Focal-Test-Based Spatial Decision Tree Learning: A Summary of Results

***Abstract*—Given a raster spatial framework, as well as training and test sets, the spatial decision tree learning (SDTL) problem aims to minimize classification errors as well as salt-and-pepper noise. The SDTL problem is important due to many societal applications such as land cover classification in remote sensing. However, the SDTL problem is challenging due to the spatial autocorrelation of class labels, and the potentially exponential number of candidate trees. Related work is limited due to the use of local-test-based decision nodes, which can not adequately model spatial autocorrelation during test phase, leading to high salt-and-pepper noise. In contrast, we propose a focal-test-based spatial decision tree (FTSDT) model, where the tree traversal direction for a location is based on not only local but also focal (i.e., neighborhood) properties of the location. Experimental results on real world remote sensing datasets show that the proposed approach reduces salt-and-pepper noise and improves classification accuracy.**

I. Introduction

Given a raster spatial framework, as well as training and test sets, the spatial decision tree learning (SDTL) problem aims to find a decision tree model that minimizes classification errors as well as salt-and-pepper noise. Figure 1 is an illustrative example. Some input features of aerial photo bands are in Figure 1(a) as well as Figure 1(b) and ground truth class labels (red for dryland, green for wetland) are in Figure 1(c). A decision tree model is learned from the dataset by C4.5 algorithm [1], and its final classification result is shown in Figure 1(d). As can be seen, the generated decision tree has very poor classification performance with lots of salt-and-pepper noise.

Societal Applications: The SDTL problem has many societal applications. In the field of remote sensing, a large amount of images of the earth surface are collected (e.g., NASA collects about 5TB data per day). SDTL can be used to classify remote sensing images into different land cover types [2], which is important in urbanization study [3], natural resource management [4][5], and disaster management [6], etc. In medical image processing, SDTL can help in lesion classification and brain tissue segmentation [7][8] on MRI images. It can also be used for galaxy classification [9] in astronomy and semi-conductor inspection [10] in material science.

Challenges: There are several challenges in the SDTL problem. First, learning samples show spatial autocorrelation in class labels and spatial heterogeneity in feature values. Thus, testing only local feature information in decision nodes results in salt-and-pepper noise (locations or pixels whose class labels are different from those for their neighbors). For example, the class labels in Figure 1(c) show strong spatial autocorrelation due to the phenomenon of "patches" (i.e., contiguous regions of the same class). The final prediction of the local-test-based decision tree generated by C4.5 algorithm exhibits salt-and-pepper noise, as shown in Figure 1(d). Second, the number of candidate trees is exponential to the number of features. Constructing an optimal decision tree (e.g., minimizing number of nodes, or maximizing average information gain, etc.) is known to be NP-hard [11].

(a) Aerial photo features in 2005 (b) Aerial photo features in 2008

(c) ground truth class labels (d) DT classification with salt-and-pepper noise

Fig. 1. Real world problem example (best viewed in color)

Related work and limitations: Figure 2 presents a classification of related work. Existing work includes non-spatial decision trees (e.g., ID3 [12], C4.5 [1] and CART [13]) and spatial entropy or information gain based decision trees [14][15][16][17]. Non-spatial decision trees make the i.i.d. assumption of learning samples and ignore the spatial autocorrelation effect. Spatial entropy or information gain based decision trees use both spatial autocorrelation level and information gain to select candidate tree node tests. However, both of these use local-test-based decision nodes (i.e., decision nodes test each learning sample independently), and thus can not adequately model spatial autocorrelation in the test phase, leading to salt-and-pepper noise. In contrast to existing work,

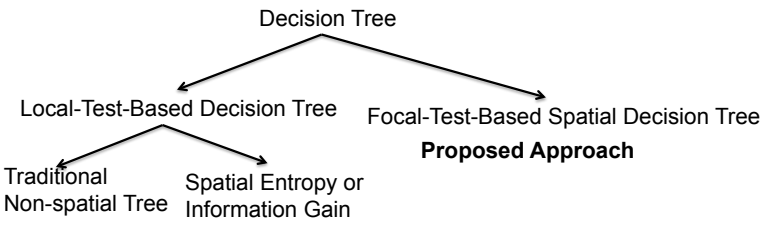
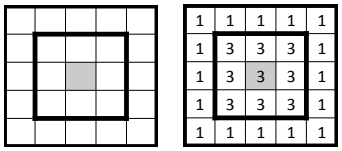


Fig. 2. Related work classification



(a) neighborhoods (b) attribute values

Fig. 3. Neighborhood, autocorrelation, and size

we propose a focal-test-based spatial decision model, whereby the tree traversal direction of a learning sample is based on not only local but also focal (neighborhood) properties of features.

