Vegetation Identification Based on Satellite Imagery

Vamsi K.R. Mantena
Old Dominion University

Ramu Pedada
Old Dominion University

Srinivas Jakkula
Old Dominion University

Yuzhong Shen
Old Dominion University, yshen@odu.edu

Jiang Li
Old Dominion University, jli@odu.edu

See next page for additional authors

Original Publication Citation

Mantena, V. K. R., Pedada, R., Jakkula, S., Shen, Y., & Li, J. (2008) Vegetation identification based on satellite imagery. In H. R. Arabnia (Ed.), *Proceedings of the 2008 International Conference on Image Processing, Computer Vision & Pattern Recognition, IPCV 2008* (pp. 245-251). CSREA Press.

VEGETATION IDENTIFICATION BASED ON SATELLITE IMAGERY

Vamsi K. R. Mantena, Ramu Pedada, Srinivas Jakkula, Yuzhong Shen and Jiang Li

Old Dominion University
Department of Electrical and Computer Engineering, Norfolk, VA 23529
{vmant002, rpeda003, sjakk001, yshen, jli}@odu.edu

ABSTRACT

Automatic vegetation identification plays an important role in many applications including remote sensing and high performance flight simulations. This paper presents a method to automatically identify vegetation based upon satellite imagery. First, we utilize the ISODATA algorithm to cluster pixels in the images where the number of clusters is determined by the algorithm. We then apply morphological operations to the clustered images to smooth the boundaries between clusters and to fill holes inside clusters. After that, we compute six features for each cluster. These six features then go through a feature selection algorithm and three of them are determined to be effective for vegetation identification. Finally, we classify the resulting clusters as vegetation and non-vegetation types based on the selected features using a multi-layer percetron (MLP) classifier. We tested our algorithm by using the 5-fold cross-validation method and achieved 96% classification accuracy based on the three selected features.

Index Terms— clustering, classification, vegetation identification, texture synthesis, feature selection.

1. INTRODUCTION

High performance flight simulations require realistic terrain textures. For example, vegetation is required to change along with different seasons. However, current synthesis methods do not allow quick seasonal adaptations. An advanced texture synthesis technique with the capability of automatic seasonal adaptation for vegetation usually consists of the following two steps: 1) it identifies vegetation land cover from the imagery and 2) automatically synthesizes texture to adapt the identified vegetation area to a different season. We focus on the first step in this paper. Vegetation area identification from satellite imagery includes image pixel clustering and cluster classification. We use the ISODATA algorithm [1] for the clustering task and utilize an MLP classifier to classify the clustered areas as vegetation and non-vegetation areas, based upon a set of selected features computed from the imagery. Note that the ISODATA algorithm is an unsupervised learning process and land cover types of the resulting clusters are unknown. Therefore, the classification step is necessary.

ISODATA is a divisive process which divides data into homogeneous clusters. There are two data passes in each iteration of the ISODATA algorithm. The first pass (multiple interactions are usually needed) finds the means for each of the predefined number of clusters. The clusters are then split and merged in the second pass, depending upon their standard deviations, distance between means, cluster size and total number of clusters. Once the stop criterion has been met, each data point is clustered into one of the predetermined clusters in terms of their distances to the cluster means. Usually, a post-processing step for smoothing cluster boundaries is needed due to noise in the image [2]. We apply morphological operations on the clustered images to smooth the boundaries between clusters and to fill holes inside clusters.

The next step is to identify the land cover type for each of the clusters. We compute a set of statistics for each of the clusters and train a classifier to classify the clusters into one of the land cover types. The statistics include mean and standard deviation from each spectral channel of the image. Note that not all the statistics are useful for the classification, we thus utilized a feature selection algorithm to identify most effective features for the classification. Finally, the classification results (vegetation or non-vegetation) are coded into the least significant bit of each pixel, and the coded image is passed to the next stage where vegetation adaptation is performed.

The paper is organized as follows. In section 2, we introduce the ISODATA algorithm. Several morphological operations used for post-processing, feature selection and classification algorithms are also addressed. Section 3 presents some of the clustering and classification results. We discuss the results in section 4 and conclude this paper in section 5.

