# Automated Feature Generation in Large-Scale Geospatial Libraries for Content-Based Indexing

Kenneth W. Tobin[1*], Budhendra L. Bhaduri[2], Eddie A. Bright[2], Anil Cheriyadat[2],

Thomas P. Karnowski[1], Paul J. Palathingal[3], Thomas E. Potok[3], Jeffery R. Price[1]

[1]Image Science and Machine Vision Group, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831-6010
{Tobinkwjr, Karnowskitp, Pricejr} @ornl.gov
http://www.ornl.gov/sci/ismv
[2]Geographic Information Science and Technology Group, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831-6017
{Bhaduribl, Brightea, Cheriyadatma} @ornl.gov
http://www.ornl.gov/sci/gist/gisthome.html
[3]Applied Software Engineering Research Group, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831-6085
{Palathingalp, Potokte} @ornl.gov
http://www.epm.ornl.gov/~v8q/

This paper reviews a method for feature analysis, segmentation, and indexing for region-based management, retrieval, and datamining of large, high-resolution geospatial libraries.

[*]Prepared by Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

# Automated Feature Generation in Large-Scale Geospatial Libraries for Content-Based Indexing


## Abstract

We describe a method for indexing and retrieving high-resolution image regions in large geospatial data libraries. An automated feature extraction method is used that generates a unique and specific structural description of each segment of a tessellated input image file. These tessellated regions are then merged into similar groups, or sub-regions, and indexed to provide flexible and varied retrieval in a query-by-example environment. The methods of tessellation, feature extraction, sub-region clustering, indexing, and retrieval are described and demonstrated using a geospatial library representing a 153 $km^2$ region of land in East Tennessee at 0.5m per pixel resolution.


## 1. Introduction

Large geospatial data libraries of remote sensing imagery are being collected today in higher resolution formats both spatially and spectrally and at an unprecedented rate. These libraries are being produced for many applications including hazard monitoring, drought management, commercial land use planning, estuary management, agricultural productivity [1], forestry, tropical cyclone detection, homeland security, and other intelligence and military applications [2, 3]. While these systems do provide end-users with useful geographic information data products, it is typically required that a user know precise information in a world-oriented dataset regarding a region of study if they are to achieve effective results.

Techniques that facilitate search and retrieval based on image content, for example in a query-by-example environment, can provide an analyst or researcher with a rapid method for searching very large geospatial libraries with minimal query specification. Content-based image retrieval (CBIR) refers to techniques used to index and retrieve images from databases based on their pictorial content [4, 5]. Pictorial content is typically defined by a set of statistical or semantic features extracted from an image to describe the spectral

content, texture, and/or shape of the entire image or of specific image regions. Region-based image retrieval is referred to as RBIR [6]. The literature contains many descriptions of systems for analyzing and accessing geospatial libraries. The systems that are most relevant to our method are described below. Other than these systems that consider content-based indexing of geospatial image information, most of the reported systems describe system evaluation methods, distribution architectures, and system interfaces that invoke classical geographical information system data, i.e., the metadata of a GIS library, not the content of the imagery that is provided from satellite and airborne image capture systems [7]. Some of these systems attempt to incorporate a combination of object detection and classifications to develop semantic information regarding the relationship between these objects and user needs [8]. Other papers describe specific details of the sub-elements of a retrieval system, e.g., the comparison of various similarity measures used [9], or the performance of a previously developed system [10].

In a geospatial library environment, searches generally produce results such as the fraction of queried cover type existing in a defined region, e.g., describing the coverage of city, urban, forest or crop type. Many CBIR methods for geospatial data attempt to produce a description of image primitives at the pixel level, e.g., based on local neighborhood structures, textures, or spectral content. Schroeder, et al., in Ref [11] describes an index based on pre-extracted content using pixel primitives. The authors demonstrate their method on Landsat Thematic Mapper (TM) and other data types that have coarse resolution (e.g., 30m per pixel) and their system uses a Bayesian-based method for probabilistic retrieval. Datcu, et al., in Ref [6] present a prototype concept for a content-based information mining system. Once again, the data presented is coarse (i.e., relatively low spatial content per pixel) and pixel-based primitives (i.e., features) are extracted for cover-type classification. While this method does describe a measure of local texture using stochastic models, the Landsat TM data does not provide the high spatial detail that we wish to consider. Stefanidis, et al., in Ref [12] and Agouris, et al., in Ref [13] describes a sketch-based image retrieval environment that moves away from the CBIR concept of indexing based on inherent texture or structural features (which are typically scale-dependent) and instead incorporates a drawing method that searches the database for edge primitives with similarity to the sketch, e.g., a sketch of an airplane used to locate similar structures on the ground. While this method moves away from the requirement of an “index image”, i.e.,

the initial image required to begin a search in a query-by-example system, it also does not allow for pre-indexing of image content since structures are located through a template matching procedure with the sketch. Although the method can be described as reasonably scale-independent (i.e., the scale being specified through the sketch), the intense processing requirements in template matching and the inability to generalize the search process across the data system limits the technique.

As the resolution of new imaging sources improves with new imaging platforms, the ability to automatically identify cover types – or more importantly structural details - by classifying pixels or local textures becomes problematic due to the highly-resolved, man-made and natural structures that are present in complex spatial arrangements. Spectral band and local pixel neighborhood content techniques (e.g., commonly used for texture feature description [14]) do not adequately account for the increased structural information that resides in higher-resolution imagery. Methods that have relied on coarser image content in the past must be modified or newly developed to optimally take advantage of increasing detail, i.e., they do not readily scale with resolution.

Fig. 1 demonstrates this point through several examples of the high-resolution imagery that will be used throughout this discussion. The indicated image regions represent a wide variety of cover types ranging from mixed deciduous and conifer forest lands to urban and industrial settings. At these resolutions and with the complex proximities of the various man-made and natural structures, it is difficult to apply pixel classification methods to segment image content. In general, pixel-based methods will result in many mixtures of discontinuous regions and ambiguity at the boundaries of different structural regions in the imagery.

At the Oak Ridge National Laboratory (ORNL) we are developing methods to automatically describe these region types in high-resolution imagery so that a large image library can be efficiently assembled and indexed to perform content-based retrievals that will accommodate searches for specific spatial structure. This system encompasses three main development areas: (1) a software agent architecture to support distributed computing and to gather image content and metadata from the web, (2) a geospatial data

modeling component to register the imagery in a consistent world-coordinate system, and (3) a RBIR component to index imagery for search and retrieval. In this paper we will focus primarily on the RBIR aspects of search and retrieval. In Section 2 we give a brief overview of the architecture of the archive generation system that has been developed. In Section 3 we review the critical components of our image region description and indexing approach. In Section 4 we describe the indexing methods used and the data structures that are developed to facilitate efficient storage and access. Finally, in Section 5 we present and discuss results obtained using the data set partially represented by Fig. 1, a total indexed land area of approximately 153 $km^2$ (59 $mi^2$) at 0.5m per pixel resolution.

## 2. Overview of System Architecture

At ORNL we have developed a system and architecture by combining novel approaches from three distinct research areas: software agents, georeferenced data modeling, and content-based image retrieval. The resulting technology represents a comprehensive image data management and analysis system. This system allows us to meet the challenges of organizing and analyzing large volumes of image data, and of automating the image consumption process to populate the database. The overall system approach breaks down into three components: (1) an innovative software-agent-driven process that can autonomously search through distributed image data sources to retrieve new and updated information, (2) a geo-conformance process to model the data for temporal currency and structural consistency to maintain a dynamic data archive, and (3) an image analysis process to describe and index spatial regions representing various natural and man-made cover types.