Contributions: There are three main contributions of the paper: (1) we define a focal-test-based spatial decision tree (FTSDT) model; (2) we propose a learning algorithm for the model with computational performance tuning; (3) we evaluate the proposed approach using experiments on real world remote sensing datasets.

Scope: This paper focuses on incorporating focal tests inside a decision tree for raster data classification. Other classificatoin algorithms such as Markov Random Field [18], Spatial Autoregression (SAR) model[19], logistic regression, neural network, etc, are beyond the scope. In addition, for simplicity, this paper only considers continuous features. The case of discrete features is not addressed.

Outline: The outline of the paper is as follows: Section 2 defines basic concepts and formalizes the SDTL problem; Section 3 introduces a naive FTSDT learning algorithm, and a faster incremental algorithm. The effectiveness and efficiency of the proposed approach are evaluated on a real world remote sensing dataset in Section 4. Section 5 discusses some relevant techniques. Section 6 concludes the paper with future work.

II. Basic Concepts and Problem Formulation

This section first introduces basic concepts. Then, it formally defines the spatial decision tree learning problem and illustrates it.

A. Basic Concepts

Indicator formula: An indicator formula represents if a test result is TRUE or not. It is defined as below, where f is the feature value of a location and δ is a threshold. For example, the indicator variables of Figure 4(a) are shown in Figure 4(b) for ($f \leq 1$).

$$I(f \leq \delta) = \begin{cases} 1 & \text{if } f \leq \delta; \\ -1 & \text{if } f > \delta. \end{cases}$$

Neighborhood relationship: A neighborhood relationship characterizes the range of dependency between spatial locations. It is commonly represented as a W-matrix, whose element $W_{i,j}$ has a non-zero value when locations i and j are **neighbors**, and has a zero value otherwise. One important characteristic of a neighborhood is its **size** (e.g., size s means a (2s+1) by (2s+1) window). For example, in the spatial raster framework of Figure 3(a), the centering pixel has two neighborhoods of different sizes: inner window of size 1 (3 by 3), outer window of size 2 (5 by 5).

Salt-and-peper noise: Salt-and-pepper noise is a single pixel (or a small group of contiguous pixels) distinct from its (or their) spatial context (e.g., neighborhood). For example, in Figure 5(f), the green pixels inside red neighborhoods are salt-and-pepper noise.

Spatial autocorrelation statistic: Given a neighborhood definition and a spatial attribute, a spatial autocorrelation statistic measures the dependency between attribute values of neighboring locations. A positive high value of the statistic indicates that nearby locations have very similar attribute values. One common spatial autocorrelation statistic is Gamma index [20] Γ, as defined in Equation (1), where $W^S_{i,j}$ is a w-matrix element with a neighborhood size s, and $S_{i,j}$ is similarity between attribute values (e.g., for Γ^S_I, it is the product of the indicator variables at locations i and j). Its **local** version is defined in Equation (2). (**Note:** Γ^S_I used in later examples refers to $\Gamma^S_I(i)$ omitting i.)

$$\Gamma = \frac{\sum_{i,j} W^S_{i,j} S_{i,j}}{\sum_{i,j} W^S_{i,j}};\ \Gamma^S_I = \frac{\sum_{i,j} W^S_{i,j} I(i) I(j)}{\sum_{i,j} W^S_{i,j}} \quad (1)$$

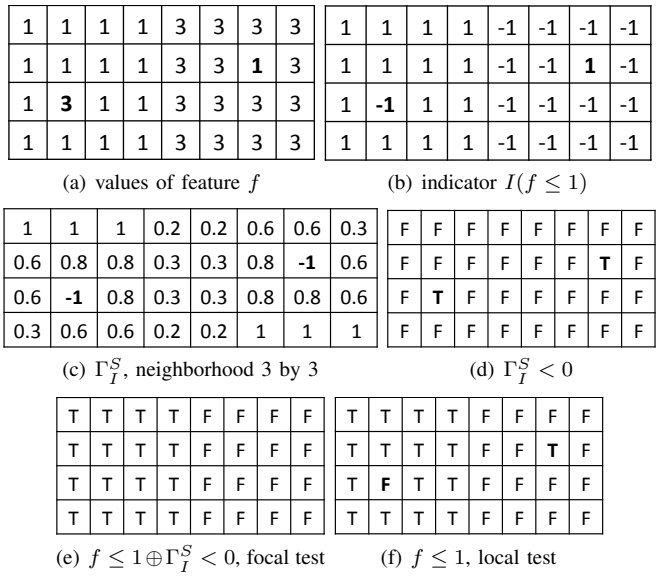
$$\Gamma(i) = \frac{\sum_{j} W^S_{i,j} S_{i,j}}{\sum_{j} W^S_{i,j}};\ \Gamma^S_I(i) = \frac{\sum_{j} W^S_{i,j} I(i) I(j)}{\sum_{j} W^S_{i,j}} \quad (2)$$