2. METHODS

2.1. ISODATA Clustering Algorithm

Commonly used clustering algorithms in remote sensing include the K-means algorithm and the ISODATA algorithm [1]. ISODATA stands for Iterative Self-Organizing Data Analysis, an advanced algorithm that can automatically adjust the number of clusters to be clustered. Both algorithms follow iterative procedures. In general, a clustering algorithm first assigns data points to arbitrary initial clusters. After that, the cluster means are recomputed and data points are reassigned. This process is repeated until the "change" between two consecutive iterations is negligibly small. The "change" can be defined in several different ways, either by measuring the mean distances difference from one iteration to another or by the percentage of pixels that change cluster membership in consecutive iterations.

The ISODATA algorithm is similar to the K-means algorithm that ISODATA dynamically adjusts cluster numbers while K-means assumes that the number of clusters is fixed and is known a priori. The ISODATA algorithm adaptively adjusts the number of clusters by splitting and merging clusters. Clusters are merged if either the number of pixels in a cluster is less than a certain threshold or if the centers of two clusters are closer than a certain threshold. A cluster is split into two different clusters if the cluster's standard deviation exceeds a predefined value and the number of pixels is as twice as the threshold set for the minimum number of members.

We found that clustering imagery based on YCbCr color space can give a better result than that on the RGB color space. The RGB color space of the image is therefore converted to YCbCr color space [3] which represents luminance, blue and red components before applying the Isodata algorithm.

The ISODATA algorithm requires a set of pre-defined parameters to control its behavior [1]. These parameters were chosen experimentally in this paper. We list those parameters and the algorithm as follows,

K = Number of clusters desired
I = Maximum number of iterations allowed,
P = Maximum number of pairs of clusters which can be merged,
Θ_N = Minimun no. of samples in each cluster (used for discarding clusters),
Θ_S = Maximum standard deviation for each cluster (used for split operation),
Θ_C = Minimum piarwise distances between cluster centers (used for merge operations).

The following steps explain the Isodata algorithm in detail.

Step 1:
We choose arbitrarily k (not necessarily equal to K) initial cluster centers: $m_1, m_2, \ldots, m_k$ for the data set $x_i, i = 1, 2, ..., N$. Here N represents the total number of samples in the image to be clustered. In this step the cluster centers are selected randomly from the samples of the image.

Step 2:
In this step each of the N individual samples is assigned to the closest cluster center. This step can be explained by the following equation,

$$x \epsilon \omega_j \quad \text{i}f D_L(x, m_j) = min D_L(x, m_i), i = 1, 2, \ldots, k \tag{1}$$

where w_j denotes the jth cluster and $D_L(x, m_j)$ represents distance between pixel x and m_j, mean of the jth cluster.

Step 3:
Discard clusters whose pixel members are less then Θ_N, i,e., if for any j, $N_j < \Theta_N$, then discard ω_j and set $k \leftarrow k - 1$.

Step 4:
Update each cluster center as

$$m_j = \frac{1}{N_j} \sum_{x \epsilon \omega_j} x, j = 1, 2, ..., k. \tag{2}$$

Step 5:
Compute average distance D_j of samples in cluster ω_j from their corresponding cluster center

$$D_j = \frac{1}{N_j} \sum_{x \epsilon \omega_j} D_L(x, m_j), j = 1, 2, ..., k \tag{3}$$

This step helps us in splitting a cluster when the cluster contains less number of pixels.

Step 6:
Compute the overall average distance of the samples from their respective cluster centers as

$$D = \frac{1}{N} \sum_{j=1}^{k} N_j D_j \tag{4}$$

which is also used for splitting operation.

Step 7:
If $k \leq \frac{K}{2}$,which is the case when the number of clusters are far less than the required number of clusters K,then there is a need to split the clusters with more number of samples (step 8 to step 10). If $k > 2K$, which is the case when the number of clusters are far greater than the required clusters K, then we merge the clusters which are closer to each other, (step 11 to step 13). Otherwise go to step 14.

Step 8:
Find the standard deviation vector $\sigma_j = [\sigma_1^j, ..., \sigma_n^j]^T$ for all clusters with each element given by,

$$\sigma_i^j = \sqrt{\frac{1}{N_j} \sum_{x \epsilon \omega_j} (x_i - m_i^j)^2} \tag{5}$$

where ($i = 1, .., n$, $j = 1, .., k$), m_i^j is the ith component of m_j and σ_i is the standard deviation of the samples in ω_j along the ith coordinate axis, and N_j is the number of samples in ω_j.