Fig. 2 represents the agent-based architecture of our design. There are five types of agents that are represented in this system. The Coordination Agent controls the workflow between the different agents. The Crawler Agent performs a depth-first search for image links on potential websites (in our case, only URL's ending with .edu, .gov and .net). The Download Agent downloads images for all the image links generated by the Crawler Agent. The Download Agent coordinates with the image repository to ensure that the image does not already exist in the repository or that the image is newer or has a higher resolution than the existing one.

The fourth type of agent is the Markup Agent. This type of agent creates XML files that have images marked up with their properties and metadata. For each image in the repository, this agent extracts image properties like height, width, bit planes, etc. In addition, this agent extracts geospatial information like the images bounding box coordinates from the accompanying metadata/world file. After collecting this information, it creates an XML file for each image in the image repository using all of the above-deduced properties. The XML files are then stored in a separate XML Repository.

Finally the fifth agent type, Extractor Agents, perform preprocessing of the images. Typically each Extractor Agent runs on a separate processor so that images can be processed in parallel. An image is first segmented into block segments of size 128×128 pixels. Once the image segments are created, a feature vector file describing each segment is created by making use of the image properties in the XML file and the feature extraction methods described below.

To deploy this agent architecture, we used the Oak Ridge Mobile Agent Community (ORMAC) framework [15]. This framework has been under development over the course of several agent-based research projects. ORMAC is a generic agent framework providing transparent agent communication and mobility across any Internet connected host [16].

The overall purpose of this agent system is to provide a comprehensive process for the continuous gathering, description, indexing, and datamining of a dynamic, high-resolution geospatial library. The remainder of this paper will focus primarily on the methods being developed for image feature description, sub-region segmentation, and indexing, i.e., once the geographic data has been located and brought into the system by the agent architecture.

## 3. Image Analysis

Once the imagery has been downloaded by the software agents, our goal is to generate a succinct description of an image-dependent number of contiguous areas. Fig. 3 provides an overview of the process.

Our approach begins with an image tile, for example of the size represented in Fig. 1. For this paper we are using image tiles that are 3100x3100 pixels representing a size of 1,750m on a side.

The tiles are tessellated into 128×128 pixel segments corresponding to 64m × 64m area. The segment size was determined heuristically by ensuring that various cover structure would be adequately represented in each segment. Fig. 4 shows examples in clockwise order from the upper left of four cover types: agricultural, forested, suburban, and industrial. A number of structure-oriented features are extracted from each segment. These features are reduced using a PCA/LDA method [17] to provide a short-length vector for segment clustering by a region growing procedure to organize similar segments into contiguous groups. Each contiguous group represents a sub-region in the original image tile and a summary feature description is generated for indexing. Also, the region boundary is run length encoded for efficient storage in the database. Finally, an indexing tree is developed using the region features by application of an approximate nearest neighbor (ANN) method as described in Section 4. The indexing tree provides $O[log_2(n)]$ retrieval efficiency from the database through a query-by-example RBIR.

### 3.1. Feature Analysis

For this research, we have focused on features that measure the texture and structure of the image segments. The spectral attributes of an image segment are also valuable and have been used by many researchers to classify cover types in geospatial data [6, 11, 18]. But over large geospatial extents, spectral information can unintentionally limit a query-based search to a confined region. This is demonstrated by considering the Landsat TM data shown in Fig. 5 (30m per pixel resolution). Although we are not using Landsat TM data for this study, the four regions show agricultural areas over a large geographical distance and include Maine, Virginia, Tennessee, and Florida. Although the same three spectral bands were used to visualize crop regions, the spectral content varies tremendously. The implication is that although it may be straight-forward to locate other agricultural regions in Maine using a Maine-based query composed of spectral features, it will be difficult to locate agricultural regions in Tennessee or Florida due to the

variation in the spectral content. To avoid this unintentional bias in our indexing and retrieval process, we have adapted two feature sets that rely primarily on edge information to describe texture and structure.

We characterize segment texture using local binary patterns (LBP) [19] and local edge patterns (LEP) [20]. In the rotation-invariant LBP texture operator, each 3×3 pixel neighborhood in the intensity image is thresholded by the intensity value of the center pixel. As there are eight neighboring pixels, each of which can be represented as a 1 or 0 (if above or below the center pixel value, respectively), it is evident that there are 256 ($2^8$) possible patterns that can result from this thresholding. Since, however, we desire rotational invariance, we note only those patterns that are unique under rotation. For example, the three patterns in Fig. 6 are all (approximately) equivalent under rotation about the center pixel. Applying this equivalence-under-rotation idea, it can be shown that there are only 36 unique patterns. This implies that every pixel in the image can be assigned a number from 1-36 depending upon its LBP. The 36-bin normalized distribution (i.e., histogram) of the LBP values in a given 128×128 image segment hence provides 36 features for that region.

The LEP is computed almost identically to the LBP, except that we examine 3×3 pixel neighborhoods in the image edge map rather than the image intensity values. When considering the edge map, we must also consider the state of the center pixel, which is a 1 if the center pixel is an edge, or a 0 if not. This doubles the number of potential patterns from 36 to 72 so that every pixel in the image can be assigned a number from 1-72 depending upon its LEP. The 72-bin normalized distribution of the LEP values hence provides 72 features.

To characterize structure in a segment, we analyze the distribution of edge orientations. The motivation to this approach is that man-made structures generally have regular edge features oriented in only a few directions (usually two for buildings) while natural image regions have randomly oriented edges. Different mixtures of man-made and natural structures will result in a variety of descriptions.

We compute local image orientation at each edge pixel using steerable filters [21]. We then find the 64-bin histogram of the edge orientation over angles from -90 to +90 degrees. The edge orientation distribution for a man-made structure is shown in the top of Fig. 7 (a) and that for the natural image is shown in the bottom of Fig. 7 (a). Note in the top of Fig. 7 (c) that there are two peaks in the edge orientation distribution near -80 degrees and +10 degrees that correspond to the orientations of the building. The distribution for the natural scene in the bottom of Fig. 7 (c) however, is approximately uniform. Since we require that the stored features be invariant to rotations of the source image, we next take the discrete or fast Fourier transform of the 64-point edge orientation histogram and keep the magnitude of the first 32 points as the final features. The magnitude of the DFT or FFT is invariant to circular shifts of the data.

The total number of texture and structure features used at this point is therefore 140 (i.e., 36+72+32). Subsequent to the feature extraction step we apply a PCA and LDA process that results in a reduction from 140 to 8 features per image segment. These features are the basis of geospatial clustering and indexing for search and retrieval.

### 3.2. Geospatial Clustering

Once the features of the image segments have been extracted, it is possible to use this feature vector as an index for retrievals. Since there are generally a large number of contiguous 128×128 segments that define a content-based region (e.g., forested, suburban, etc.), we seek to group neighboring segments with similar features together to form a sub-region within an image tile or tiles. We perform a geospatial clustering procedure using a region growing technique to connect large contiguous and homogeneous segments of similar structure and texture characteristics.

Region growing is initialized by randomly selecting a seed segment at location $(x,y)$, where $(x,y)$ designates a coordinate of the corner or centroid of a segment. A segment with feature vector $\boldsymbol{v}(x,y)$ is merged with a neighboring segment with feature vector $\boldsymbol{v'}(x\pm1, y\pm1)$, or with a segment group with mean vector $\langle\boldsymbol{v'}\rangle$ if,

- the coordinate of the neighboring segment or of the closest segment group is an element of the set $\{(x\pm1, y\pm1)\}$,
- $|\boldsymbol{v}-\boldsymbol{v'}| < T_1$ or $|\boldsymbol{v}-\langle\boldsymbol{v'}\rangle| < T_1$, where $T_1$ is a user-specified threshold,
- the resulting variance, $\sigma^2$, of the new segment group is less than $T_2$, where $T_2$ is a user-specified threshold used to limit the variance.