Due to spatial heterogeneity, a local autocorrelation statistic may show different trends from a global autocorrelation statistic, especially for salt-and-pepper noise pixels, whose local gamma values are negative. For example, given the gamma index definition above and a neighborhood size 3 by 3, the global autocorrelation of Figure 4(b) is 0.45, but local autocorrelation of the pixel at (3,2) is -1. Moreover, a local autocorrelation level is sensitive to neighborhood size choices. For example, the local gamma of the centering pixel in Figure 3 (b) is 1 in a 3 by 3 neighborhood, but -0.33 in a 5 by 5 neighborhood.

Local test based tree node: A local test based tree node tests the feature value of the location itself, either against a threshold (continuous features) or against a set of possible values (discrete features)[21][11], and then determines the sample's tree traversal direction. For example, tree node test results of Figure 4(a) are shown in Figure(f), where there are two salt-and-pepper noise pixels.

Focal function: A focal function is a function of a set of cells within the neighborhood of the current cell in a raster framework. For example, local gamma index Γ^S_I introduced previously is a focal function. Given feature f shown in Figure 4(a), its focal function Γ^S_I is shown in Figure 4(c).

Focal (function) test: A focal (function) test is a test of focal function value against a threshold. For example, $\Gamma^S_I < 0$ is a focal test and its test result TRUE means that the current



(a) values of feature f (b) indicator $I(f \leq 1)$

(c) Γ_I^S, neighborhood 3 by 3 (d) $\Gamma_I^S < 0$

(e) $f \leq 1 \oplus \Gamma_I^S < 0$, focal test (f) $f \leq 1$, local test

Fig. 4. Example of indicator variable, local gamma, local and focal tests

pixel may be salt-and-pepper noise. The focal test result of Figure 4(a) is shown in Figure 4(d).

Focal test based spatial tree node: A focal-test-based spatial tree node is a decision node which checks not only local but also focal properties of a location. For example, $(f \leq \delta) \oplus (\Gamma_I^S < 0)$ ($\oplus$ means "XOR") can define a focal-test-based decision node. The focal test part can help identify salt-and-pepper noise locations, whose final tree traversal directions are corrected via "XOR" operator. For example, the final focal test result of Figure 4(a), shown in Figure 4(e), has fewer salt-and-pepper noise pixels compared with local test result shown in Figure 4(f).

Focal test based spatial decision tree (FTSDT): An FTSDT is a tree model made up of focal-test-based spatial tree nodes. Figure 5(g) is an example, where the focal function is Γ_I^S with $s = 1$.

B. Problem Definition

Formally, the spatial decision tree learning problem is defined as follows:

Given:

- A spatial framework S
- A spatial neighborhood definition N, and its maximum size S_{max}
- Training and test samples (with features and class labels) drawn from S

Find:

- A decision tree model based on training samples.

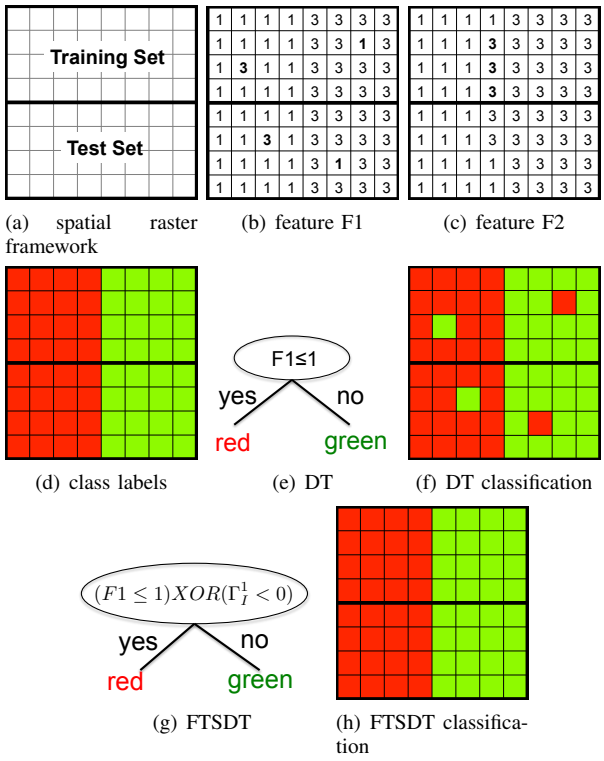
Objective:

- Minimize classification errors as well as salt-and-pepper noise

Constraints:

- Training samples form contiguous patches of locations in S
- Spatial autocorrelation exists in class labels
- Spatial framework S is a 2-D regular grid

Example: Consider a raster spatial framework S (8 by 8) shown in Figure 5(a), which consists of training pixels on the upper half and test pixels on the lower half. Neighborhood N is defined as a square window, whose maximum size is 1 (3 by 3). The minimum tree node size is 4. Figure 5(b)(c) show two candidate features F_1 and F_2. Figure 5(d) shows class labels. The output local-test-based traditional decision tree learned from the training set is shown in Figure 5(e), whose classification has salt-and-pepper noise, as shown in Figure 5(f). In contrast, the output focal-test-based spatial decision tree is shown in Figure 5(g), and its classification is shown in Figure 5(h), which has no salt-and-pepper noise.



(a) spatial raster framework (b) feature F1 (c) feature F2

(d) class labels (e) DT (f) DT classification

(g) FTSDT (h) FTSDT classification

Fig. 5. Illustrative problem example (best viewed in color)

III. FTSDT Learning and Prediction Algorithms

This section introduces a naive algorithm of focal-test-based spatial decision tree learning with an illustrative execution trace. It then introduces a faster incremental algorithm to avoid redundant focal function computation in the training phase.

A. A Naive Algorithm

The Naive Algorithm has two phases: a training phase, named FTSDT-Train-Naive; and a prediction phase, named FTSDT-Predict.

FTSDT-Train-Naive: The FTSDT-Train-Naive algorithm (Algorithm 1) learns an FTSDT classifier from training samples. It is a divide and conquer method with a greedy strategy (i.e., maximizing information gain).

Steps 1 to 2 check the stopping criteria. If the number of training samples is smaller than a minimum tree node size, or

the class labels are identical, a leaf labeled with the majority class will be returned.

Steps 3 to 14 select the best candidate feature, threshold, and neighborhood size through exhaustive enumeration. More specifically, for each candidate neighborhood size, feature, and threshold (generated by picking up unique feature values after sorting), training samples are split based on focal test results (Node-Split-Naive subroutine) and the information gain is evaluated.

Steps 15 to 17 find the neighborhood size, feature, and threshold with the maximum information gain and create an internal node. The training samples are then split into two subsets based on the focal test results.

Steps 18 to 20 recursively call the training algorithm on each subset to construct a subtree. Finally, the internal node is returned.

Algorithm 1 FTSDT-Train-Naive(T, C, S_{max}, N_0)

Input:
- T: training samples where $T[i][f]$ is f^{th} feature value in i^{th} sample
- C: class labels where $C[i]$ is class label of i^{th} sample
- S_{max}: maximum neighborhood size
- N_0: minimum decision tree node size

Output:
- root of an FTSDT model

1: **if** $|T| < N_0$ or C unique class **then**
2: L=CreateLeaf(class(C)); **Return** L;
3: $IG_0 = -\infty$
4: **for each** candidate neighborhood size $s \in \{0...S_{max}\}$ **do**
5: **for each** candidate feature $f \in \{1...F\}$ **do**
6: Sort feature f values $T[i][f]$ ($i \in \{1...N\}$) in ascending order
7: **for each** $i \in \{1...(N-1)\}$ **do**
8: **if** $T[i][f] < T[i+1][f]$ **then**
9: $\delta = (T[i][f] + T[i+1][f])/2$
10: $\{T_1, T_2\}$=**Node-Split-Naive**(T, f, δ, s);
11: Split C into $\{C_1, C_2\}$ according to $\{T_1, T_2\}$
12: IG=**InformationGain**(C,C_1,C_2)
13: **if** $IG > IG_0$ **then**
14: $IG_0 = IG$; $s_0 = s$; $f_0 = f$; $\delta_0 = \delta$;
15: I=CreateInternalNode(f_0, δ_0, s_0);
16: $\{T_1, T_2\}$=**Node-Split-Naive**(T, f_0, δ_0, s_0);
17: Split C into C_1 and C_2 based on $\{T_1, T_2\}$
18: I.LeftChild=**FTSDT-Train-Naive**(T_1, C_1, S_{max}, N_0)
19: I.RightChild=**FTSDT-Train-Naive**(T_2, C_2, S_{max}, N_0)
20: **Return** I

Node-Split-Naive: The Node-Split subroutine (Algorithm 2) splits the training samples into two subsets based on their focal test results in the decision node.