Step 9:
Find maximum component of each σ_j as σ^j_{max} for $j = 1, ...k$.

Step 10:
For any σ^j_{max}, $j = 1, ...k$, if all of the following are true,
$\sigma^j_{max} > \Theta_S$,
$D_j > D$,
$N_j > 2\Theta_N$
then split the cluster m_j into two new cluster centers m^+_j and m^-_j by adding $\pm\delta$ to the component of m_j corresponding to σ^j_{max}, where δ can be $\alpha\sigma^j_{max}$, for some $\alpha > 0$. Then delete m_j and let $k \leftarrow k + 1$ and go to step2.
Else go to step 14.

Step 11:
Find pairwise distances between every two cluster centers as D_{ij},

$$D_{ij} = D_L(m_i, m_j), \tag{6}$$

for all $i \neq j$. Arrange these $\frac{k(k-1)}{2}$ distances in ascending order.

Step 12:
Find no more than P smallest D_{ij}'s those are also smaller than Θ_c and keep them in ascending order.
$D_{i_1j_1} \leq D_{i_2j_2} \leq ... \leq D_{i_Pj_P}$
These are the P small clusters to be merged.

Step 13:
Perform pairwise merge for $l = 1, ..., P$. If neither of m_{i_l} nor m_{j_l} has been used, then merge the above centers to form a new cluster with mean given by,

$$m = \frac{1}{N_{i_l} + N_{j_l}}[N_{i_l}m_{i_l} + N_{j_l}m_{j_l}] \tag{7}$$

Delete m_{i_l} and m_{j_l}, set $k \leftarrow k - 1$ and got to step 2.

Step 14:
Terminate if maximum number of iterations I is reached. Otherwise, go to step 2.

The above algorithm was successfully applied to satellite images to cluster images as different land cover types.

2.2. Morphological Operations

When we classify images using the above algorithm, we notice that the resulting clusters are not uniform. Instead, they have holes inside clusters due to the fact that pixels inside one area can represent different types of land cover. For example, pixels representing bare soil (especially humid one) or trees can have values close to those of pixels representing tarmac road. These two types of pixels can be classified into the same cluster. The same problem may appear in case of mixed pixels (where one pixel value may represent more than one type of object). On the other hand, one object (e.g. vegetation cover) may be covered by other objects like vehicles, houses nearby and their shadows. In this case it is not possible to classify all pixels "geographically" representing tree as tree and grass as grass. Therefore, "holes or gaps" in clustered image may appear. This type of problems are well known and were presented in literature [4]. For those cases, we need to uncover the real structure of images by applying morphological operations on the clustered images.

Closing and opening [5] are well known operations in image morphology to remove/fill the gaps in an image. In our experiments, closing operation is performed with size of the structuring element equal to the maximum size of the hole or gap in one cluster. There are other regions in the image where there may exist small gaps between vegetation regions. To fill these small gaps, opening operation is performed on the image obtained in the previous stage with the size of structuring element equal to the maximum size of the gap or hole. By using these operations the clustered image also gets smoothened. The sequence of operations and size of structuring elements may vary depending upon the types of the clustered images.

2.3. Feature Selection and Classification

A cluster in a clustered image is a group of pixels having similar characteristics. However, vegetation type of a cluster is unknown, i.e., we do not know if a cluster is grass or sand. In this paper, we use supervised learning to recognize the identity of the clusters. We classify the clusters into two types: vegetation and others. We compute six features for each of the clusters including mean and standard deviation of Y, Cb and Cr channels. Therefore, we have six features for each cluster.

Those six features are derived heuristically and they are not equally important. Irrelevant features can lead to several problems in modeling: 1) Training an unnecessarily large network requires more computational resources and memory, 2) high dimensional data may have the curse of dimensionality problem if the available data is limited, and 3) training algorithms for large networks can also have convergence difficulties and poor generalization. We utilize a feature selection algorithm developed in our previous work [6] to select a compact set of features that leads to an accurate model based on the available data.