The merging process is continued until all the segments in the image tile have been tested. A subsequent step is then performed on this data to reevaluate smaller sub-regions for connectivity. During this second stage, a region merging operation is performed to join independent segments to existing sub-regions if,

- the segment is spatially connected to one or more sub-regions (i.e, a sub-region being defined as > 1 connected segment from the initial clustering pass)
- the feature vector distance, $|\boldsymbol{v}-\langle\boldsymbol{v'}\rangle|$, between the segment and the sub-region is the lowest among all of the independent sub-regions neighboring the segment, and,
- the resulting variance of the newly merged region does not exceed a user-defined threshold, i.e., $|\boldsymbol{\sigma}^2| < T_3$.

Once the contiguous segment regions have been determined, each segment group (image sub-region) has 16 descriptive features associated with it, i.e., each sub-region is described by vector $\boldsymbol{w} = (\langle f_1\rangle, \langle f_2\rangle, \ldots, \langle f_N\rangle\ldots, \sigma_1^2, \sigma_2^2, \ldots, \sigma_N^2)^t$, for $N$=8, where $\langle f_n\rangle$ is the average of the $n$-th feature across the ensemble of segments in that group, and $\sigma_n^2$ is the corresponding variance of that feature. It is this sub-region description that is used for indexing and retrieval in the RBIR library. Fig. 8 shows typical results of this merging process.

## 4. Database Indexing

The goal of indexing is to organize the image data (e.g., filenames, features, indexing codes, etc.) in the database such that a ranked list of nearest neighbors can be efficiently retrieved in response to a query without performing an exhaustive comparison to all the records in the database. For our RBIR system this

is achieved by generating a binary decision tree of the image features, $\boldsymbol{w}$, described above. A *bin* is defined as a bottom-level element in our tree structure, sometimes described as a "leaf" or terminal node, that contains a small list of images, e.g., a bottom-level bin may contain a list of image vectors $\{\boldsymbol{w}_a, \boldsymbol{w}_b, \boldsymbol{w}_c, \ldots\}$. Under our RBIR architecture, a query vector is compared at the top level to each of two sub-nodes and a decision is made as to which sub-tree to take. There are many ways to implement decision trees. For this work we have implemented an ANN indexing and search method that builds on *kd-tree* methods [22, 23]. Whereas an exhaustive nearest-neighbor search of the $n$ vectors (i.e., images) in the database would be of $O(n)$ computations, the *kd-tree* approach is of $O(log_2(n))$. This is explained through Fig. 9 that shows a simple example of a two-dimensional feature space, $(f_x, f_y)$, containing 18 image vector points partitioned into a *kd-tree* structure where each bin, or leaf node, contains 3 points (i.e., image vectors). The *kd-tree* method allows for the rapid retrieval of the closest bin to the query point, $\boldsymbol{Q}$, but the data in this bin are not necessarily the closest points and the nearest-neighbor result can be in error by an amount, $\varepsilon$.

The ANN method incorporates a search window that results in the collection of neighboring bins about the query point. As this window increases in radius, the nearest neighbor error, $\varepsilon$, decreases, but the performance of the system also decreases to $O(n)$. The efficiency of the ANN method is proportional to $O((1/\varepsilon)^{N/2} log(n))$, where $N$ is the dimension of the feature space, $n$ is the number of data points (i.e., indexed regions in the system), and $\varepsilon$ is the nearest neighbor error [22]. The nearest-neighbor error is therefore inversely proportional to the size of the search window as shown in Fig. 9. As the radius of the search window increases, neighboring bins containing additional image vectors are included in the final nearest-neighbor search. As the radius continues to grow, the system approaches the complexity of an exhaustive nearest-neighbor search. Therefore, the accuracy of the RBIR system is selectable as a trade-off between nearest neighbor performance and computational efficiency. Through this approach we have demonstrated retrieval efficiencies on the order of 5 seconds for 100,000 indexed images, which scales to on the order of 15 seconds for a database of 1,000,000 images [23].

Once the image features and their corresponding bins in the *kd-tree* have been determined, the image data structure is added to the database as shown schematically in Fig. 10. Each image tile in the data set is

comprised of a previously undetermined number of sub-regions. In the database, a unique identifier, $UID_n$, is assigned to each image sub-region (i.e., a segment cluster), following this is the path to the image tile in which it resides, the run-length-encoded (RLE) boundary of the sub-region, and the set of features and the *bin* code for the tree search. Also note in the figure that multiple sets of features and bin codes are represented. With this RBIR architecture, we can easily append the system to include other sets of features. For example, we may add spectral features to the query process or we may include other metadata associated with the imagery to augment query flexibility. Fig. 11 shows this schematically in terms of a Venn diagram. In our previous implementation of this system for industrial applications [23, 24], a query is performed against each discrete category of features for which an independent indexing tree has been generated. The returned vectors that overlap these sets (i.e., the reduced set representing the Boolean intersection in Fig. 11) are concatenated and ranked for display relative to their distance from the query vector, $d_n(\mathbf{Q},\mathbf{w}_n)$, where $d_n(\mathbf{Q},\mathbf{w}_n) = \| \mathbf{Q} - \mathbf{w}_n \|$. A similarity value is determined based on this distance as $S_n(\mathbf{Q},\mathbf{w}_n) = 1-d_n(\mathbf{Q},\mathbf{w}_n) / \sqrt{N}$, where $N$ is the dimension of the concatenated feature space and the metric ranges from [0,1], with 0 being most dissimilar and 1 being most similar.

## 5. Results

The results presented here are for a geospatial library composed of 50 image tiles (3500x3500 pixels, 1750m×1750m, 0.5m per pixel) representing approximately 153 $km^2$ of land in and around the U.S. Department of Energy's Oak Ridge Reservation [25, 26]. For this demonstration, the region (i.e., over all tiles) was tessellated into 39,200 segments of size 128×128 pixels each. Features were then extracted for each of the segments, which were subsequently clustered as described in Sections 3.1 and 3.2 above. The number of sub-regions developed through geospatial clustering was 4,810, resulting in a reduction of 88% in the number of unique, spatially distinct sub-region objects indexed for retrieval. Fig. 12 shows a composite view of the entire geospatial library in (a) along with a representation of the sub-region segmentation achieved through clustering of the image segments in (b). Note that many of the large, contiguous areas of forest and water regions have been clustered, therefore reducing the complexity of the database (e.g., through the reduction of the number unique indices that must be maintained) and the simplification of the search environment for the user, e.g., by reducing the need to view many image

segments of nearly identical content. Also note that the urban and industrial region types have maintained higher granularity in the system. These clustered regions are typically smaller due to their more complex and dissimilar structural content. This also benefits the user when a detailed search for subtle differences in man-made structure is the query goal.

For demonstration purposes, we have indexed the original 39,200 segments in one descriptive dataset, and the 4,810 sub-regions in another dataset, where each set could then be queried as independent systems and compared. Fig. 13 shows an example retrieval of several different cover types at the image segment level. In the figure, each of the images in the left-hand column of each row, (a) – (d), represents the query index. The remaining five images in each row represent the top five matching results of the query for an industrial complex in (a), suburban area in (b), an agricultural area in (c), and a specific search for striped parking lots and roadways in (d). Note the generality and flexibility of the system to locate imagery of an extreme variety of cover types and detail using only eight structure-based features (from the original 140).