Step 1 initializes the two subsets as empty sets. One subset is to contain samples with node test results TRUE and the other is to contain samples with test results FALSE.

Steps 2 to 10 compute the focal tree node test result of each training sample and add the sample to its proper subset accordingly. For each training sample, it computes the local test result (shown as $LocalVal[i]$), the focal function value (shown as $FocalVal[i]$), and the focal test result (shown as $FocalRes[i]$). The final (focal) test result is a logical operation (shown as $\bigotimes$) of the local test result and the focal function test result. For example, we may specify the focal function as Γ_I^S (introduced earlier in 2.1), focal function test as $\Gamma_I^S < 0$, and the logical operator as $\oplus$ (i.e., "XOR").

Algorithm 2 Node-Split-Naive(T, f, δ, S)

Input:
- T: training samples where $T[i][f]$ is f^{th} feature value in i^{th} sample
- f: a feature index
- δ: threshold of feature test
- S: neighborhood size

Output:
- $\{T_1, T_2\}$: sample subsets with test result TRUE and FALSE respectively

1: $T_1 = T_2 = \emptyset$
2: **for each** $i \in \{1...N\}$ **do**
3: $LocalRes[i] = (T[i][f] \leq \delta)$
4: $FocalVal[i]$=**FocalFunction**(T,i,f,δ,S);
5: $FocalRes[i]$=**FocalTest**(FocalVal[i]);
6: $FinalRes[i] = LocalRes[i] \bigotimes FocalRes[i]$
7: **if** $FinalRes[i]$==TRUE **then**
8: $T_1 = T_1 \cup \{T[i]\}$
9: **else**
10: $T_2 = T_2 \cup \{T[i]\}$
11: **return** $\{T_1, T_2\}$

An Execution Trace of FTSDT-Train-Naive: Consider the problem example in Figure 5, where the training set has two features F_1 and F_2, the maximum neighborhood size is 1 (3 by 3 neighborhood), and the minimum tree node size is 4. Assume the focal function is Γ_I^S (local gamma index of indicator variable, introduced earlier), the focal test function is $\Gamma_I^S < 0$ and the final tree node test is $f \leq \delta \oplus \Gamma_I^S < 0$. We now execute the FTSDT-Train-Naive algorithm (**Algorithm 1**) on the example as follows.

Steps 1 to 2 check the stop criteria. Since there are 8 samples (above the minimum node size 4) from 2 different classes, the stop criterion is not met.

Steps 4 to 9 enumerate parameter space $< s, f, \delta >$. Since F_1 and F_2 both have only two distinct values, the only candidate threshold is $\delta = 1$. Thus, there are four candidates, with $s = 0, 1$ and $f = F1, F2$ respectively. They are represented in Figure 6, Figure 7, Figure 8, and Figure 9 respectively.

Step 10 calls the subroutine **Node-Split-Naive**. It computes the local test result ($f \leq \delta$), focal function (Γ_I^S), focal test result ($\Gamma_I^S < 0$), as well as the the final test result ($f \leq \delta \oplus \Gamma_I^S < 0$) for each training sample. For example, in Figure 8 for candidate 3, values of a feature F_1 are shown in Figure 8(a); indicator variables are shown in Figure 8(b); focal function values Γ_I^1 are shown in Figure 8(c), where two salt-and-pepper noise pixels have negative values; the focal test result is shown in Figure 8(d); final test result is shown in Figure 8(e), where the red class has final test result as TRUE, and the green class as FALSE. For candidate 1 (shown in Figure 6), $s = 0$ (i.e., local), $\Gamma_I^S = 0$ and focal test results are always FALSE. Thus, the final focal test result ("XOR" operation) is the same as the traditional local test.