The feature selection algorithm 1) determines an appropriate piecewise linear network (PLN) model for the given data set, 2) applies the orthonormal least squares procedure to the PLN model and searches for useful feature subsets using a floating search algorithm. The floating search prevents the nesting effect. The feature selection algorithm is computationally efficient because only one data pass is required.

The feature selection algorithm is explained in detail as follows.

Step 1 Design a piecewise linear classifier for given data: Piecewise linear classifier approximates the general Bayes discriminant. The available data is divided into a set of clusters where a local linear model is obtained for each cluster, by solving a set of linear equations. Neural classifiers including the PLC are usually designed by minimizing the standard training error,

$$E = \sum_{i=1}^{N_c} E(i) = \frac{1}{N_v} \sum_{i=1}^{N_c} \sum_{p=1}^{N_v} [t_p(i) - y_p(i)]^2 \qquad (8)$$

where N_c is the number of classes and $E(i)$, the mean-squared error for the ith output. Here $t_p(i)$ denotes the ith desired output for the pth input pattern, $y_p(i)$ denotes the ith observed output for the pth input pattern, and N_v denotes the total number of data patterns. In the PLC, $y_p(i)$ is the output from the piecewise linear network,

$$y_p(i) = \sum_{j=1}^{N+1} w^{(q)}(i,j) x_p^{(q)}(j) \qquad (9)$$

where N is the number of features, $w^{(q)}(i,j)$ denotes the model weight to the ith output from the jth feature in the qth cluster, $x_p^{(q)}(j)$ is the jth feature in the qth cluster, and $x_p^{(q)}(N+1)$ is the bias term which equals one. The available data is divided into a set of clusters where a local linear model is obtained for each cluster, by solving a set of linear equations. We assume that $t_p(i_c) = b$ and $t_p(i_d) = -b$ where i_c denotes the correct class number and i_d any incorrect class number for the current data pattern. If $i_c = \arg\max_i y_p(i)$, we say the PLC classifies the current pattern correctly. Otherwise, a classification error is counted.

Step 2 Search a list of good feature combinations using the floating search algorithm
We design a floating search method through Piecewise Linear Orthonormal Least Square (PLOLS) procedure in this section. The PLOLS procedure utilizes the modified Schmidt procedure to make each features in each cluster orthonormal. This procedure passes through the data set once, and all information needed for searching good combination of features is stored in the auto-correlation and cross-correlation matrices. Therefore, our feature selection algorithm is very efficient. Based on equations (8) and (9), the modified desired output may be represented in a matrix form as

$$\mathbf{t}' = \mathbf{x}^{(q)} \mathbf{w}^{(q)} + \Xi^{(q)} \qquad (10)$$

where each row in matrix $\mathbf{x}^{(q)}$ represent one feature vector that was assigned to the qth cluster, $\mathbf{w}^{(q)}$ denotes weight matrix in the qth cluster, and $\Xi^{(q)}$ are residual errors in the qth cluster. We apply the modified Schmidt procedure to each cluster, yielding the piecewise linear orthogonal (PLO) system

$$\mathbf{t}' = \mathbf{\Theta}^{(q)} \mathbf{A}^{(q)} \mathbf{w}^{(q)} + \Xi^{(q)} = \mathbf{\Theta}^{(q)} \mathbf{w}_o^{(q)} + \Xi^{(q)}. \qquad (11)$$

We need the following four definitions to describe our proposed feature selection algorithm. Let $\mathbf{X}(d) = \{x(i) : 1 \le i \le d, x(i) \in \mathbf{Z}\}$ be a subset of d features from the set $\mathbf{Z} = \{z(i) : 1 \le i \le N\}$ of N available features. Suppose we partitioned the feature space into N_c clusters and obtained its PLO system as (11),
Definition 1:
The individual fitness of one feature, $x(i)$, is

$$\mathbf{S}_0(\mathbf{x}(i)) = \sum_{k=1}^{M} \sum_{q=1}^{N_c} (w_o^{(q)}(k,i))^2, \qquad (12)$$

which is the total variance explained for all outputs due to the ith feature.
Definition 2:
The fitness of a set of $\mathbf{X}(d)$ is measured as,