Although this type of segment-level query can be useful to an analyst, it can also provide too much redundant information regarding large spatial extents of visually similar imagery. For example, forested or agricultural regions may occupy a large fraction of the data library in a rural region such as exists in this dataset. An ability to collect together these large geospatial regions, or alternatively, to search for small man-made structures of a particular type distributed throughout large contiguous rural areas (i.e., a needle in a haystack) would also be useful. Therefore, in Fig. 14, we show examples of two queries performed on the geospatially clustered regions of the image library, i.e., the second dataset. In these screen shots, the query image is located in the upper left-hand corner of the display. In (a) we see the results of a query based on an image sub-region of an industrial complex. The returned images are ordered in terms of similarity, $S_n(\boldsymbol{Q},\boldsymbol{w_n})$, from left to right and top to bottom and range over the first 10 returned images from 92% to 85% similarity. In (b) we show the results of a query made using a large, contiguous deciduous forest region. For this example, the similarity of the first 12 returned images ranges from 92% to 88%. Notice the difference in geospatial extent of the results returned by each query. In the case of the industrial complex query, the size of the average sub-region is around 12 segments, while the size of the forested sub-

regions is in the hundreds of segments. This demonstrates the ability to query the library over both large and small geospatial extents providing both filtering of large, similar ranges of cover type, and small detailed regions of more complex man-nade structures. Also, note that the similarity between regions (of any type) is a function of the structural properties of the entire sub-region and not the area or shape of the sub-region. Although it is not clear that shape and area are relevant descriptive features for this system today, these or other features could easily be incorporated into the system in the future according to our discussion in Section 4 (e.g., see Fig. 11).

## 6. Conclusions

In this paper we have presented a novel method for automated feature extraction, spatial clustering, and indexing of a large geospatial image library. The ability to describe broad structural content in high-resolution imagery in a succinct format (i.e., 8 features) and cluster similar content to form large, contiguous image sub-regions is the unique contribution of this work. Although retrieval experiments were described for a relatively small geospatial data set (153 $km^2$), the system architecture and processing methodology have been developed to facilitate very large data libraries that can be maintained and updated in a dynamic manner through distributed computing within a software agent architecture. The feature analysis and indexing approach used in this research provides an efficient and flexible method for describing a broad range of cover types while allowing a user to locate very specific structural detail in a query-by-example environment. Future work in this area will include the incorporation of other geographical information metadata into the query process along with the addition of spectrally-based features for augmenting the specificity of local searches within a geospatial region of interest. Our indexing procedure allows us to consider spatial relationships between sub-regions, which will also be investigated to strengthen the query process. Finally, the infrastructure for detecting temporal changes is a fundamental element of our architecture. In future work, we will take advantage of this system characteristic to automate/assist in the detection of geospatial change.

## 7. References

1. Moran, M.M., (2000), “Image-Based Remote Sensing for Precision Crop Management – A Status Report,” Proceedings of Space 2002: the Seventh International Conference and Exposition on Engineering, Construction, Operations and Business in Space, p. 185-193.

2. Tapley, B.D., et al., (2001), “A Vision for Creating Advanced Products From EOS Core System Data to Support Geospatial Applications in the State of Texas,” International Geoscience and Remote Sensing Symposium, Vol.2, p. 843-845.

3. You, J., et al., (2004), “On Hierarchical Content-Based Image Retrieval by Dynamic Indexing and Guided Search,” Proceedings of SPIE, Storage and Retrieval Methods and Applications for Multimedia, Vol. 5307, p. 559-570.

4. Santini, S., (2001), Exploratory Image Databases, Content-based Retrieval, Academic Press, San Fransisco, CA.

5. E. Vicario (1998), Editor, Image Description and Retrieval, Plenum Press, New York, NY.

6. Datcu, M., et al., (2003), “Information Mining in Remote Sensing Image archives: System Concepts,” *IEEE Trans. On Geoscience and Remote Sensing*, Vol. 41, No. 12, Dec. 2003, p. 2923-2936.

7. Hsu, R.C., et al., (2004), “A Mobile Agent Based Image Retrieval and Delivery System for Remote Sensing Satellite,” Proceedings of the SICE, SICE Annual Conference, p. 2227-2232.

8. Grant, C.W., (2004), “Data Structures and Algorithms for Graph Based Remote Sensed Image Content Storage and Retrieval, *IEEE International Geoscience and Remote Sensing Symposium*, Vol. 3, p. 2159-2162.

9. Bao, Q., et al., (2004), “Comparative Studies on Similarity Measures for Remote Sensing Image Retrieval,” *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Vol. 1, p. 1112-1116.

10. Daschiel, H., et al., (2005), “Information Mining in Remote sensing Image Archives: System Evaluation,” *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 43, No. 1, p. 188-199.

11. Schroder, M., et al., (2000), “Interactive Learning and Probabilistic Retrieval in Remote Sensing Image archives,” *IEEE Trans. on Geoscience and Remote Sensing*, Vol. 28, No. 5, Sept., p. 2288-2298.

12. Fritsch, D., et al., (1998), “Sketch-Based Image Retrieval in an Integrated GIS Environment,” ISPRS Commission IV Symposium on GIS – Between Vision and Applications, Germany, Vol. 32/4.

13. Agouris, P., et al., (1999), “An Environment for Content-Based Image Retrieval from Large Spatial Databases,” *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 54, p. 263-272.

14. Newsam, S., et al., (2004), “Using Texture to Analyze and Manage Large Collections of Remote Sensed Image and Video Data,” Applied Optics, Vol. 43, No. 2, p. 210-217.

15. Potok, T., et al., (2003), "VIPAR: Advanced Information Agents Discovering Knowledge in an Open and Changing Environment," Proc. 7th World Multiconference on Systemics, Cybernetics and Informatics, Orlando FL, pp. 28-33, July 27-30.

16. Yang, Y., et al., (2000), “A Mobile Agent-Based Architecture fro On-Demand Processing of Remote-Sensing Archive,” Proc. Of the 4th International Conference on High-Performance Computing, IEEE Computer Society Press, p. 538-541.

17. Bingham, P.R., et al., (2004), “Semiconductor Sidewall Shape Estimation,” *SPIE Journal of Electronic Imaging*, Vol. 13, No. 3, July.

18. Harvey, N.R., et al., (2002), “Comparison of GENIE and Conventional Supervised Classifiers for Multispectral Image Feature Extraction,” *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 40, No. 2, Feb., p. 393-404.

19. Pietikainen, M., et al., (2000). “Rotation-invariant texture classification using feature distributions,” *Pattern Recognition*, Vol. 33, No.1, pp. 43-52.

20. Yao, C.-H. and Chen, S.-Y., (2003), “Retrieval of translated, rotated and scaled color textures,” *Pattern Recognition*, Vol. 36, pp. 913-929, No. 4.

21. Freeman, W. and Adelson, E. (1991), “The design and use of steerable filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 9, pp. 891-906.

22. Arya, S., et al., (1994), “An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions”, Proc. of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 573-582.

23. Tobin, K. W., et al., (2002), "Content-based Image Retrieval for Semiconductor Process Characterization", *EURASIP Journal on Applied Signal Processing*, Special Issue on Applied Visual Inspection, Vol. 2002, No. 7.

24. Bingham, P.R., et al., (2004), "Semiconductor Sidewall Shape Estimation," *SPIE Journal of Electronic Imaging*, Vol. 13, No. 3.

25. Tuttle, M., and Pace, P., (1996), "ORNL Basemapping and Imagery Project: Data Collection, Processing, and Dissemination," Geographic Information System (GIS) Environmental management conference, Reno, NV, CONF-9603148-1.