1	1	1	1	3	3	3	3
1	1	1	1	3	3	**1**	3
1	**3**	1	1	3	3	3	3
1	1	1	1	3	3	3	3

(a) training: F_1

T	T	T	T	F	F	F	F
T	T	T	T	F	F	**T**	F
T	**F**	T	T	F	F	F	F
T	T	T	T	F	F	F	F

(b) local test $F_1 \leq 1$

Fig. 6. Candidate 1: F_1, $\delta = 1$, and $s = 0$ (local test)

1	1	1	1	3	3	3	3
1	1	1	**3**	3	3	3	3
1	1	1	**3**	3	3	3	3
1	1	1	**3**	3	3	3	3

(a) training: F_2

T	T	T	T	F	F	F	F
T	T	T	**F**	F	F	F	F
T	T	T	**F**	F	F	F	F
T	T	T	**F**	F	F	F	F

(b) local test $F_2 \leq 1$

Fig. 7. Candidate 2: F_2, $\delta = 1$ and $s = 0$ (local test)

1	1	1	1	3	3	3	3
1	1	1	1	3	3	**1**	3
1	**3**	1	1	3	3	3	3
1	1	1	1	3	3	3	3

(a) training F_1

1	1	1	1	-1	-1	-1	-1
1	1	1	1	-1	-1	**1**	-1
1	**-1**	1	1	-1	-1	-1	-1
1	1	1	1	-1	-1	-1	-1

(b) indicator $I(F_1 \leq 1)$

1	1	1	0.2	0.2	0.6	0.6	0.3
0.6	0.8	0.8	0.3	0.3	0.8	**-1**	0.6
0.6	**-1**	0.8	0.3	0.3	0.8	0.8	0.6
0.3	0.6	0.6	0.2	0.2	1	1	1

(c) focal function, Γ_I^S

F	F	F	F	F	F	F	F
F	F	F	F	F	F	**T**	F
F	**T**	F	F	F	F	F	F
F	F	F	F	F	F	F	F

(d) focal test Γ_I^S <0

T	T	T	T	F	F	F	F
T	T	T	T	F	F	F	F
T	T	T	T	F	F	F	F
T	T	T	T	F	F	F	F

(e) $F_1 \leq 1 \oplus \Gamma_I^S$ <0, focal test

Fig. 8. Candidate 3: F_1, $\delta = 1$ and $s = 1$ (focal test: 3 by 3 neighborhood)

1	1	1	1	3	3	3	3
1	1	1	**3**	3	3	3	3
1	1	1	**3**	3	3	3	3
1	1	1	**3**	3	3	3	3

(a) training F_2

1	1	1	1	-1	-1	-1	-1
1	1	1	**-1**	-1	-1	-1	-1
1	1	1	**-1**	-1	-1	-1	-1
1	1	1	**-1**	-1	-1	-1	-1

(b) indicator $I(F_2 \leq 1)$

1	1	1	0.2	0.6	1	1	1
1	1	0	0	0.8	1	1	1
1	1	0.3	0.3	1	1	1	1
1	1	0.2	0.2	1	1	1	1

(c) focal function, Γ_I^S

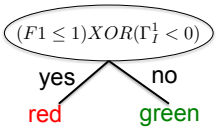
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F

(d) focal test Γ_I^S <0

T	T	T	T	F	F	F	F
T	T	T	**F**	F	F	F	F
T	T	T	**F**	F	F	F	F
T	T	T	**F**	F	F	F	F

(e) $F_2 \leq 1 \oplus \Gamma_I^S$ <0, focal test

Fig. 9. Candidate 4: F_2, $\delta = 1$, and $s = 1$ (focal test: 3 by 3 neighborhood)



(a) output FTSDT

1	1	1	1	3	3	3	3
1	**3**	1	1	3	3	3	3
1	1	1	1	3	**1**	3	3
1	1	1	1	3	3	3	3

(b) test set F_1

1	1	1	1	-1	-1	-1	-1
1	**-1**	1	1	-1	-1	-1	-1
1	1	1	1	-1	**1**	-1	-1
1	1	1	1	-1	-1	-1	-1

(c) $I(F_1 \leq 1)$

1	1	1	0.2	0.2	0.6	0.6	0.3
0.6	**-1**	0.8	0.3	0.3	0.8	0.8	0.6
0.6	0.8	0.8	0.3	0.3	**-1**	0.8	0.6
0.3	0.6	0.6	0.2	0.2	1	1	1

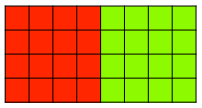
(d) focal function, Γ_I^S

F	F	F	F	F	F	F	F
F	**T**	F	F	F	F	F	F
F	F	F	F	F	**T**	F	F
F	F	F	F	F	F	F	F

(e) focal test, $\Gamma_I^S < 0$

T	T	T	T	F	F	F	F
T	T	T	T	F	F	F	F
T	T	T	T	F	F	F	F
T	T	T	T	F	F	F	F

(f) $F_1 \leq 1 \oplus \Gamma_I^S < 0$, focal test



(g) FTSDT prediction

Fig. 10. Output FTSDT and its prediction

Steps 11 to 14 split the training samples according to the final tree node test results and compute the information gain. In the example, the information gain values of four candidates are 0.663, 1, 0.626, and 0.626 respectively. Thus, feature F_1, threshold 1, and neighborhood size $s = 1$ (3 by 3) are the greedy choices.