$$J(\mathbf{X}(d)) = \sum_{i=1}^{d} \sum_{k=1}^{M} \sum_{q=1}^{N_c} (w_o^{(q)}(k,i))^2, \qquad (13)$$

which is the total variance explained for all outputs due to all features in the set $\mathbf{X}(d)$.
Definition 3:
The fitness $S_{d-1}(x(i))$ of the feature $x(i)$, $1 \le i \le d$, in the set $\mathbf{X}(d)$ is defined by

$$\mathbf{S}_{d-1}(\mathbf{x}(i)) = \sum_{k=1}^{M} \sum_{q=1}^{N_c} (w_o^{(q)}(k,i))^2, \qquad (14)$$

where $x(i)$ is the last feature in the set $\mathbf{X}(d)$ that is made orthonormal to the other bases in the modified Schmidt procedure.
Definition 4:
The fitness $S_{d+1}(x(i))$ of the feature $x(i)$ with respect of $\mathbf{X}(d)$, where $x(i) \in \mathbf{Z} - \mathbf{X}(d)$, is

$$\mathbf{S}_{d+1}(\mathbf{x}(i)) = \sum_{k=1}^{M} \sum_{q=1}^{N_c} (w_o^{(q)}(k,i))^2, \qquad (15)$$

where $x(i)$ is made orthonormal to $\mathbf{X}(d)$, to get $w_o(k,i), k = 1, 2, \cdots, M$. These four definitions are fitness measures for one feature or feature combinations that will guide the feature selection process.

Algorithm Description
We are now ready to describe the proposed feature selection algorithm for selecting N_s features from N available features.

1. Using the error and trial method to determine an appropriate number of clusters, N_c, which will be used in the PLC.
2. Design an N_c cluster PLC for the data by solving a set of linear equations for each cluster.
3. Change desired output using the OR algorithm.
4. Search a list of good feature combinations using the floating search algorithm, based on the above four definitions.

Advantages of the proposed algorithm are as follows: 1) It selects features rather than a combination of all the features such as those selected by transformation based methods (PCA, Wavelet), 2) It considers interactions among features and measures the correlations via the amount of explained variance by features, 3) It is computationally efficient, 4) It automatically handles the extremely unbalanced data sets where the number of instances in some classes are significantly more than those in other classes, and 5) The algorithm produces a list of best combinations that contain different numbers of features, users then have the flexibility to choose one based on performance.

After the compact set of features are selected, we use an MLP classifier to classify the cluster to one of the two classes [7][8]. The classifier utilized a new objective function that had more free parameters than the classical objective functions, and used an iterative minimization technique to solve multiple sets of numerically ill-conditioned linear equations. An enhanced feedforward network training algorithm was also used to reduce a separate error function with respect to hidden layer weights. The MLP classifier is explained in detail as follows.

We are given a set of N_v training patterns (x_p, t_p) where the pth input vector x_p and pth desired output vector t_p have dimension N and N_c respectively. A three layer, fully connected MLP networks with sigmoid activation function for the hidden layer is used. For the pth pattern, the jth hidden unit net and activation functions are

$$net_p(j) = \sum_{k=1}^{N+1} w(j,k) \cdot x_p(k) \tag{16}$$

$$O_p(j) = f(net_p(j)) = \frac{1}{1 + exp^{(-net_p(j))}} \tag{17}$$

the ith observed output is

$$y_p(i) = \sum_{k=1}^{N+1} w_{oi}(i,k) \cdot x_p(k) + \sum_{j=1}^{N_h} w_{oh}(i,j) \cdot O_p(j) \tag{18}$$

where $w_{oi}(i,k)$ and $w_{oh}(i,j)$ are weights connecting to the ith output unit from the kth input and jth hidden unit respectively. For the jth hidden unit and pth pattern, the desired net function $net_{pd}(j)$ is constructed as

$$net_{pd}(j) \cong net_p(j) + Z \cdot \delta_p(j) \tag{19}$$

Z is the learning rate and $\delta_p(j)$ is the delta function of the jth hidden unit and is defined as