26. Tiled Data Based Dictionary (1998), OREISE Base Mapping Project, Oak Ridge Environmental Information System, http://www-oreis.bechteljacobs.org/oreis/help/oreishome.html

# Figure Captions

**Fig. 1. Examples of a wide variety of spatial data regions that may exist in a large geospatial image database including, (a) forest, (b) agriculture, (c) water structure (locks, dams, etc.), (d) urbanized area, and (e) industrial sites. The resolution of these scenes are 0.5m per pixel.**

**Fig. 2. Schematic representation of the agent architecture.**

**Fig. 3. Process flow shows tessellation of the input tile, feature extraction, segment clustering, indexing, and database building for query-based search and retrieval.**

**Fig. 4. Example image segments representing four cover types.**

**Fig. 5. Examples from the Landsat Thematic Mapper showing variation in spectral response across large geospatial extents. Three of six spectral bands have been selected for display that emphasize variations in crop cover. (UL) Maine. (UR) Virginia. (LL) Tennessee. (LR) Florida.**

**Fig. 6. Three local binary patterns (LBP) that are equivalent under rotation about the center pixel.**

**Fig. 7. (a)Image segment, (b) edge map, and (c) edge orientation of (b). The top row represents a man-made structure and the bottom row a natural scene.**

**Fig. 8. Region growing results across three tiles from the image library. Each bordered region represents one homogeneous, connected group of segments as determined by their texture and structure features.**

**Fig. 9. An example of a simple feature space showing a *kd-tree* bin structure for an ANN search region about a query point, *Q*.**

**Fig. 10. Schematic diagram of database structure for the RBIR system.**

**Fig. 11. Diagram showing the search relationship between feature categories and metadata in the RBIR system.**

**Fig. 12. Composite image in (a) of the 50 image tiles comprising the 153 km$^2$ geospatial library. The composite in (b) shows the 4,810 sub-regions determined by spatial clustering the original 39,200 segments.**

**Fig. 13. Examples of the retrieval of image segments. The query images are in the left column. (a) an industrial complex, (b) a suburban setting, (c) agriculture, and (d) striped pavement.**

**Fig. 14. Examples of the retrieval of geospatial clusters. (a) An industrial complex, (b) a large expanse of deciduous forest, (c) an agricultural region.**

(a)
(b)
(c)
(d)
(e)

Fig. 1

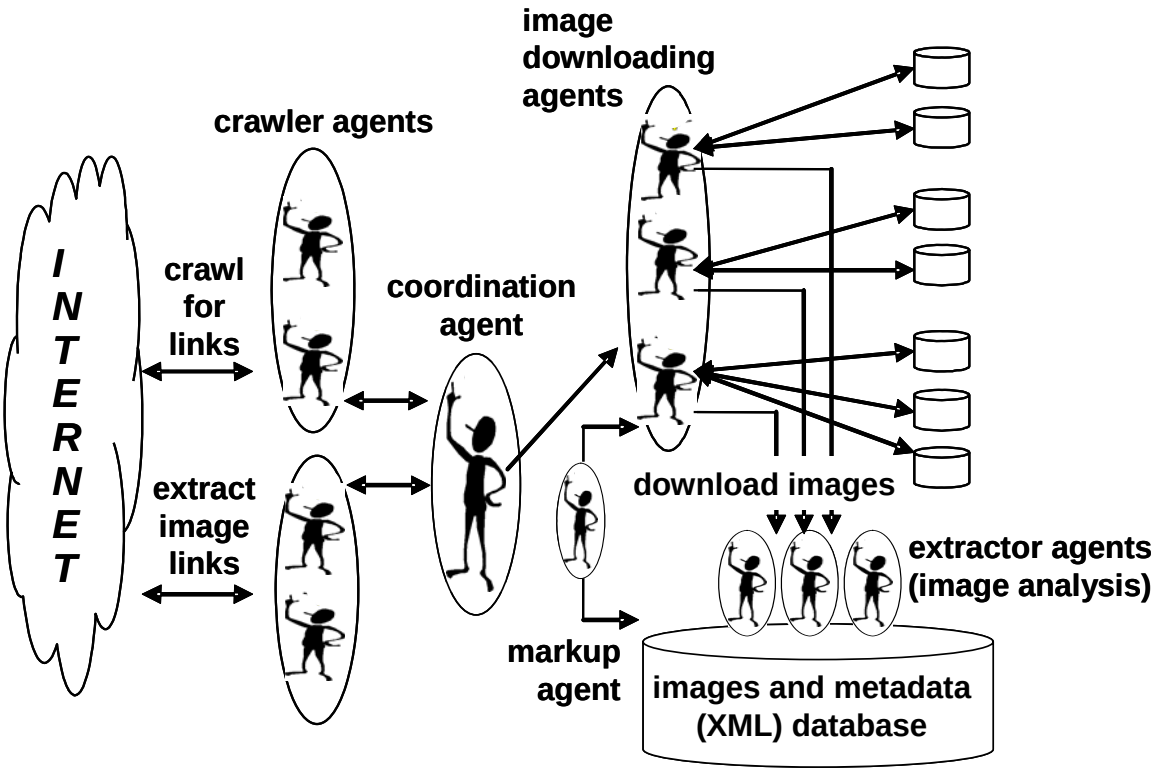
image downloading agents
crawler agents
INTERNET
crawl for links
coordination agent
extract image links
download images
extractor agents (image analysis)
markup agent
images and metadata (XML) database

Fig. 2

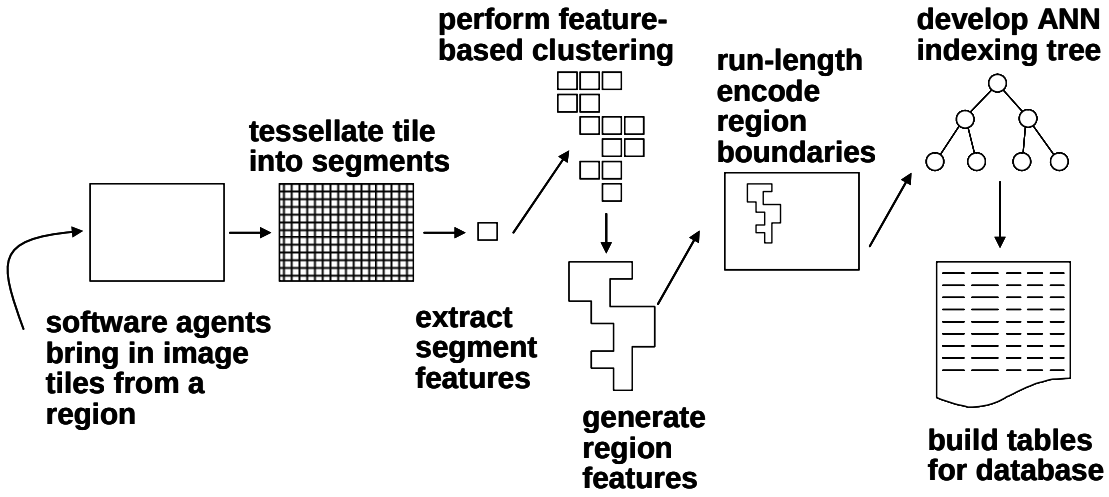
perform feature-based clustering
develop ANN indexing tree
run-length encode region boundaries
tessellate tile into segments
software agents bring in image tiles from a region
extract segment features
generate region features
build tables for database

