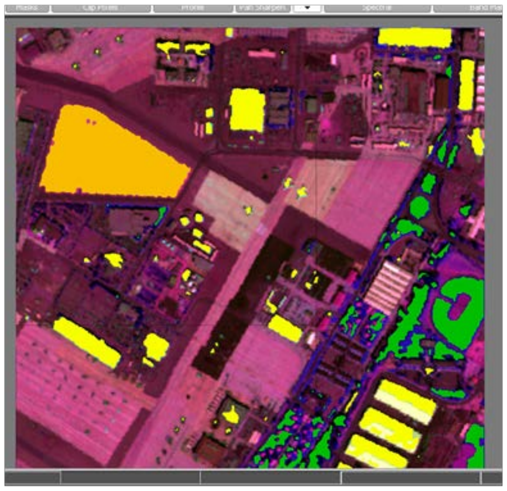


Figure 11: Extracted regions are overlaid onto the original image shown above. Notice how spurious regions are eliminated whereas legitimate small regions (such as the planes) are preserved.