$$\delta_p(j) = f'(net_p(j)) \sum_{i=1}^{N_c} \delta_{po}(i) w_o(i,j) \tag{20}$$

where $\delta_{po}(i)$ is the delta function of the ith output layer,

$$\delta_{po}(i) = t_p(i) - y_p(i) \tag{21}$$

The hidden weights are updated as

$$w(j,k) \leftarrow w(j,k) + Z \cdot e(j,k) \tag{22}$$

where $e(j,k)$ is the hidden weight change. With the basic operations and (19-21), we can use the following equation to solve for the changes in the hidden weights,

$$net_{pd}(j) + Z \cdot \delta_p(j) \cong \sum_{k=1}^{N+1} [w(j,k) + Z \cdot e(j,k)] \cdot x_p(k) \tag{23}$$

we obtain

$$\delta_p(j) \cong \sum_{k=1}^{N+1} e(j,k) \cdot x_p(k) \tag{24}$$

Before solving (24) in the least squares sense, an objective function for the jth hidden unit is defined as

$$E_\delta(j) = \sum_{p=1}^{N_v} [\delta_p(j) - \sum_{k=1}^{N+1} e(j,k) x_p(k)]^2 f'(net_p(j)) \tag{25}$$

which is minimized with respect to $e(j,i)$ using the conjugate gradient method and we obtain the hidden weights change $e(j,k)$, we then update the hidden weights by performing (22).

Refer to our previous publications [6, 7, 8] for more details of the feature selection and the MLP classifier training algorithms.

2.4. Data Acquisition

The satellite imagery has been obtained from the Google Earth. The images we used in the paper encircle a range of vegetation such as trees and grass and also include other areas like road, soil, cars and buildings. To overcome the resolution problems, the images were captured at optimum height and the captured images were stored in standard image formats using snagit image capturing software. We collected 120 images and present several examples at the end of this paper.

3. RESULTS

To study the effectiveness of the ISODATA algorithm, we have applied it to the 120 satellite images[1]. As mentioned earlier, morphological operations are needed to fill the holes or gaps formed due to mixed pixels. Fig. 1 and Fig. 2 show the results of our experiments where the first column includes original images, the second column contains clustered images produced by the ISODATA algorithm, and the third column consists of outputs from morphological operations. Note that both "grass" and "tree" clusters in the Figures are treated as "vegetation" type in the classification followed, though the ISODATA algorithm clustered them into different clusters.

We computed the six features for each of the clusters and assigned one class type (vegetation or non-vegetation) to each of the clusters by inspection. The feature selection algorithm selected three of the features (means of the Y and Cb channels and standard deviation of the Cb Channel) and the other three features were turned out to be not useful. To test if a classifier can automatically identify vegetation type, we used the 5-fold cross-validation method on the 120 images and achieved 96% classification accuracy based on the three selected features. The classifier involved in the 5-fold cross-validation was an MLP classifier with 3 inputs, 1 hidden unit and 2 outputs, and the classifier was trained by the method proposed in [7, 8].

4. DISCUSSION

By comparing the ISODATA clustered images in the second column of Fig. 1 and Fig. 2 with the original images in the first column, we conclude that ISODATA correctly clustered areas having similar characteristics in most cases. However, there are small holes and gaps in some of the clustered images. After morphological operations, it is obvious that all the gaps have been filled/removed and the boundaries are smoothened without affecting the natural shape of the clustered image (see the third column in Fig. 1 and Fig. 2).

The 5-fold cross-validation results showed that it is possible to automatically identify vegetation from other land covers. The classifier can discriminate the two types with an accuracy of up to 96% in the cross-validation. Note that this paper focus on the first part of a flight simulation project. The second part of the project, vegetation adaptation, will automatically adapt the identified vegetation areas to different seasons. We reported the second part in [9].

5. CONCLUSION

In this paper we presented a method to automatically identify vegetation based upon satellite imagery. The identified vegetation area can be then adapted to different seasons by texture synthesis techniques [9]. We first used the ISODATA algorithm to cluster pixels in an image to similar clusters; we then applied morphological operations to remove holes or gaps inside the clusters. We then used the feature selection algorithm to select the most relevant features which can discriminate the vegetation regions from other regions. Finally, we utilized a classifier to classify the resulting clusters into one of the two land cover types: vegetation and "others" from the selected features. Experiments showed that the proposed algorithm can effectively cluster pixels in an image to the two types of land covers and the classifier can identify them with an accuracy of 96%.