Fig. 3

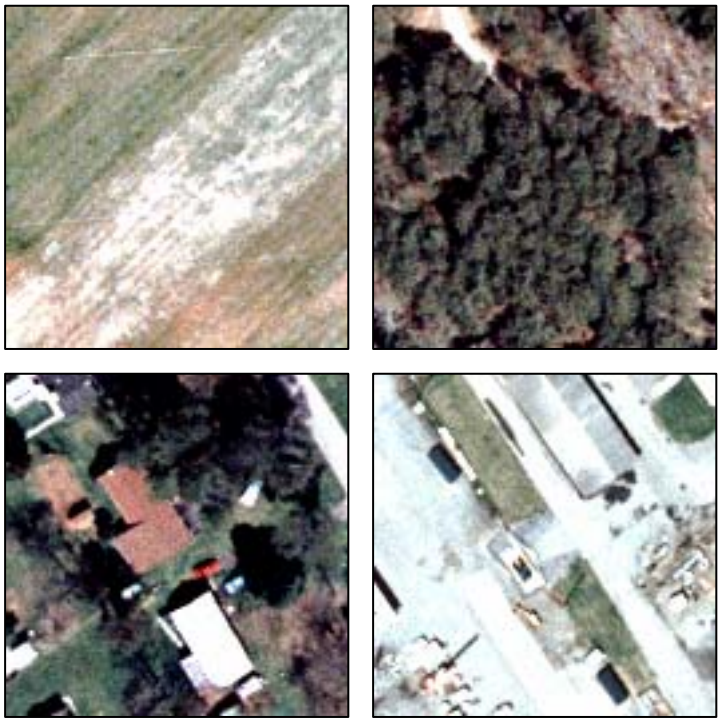

Fig. 4

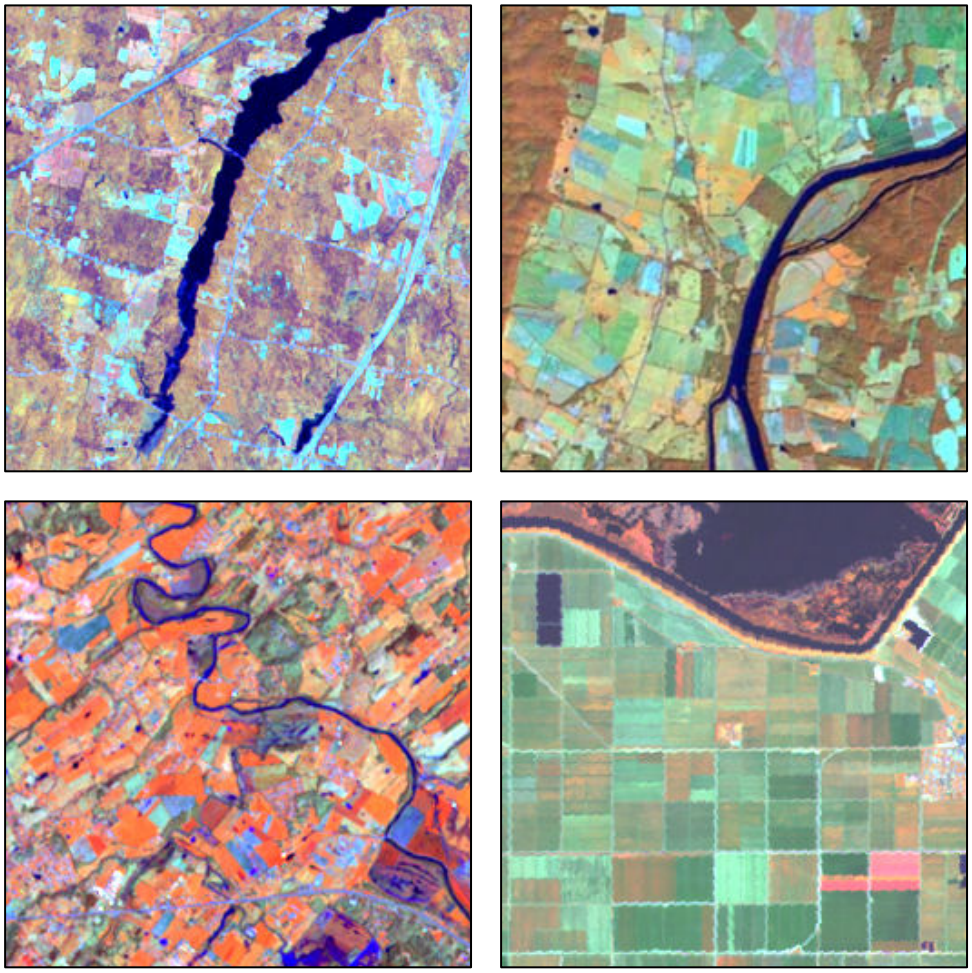

Fig. 5

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | | 0 |
| 1 | 0 | 0 |

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | | 0 |
| 0 | 0 | 0 |

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | | 0 |
| 0 | 0 | 0 |

Fig. 6

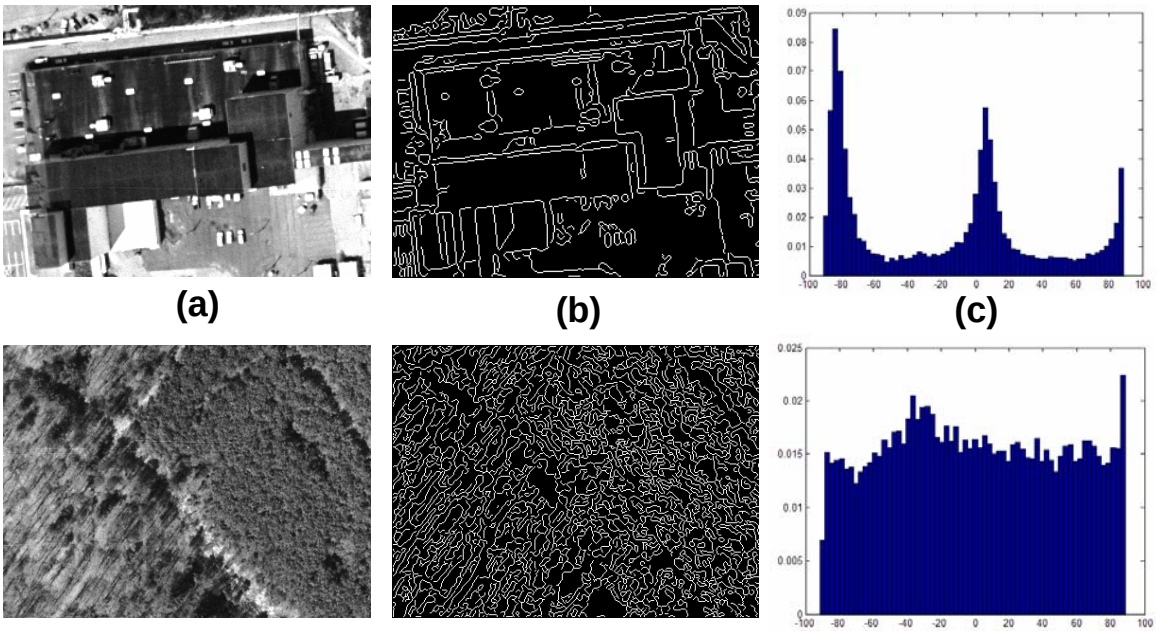


Fig. 7

Fig. 8

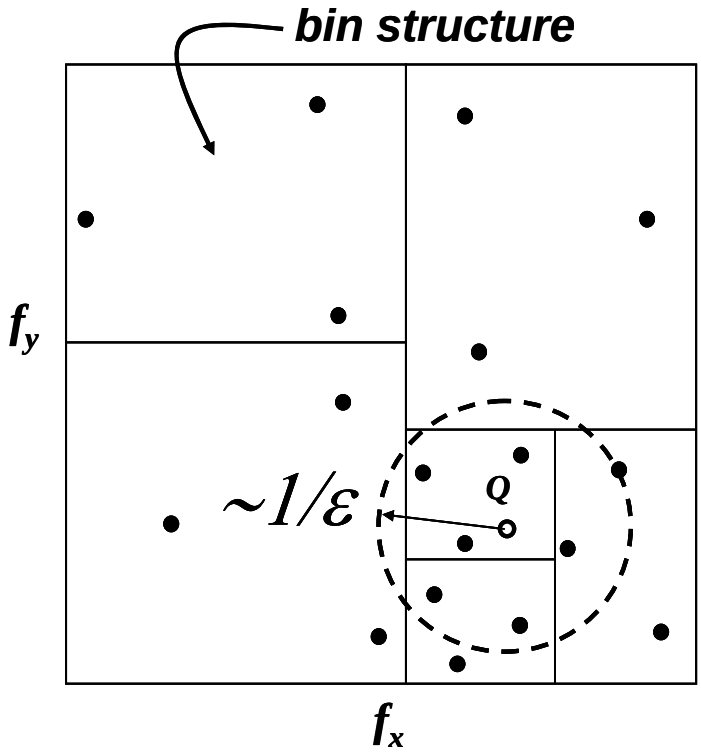
bin structure
$f_y$
$\sim 1/\varepsilon$
Q
$f_x$