Steps 15 to 20 create an internal node based on the selected parameters, as shown in Figure 10(a). Training pixels are split accordingly. Thus, four red pixels are in one subset, and four green pixels are in another subset. Since each subset has the same class, leaf nodes are returned.

FTSDT-Predict: The FTSDT-Predict algorithm (Algorithm 3) uses an FTSDT to predict class labels of test samples based on their feature values and a spatial neighborhood structure. It is a recursive algorithm. If the tree node is a leaf, then the class label of the leaf is assigned to all current samples. Otherwise, samples are split into two subsets according to the focal test results in the root node, and each subset is classified by its corresponding subtree.

Algorithm 3 FTSDT-Predict(R, T)

Input:
- R: Root of an FTSDT model
- T: training samples where $T[i][f]$ is f^{th} feature value in i^{th} sample

Output:
- C: class labels where $C[i]$ is class label of i^{th} sample

1: **if** $R.type == Leaf$ **then**
2: $C = R.class$; **Return** C
3: $f_0 = R.f$, $\delta_0 = R.\delta$, $s_0 = R.s$
4: $\{T_1, T_2\}$=**Node-Split-Naive** (T, f_0, δ_0, s_0, 0)
5: C_1=**FTSDT-Predict** ($R.Left$, T_1);
6: C_2=**FTSDT-Predict** ($R.Left$, T_2);
7: **return** C=Merge(C_1,C_2)

Execution Trace of FTSDT-Predict: Given the previously constructed FTSDT model (Figure 10(a)), and feature F_1 from test set (Figure 10(b)), the FTSDT-Predict algorithm first finds that the current node is a non-leaf. Then, the test samples are split into two subsets based on results of the focal test ($F_1 \leq 1 \oplus \Gamma_I^1 < 0$), as shown in Figure 10(c)(d)(e)(f). The

16 pixels on the left have final test results TRUE, and thus traverse the left branch. In the next recursion, they are labeled as the "red" class by a leaf node. Similarly, the 16 pixels on the right are labeled as the "green" class.

B. Computational Optimization: An Incremental Algorithm

The incremental algorithm (Algorithm 4 and Algorithm 5) modifies the parameter space enumeration process used in the naive algorithm. The key idea is to avoid redundant focal function computation. An important observation is that, under the same candidate neighborhood size and candidate feature, if the currently evaluated threshold is contiguous to the previous threshold (in sorted feature values), then only the focal function values of the current pixel and its neighbors need updating.

Algorithm 4 FTSDT-Train-Incremental(T,C, S_{max}, N_0)

Input:

- T: training samples where $T[i][f]$ is f^{th} feature value in i^{th} sample
- C: class labels where $C[i]$ is class label of i^{th} sample
- S_{max}: maximum neighborhood size, N_0: minimum decision tree node size

Output:

- root of an FTSDT model

1: **if** $|T| < N_0$ or C unique class **then**
2: L=CreateLeaf(class(C)); **Return** L;
3: **for each** candidate neighborhood size $s \in \{0...S_{max}\}$ **do**
4: **for each** candidate feature $f \in \{1...F\}$ **do**
5: Sort feature f values $T[i][f]$ ($i \in \{1...N\}$) in ascending order
6: $i_{last} = -1$; Initialize $LocalRes$, $FocalRes$;
7: **for each** $i \in \{1...(N-1)\}$ **do**
8: **if** $T[i][f] < T[i+1][f]$ **then**
9: $\delta = (T[i][f] + T[i+1][f])/2$
10: **if** $i_{last} < i - 1$ **then**
11: $\{\{T_1, T_2\}, FocalVal, LocalRes, FocalRes\}$=**Node-Split-Naive**($T$, f, δ, s)
12: Split C into $\{C_1, C_2\}$ according to $\{T_1, T_2\}$
13: **else**
14: **Node-Split-Incremental**(T, f, δ, i, s, $\{T_1, T_2\}$, $\{C_1, C_2\}$, $FocalVal$, $LocalRes$, $FocalRes$)
15: $i_{last} = i$; IG=**InformationGain**(C, C_1,C_2);
16: **if** $IG > IG_0$ **then**
17: $IG_0 = IG$; $s_0 = s$; $f_0 = f$; $\delta_0 = \delta$; //IG_0 initialized as $-\infty$
18: I=CreateInternalNode(f_0, δ_0, s_0); $\{\{T_1, T_2\}, ...\}$=**Node-Split-Naive**(T,f_0,δ_0,s_0);
19: Split C into $\{C_1, C_2\}$ based on $\{T_1, T_2\}$
20: I.LeftChild=**FTSDT-Train-Incremental**(T_1,C_1, S_{max}, N_0)
21: I.RightChild=**FTSDT-Train-Incremental**(T_2,C_2, S_{max}, N_0)
22: **Return** I