[1]We utilized a fast implementation of the ISODATA algorithm by professor David Mount from the University of Maryland. See online code: http://www.cs.umd.edu/ mount/Projects/ISODATA/

6. REFERENCES

[1] James B. Campbell, *Introduction to Remote Sensing*, The Guilford Press, Fourth edition, 2007.

[2] J. P. Wayman, R. H. Wynne, J. A. Scrivani, and G. A. Burns, "Landsat tm-based forest area estimation using iterative guided spectral class rejection," *Photogrammetric Engineering and Remote Sensing*, vol. 67, pp. 1155–1166, 2001.

[3] G. Sharma and H. J. Trusell, "Digital color imaging," *IEEE Transactions on Image Processing*, vol. 6, pp. 901–932, 1997.

[4] Stephen S. Wilson, "Theory of matrix morphology," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 6, 1992.

[5] Su Chen and Robert M. Haralick, "Recursive erosion, dilation, opening, and closing transforms," *IEEE Transactions on Image Processing*, vol. 4, no. 3, 1995.

[6] Jiang Li, Michael T. Manry, Pramod L. Narasimha, and Changhua Yu, "Feature selection using a piecewise linear network," *IEEE Transaction on Neural Network*, vol. 17, no. 5, pp. 1101–1105, 2006.

[7] Jiang Li, Michael T. Manry, Li-Min Liu, Changhua Yu, and John Wei, "Iterative improvement of neural classifiers," *Proceedings of the Seventeenth International Conference of the Florida AI Research Society*, May 2004.

[8] R. G. Gore, Jiang Li, M. T. Manry, L. M. Liu, and Changhua Yu, "Iterative design of neural network classifiers through regression," *Special Issue of International Journal on Artificial Intelligence Tools*, vol. 14, no. 1-2, pp. 281–302, 2005.

[9] Srinivas Jakkula, Vamsi K. R. Mantena, Ramu Pedada, Yuzhong Shen and Jiang Li, "Seasonal adaptation of vegetation color in satellite images," *IPCV Conference*, 2008, submitted.