Fig. 9

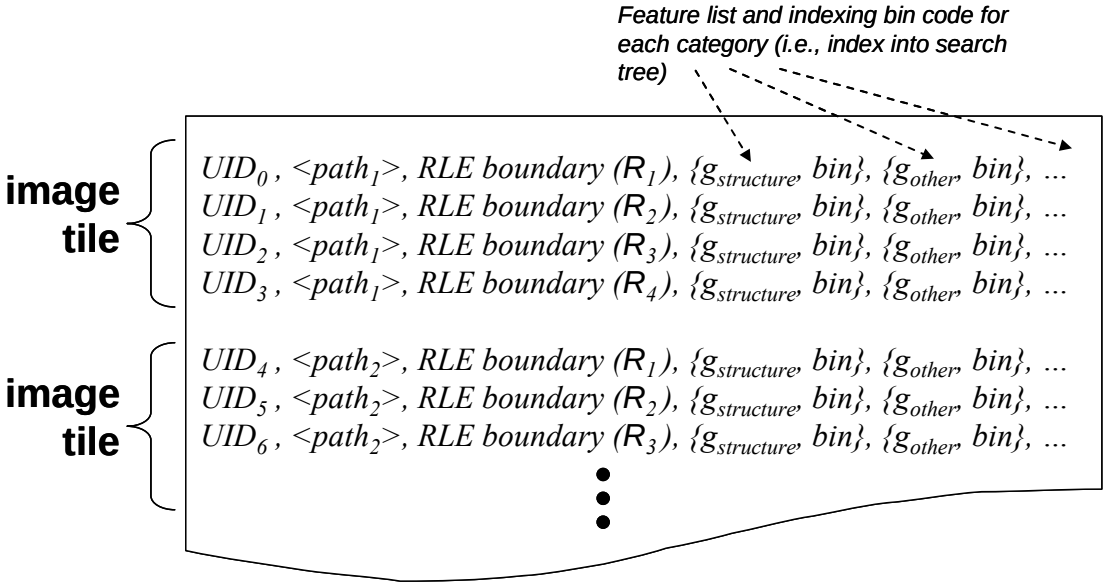
Feature list and indexing bin code for each category (i.e., index into search tree)
image tile
$UID_0$ , <$path_1$>, RLE boundary ($R_1$), {$g_{structure}$, bin}, {$g_{other}$, bin}, ...
$UID_1$ , <$path_1$>, RLE boundary ($R_2$), {$g_{structure}$, bin}, {$g_{other}$, bin}, ...
$UID_2$ , <$path_1$>, RLE boundary ($R_3$), {$g_{structure}$, bin}, {$g_{other}$, bin}, ...
$UID_3$ , <$path_1$>, RLE boundary ($R_4$), {$g_{structure}$, bin}, {$g_{other}$, bin}, ...
image tile
$UID_4$ , <$path_2$>, RLE boundary ($R_1$), {$g_{structure}$, bin}, {$g_{other}$, bin}, ...
$UID_5$ , <$path_2$>, RLE boundary ($R_2$), {$g_{structure}$, bin}, {$g_{other}$, bin}, ...
$UID_6$ , <$path_2$>, RLE boundary ($R_3$), {$g_{structure}$, bin}, {$g_{other}$, bin}, ...