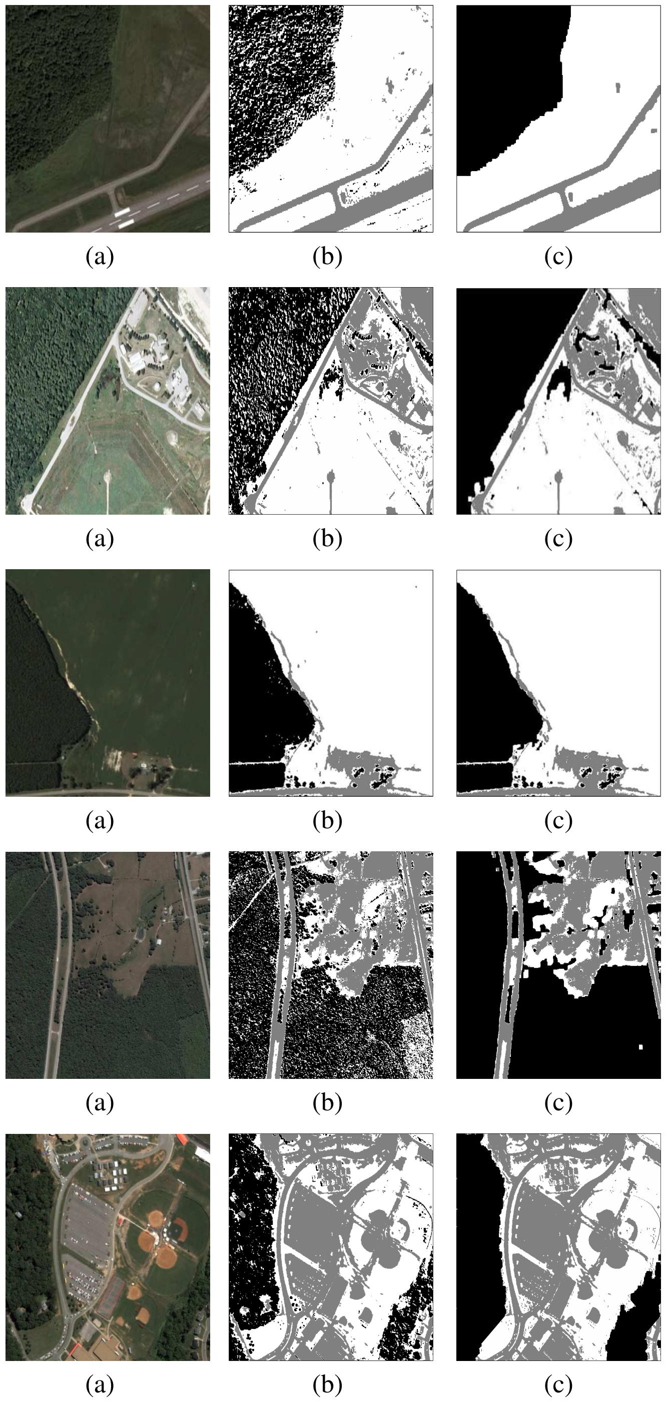

Fig. 1. (a). Input Image. (b) Result of ISODATA algorithm. (c) Result of morphological operations.

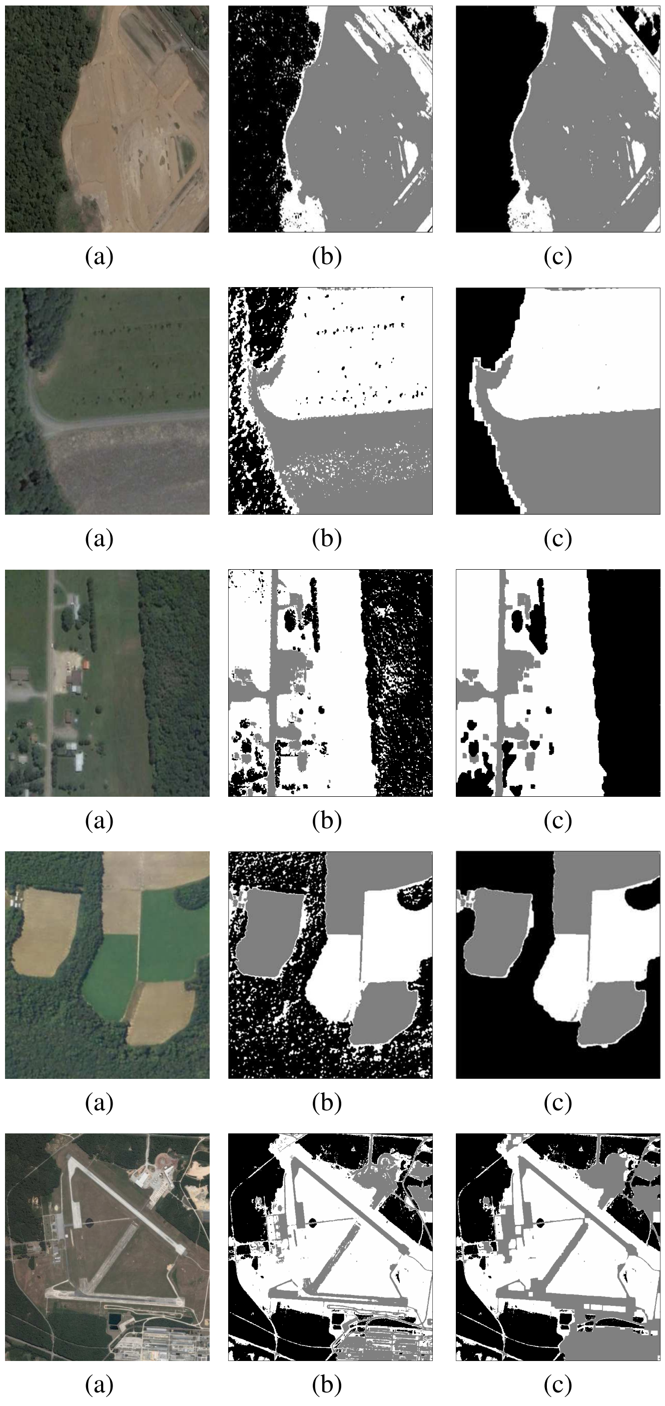

Fig. 2. (a). Input Image. (b) Result of ISODATA algorithm. (c) Result of morphological operations.