Fig. 10

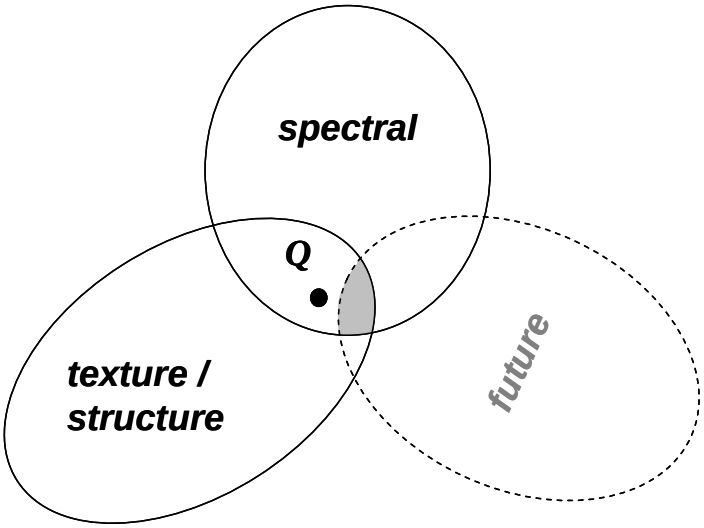


Fig. 11

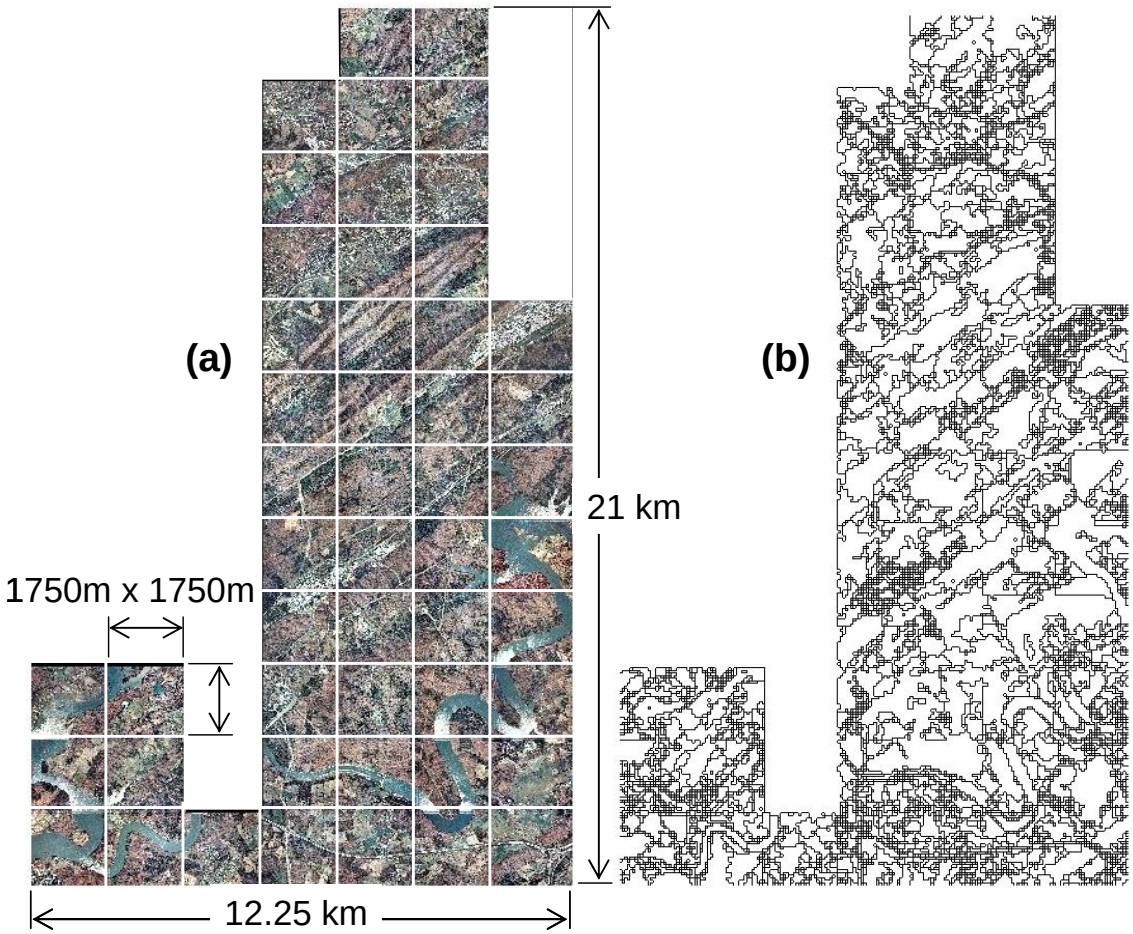


Fig. 12

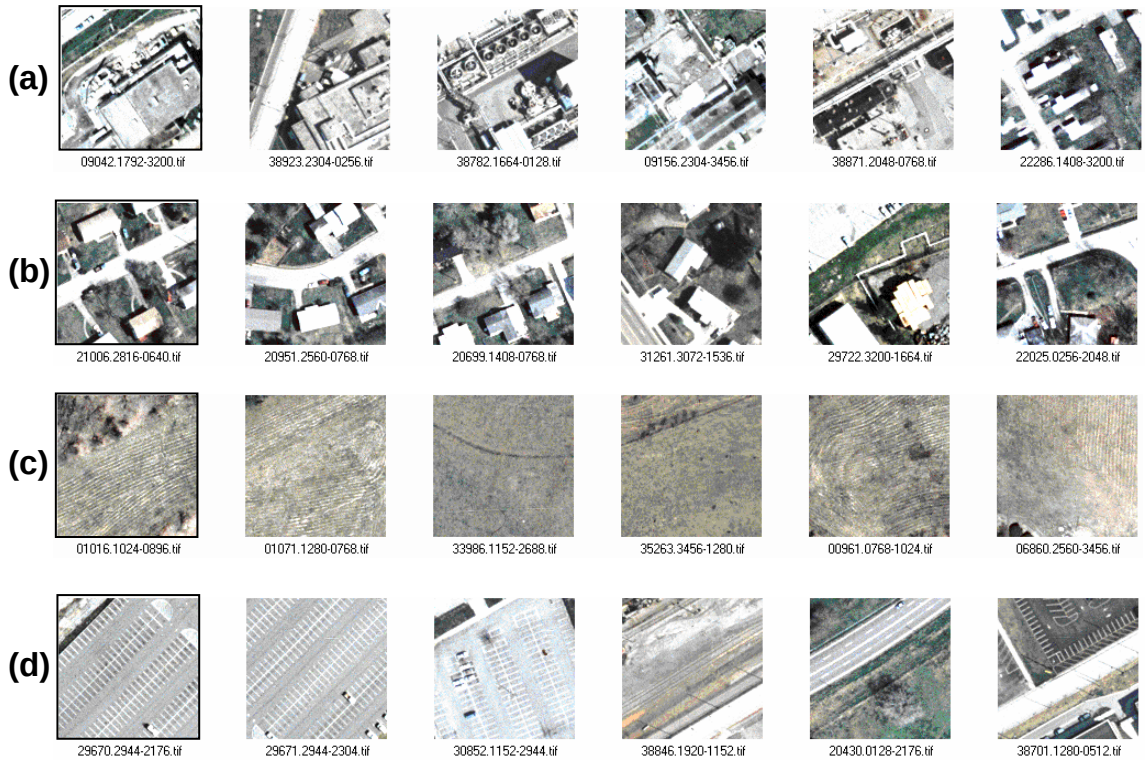
(a)
09042.1792-3200.tif
38923.2304-0256.tif
38782.1664-0128.tif
09156.2304-3456.tif
38871.2048-0768.tif
22286.1408-3200.tif
(b)
21006.2816-0640.tif
20951.2560-0768.tif
20699.1408-0768.tif
31261.3072-1536.tif
29722.3200-1664.tif
22025.0256-2048.tif
(c)
01016.1024-0896.tif
01071.1280-0768.tif
33986.1152-2688.tif
35263.3456-1280.tif
00961.0768-1024.tif
06860.2560-3456.tif
(d)
29670.2944-2176.tif
29671.2944-2304.tif
30852.1152-2944.tif
38846.1920-1152.tif
20430.0128-2176.tif
38701.1280-0512.tif

Fig. 13

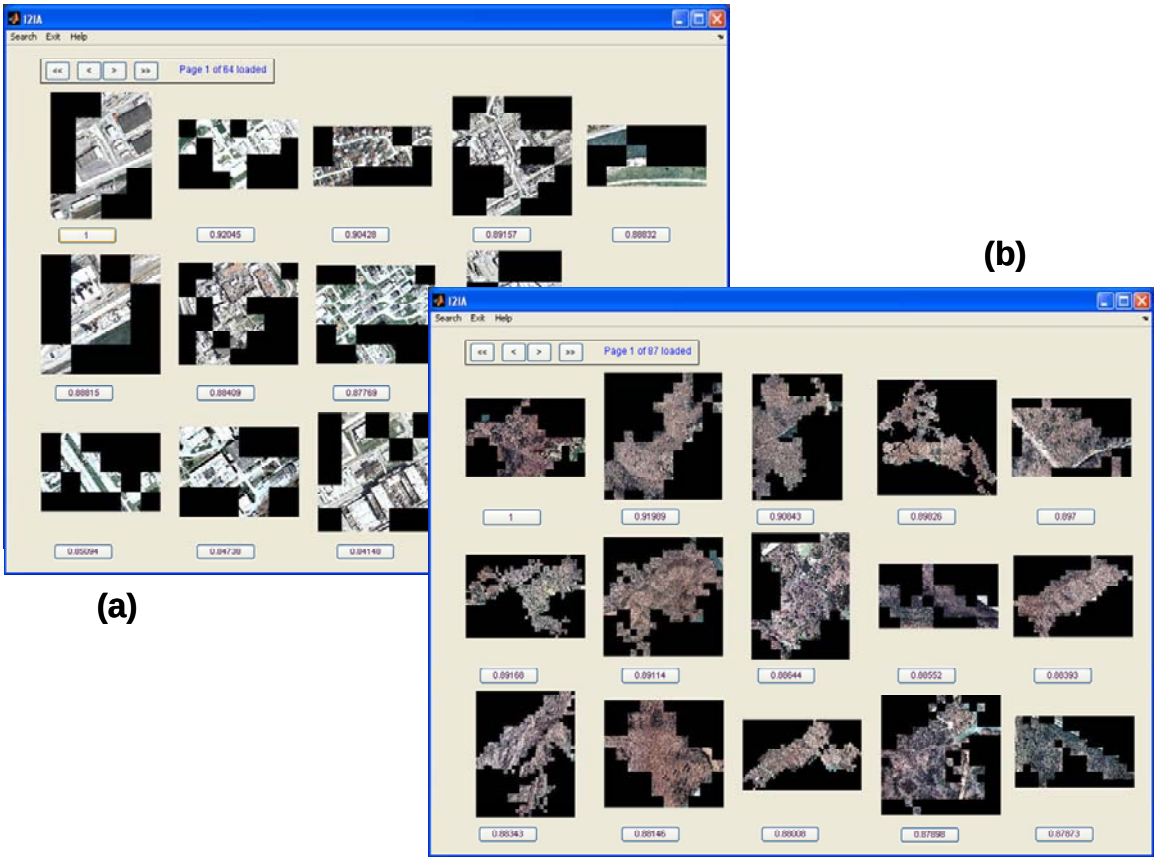
I2IA
Search Exit Help
Page 1 of 64 loaded
1
0.92045
0.90428
0.89157
0.88832
0.88815
0.88409
0.87769
(a)
(b)
I2IA
Search Exit Help
Page 1 of 97 loaded
1
0.91909
0.90043
0.89026
0.897
0.89160
0.89114
0.88644
0.88552
0.88393
0.88343
0.88146
0.88008
0.87898
0.87873

Fig. 14