Based on this observation, the FTSDT-Train-Incremental algorithm (Algorithm 4) modifies the naive algorithm in two ways: First, it maintains focal function values as well as local and focal test results of all samples (variable returned by Node-Split-Naive now in step 11). Second, when the current threshold is immediately next to the previously evaluated candidate threshold ($i_{last} = i - 1$, namely, only one training sample updates its indicator variable $I(f \leq \delta)$ when the current threshold replaces the previous threshold), Node-Split-Incremental (Algorithm 5) is called instead of Node-Split-Naive, in order to make incremental updates to the focal function values as well as the local and focal test results (step 14).

Specific incremental update methods for Node-Split-Incremental function depend on the given focal function definition. For example, given a focal function test of $\Gamma_I^S < 0$ (defined in 2.1), $FocalVal[i] = \Gamma_I^S(i)$ will be updated to its negative (since only $I(i)$ flip the sign), $FocalRes[i]$ stays the same (both $I(i)$ and Γ_I^S flip signs) unless $FocalVal[i] = \Gamma_I^S(i) = 0$, in which case it is negated. Similarly, we can update $FocalVal[j]$ and $FocalRes[j]$, where $j \in N^S(i)$.

Algorithm 5 Node-Split-Incremental(T, f, δ, i, s, $\{T_1, T_2\}$,$\{C_1, C_2\}$, $FocalVal$,$LocalRes$,$FocalRes$)

Input:

- T: training samples where $T[i][j]$ is j^{th} feature value in i^{th} sample
- f: feature, δ: local test threshold
- i: current sample location, also largest index of samples below threshold
- S: neighborhood size, $\{T_1, T_2\}$,$\{C_1, C_2\}$: samples, class labels in two subsets
- $FocalVal$: focal function value of i^{th} sample
- $LocalRes$, $FocalRes$: local test and focal test result of i^{th} sample

Output:

- $\{T_1, T_2\}$,$\{C_1, C_2\}$, $FocalVal$, $LocalRes$, $FocalRes$ (all updated on original array)

1: Update $FocalVal[i]$, $FocalRes[i]$, and then $FinalRes[i]$
2: **for each** $j \in N^s(i)$, $N^s(i)$ is i's neighborhood of size s **do**
3: Update $FocalVal[j]$, $FocalRes[j]$, and then $FinalRes[j]$
4: Update $\{T_1, T_2\}$,$\{C_1, C_2\}$ according to updates on $FinalRes[i], FinalRes[j]$ ($j \in N^s(i)$)

IV. Evaluation

We evaluated the proposed focal-test-based spatial decision tree learning algorithm on a real world remote sensing dataset. The goal was to investigate the following questions:

- Does the focal-test-based spatial decision tree reduce salt-and-pepper noise compared with a local-test-based decision tree?
- Does the focal-test-based spatial decision tree improve classification accuracy compared with a local-test-based decision tree?
- Does the focal-test-based spatial decision tree have smaller tree size compared with a local-test-based decision tree?
- How does the computational performance of the incremental algorithm compare with that of the naive algorithm?

A. Experiment Setup

Experiment design: The experiment design is shown in Figure 11. For effectiveness evaluation, there are two candidate

learning algorithms: the FTSDT learner (focal-test-based spatial decision tree) and LTDT (local-test-based decision tree, we used traditional decision tree (DT) C4.5 as a representative) learner. Output LTDT and FTSDT models learned from the training set were used to classify the test set. Their classification accuracy, salt-and-pepper noise, and tree sizes were compared. For computational performance evaluation, we compared the naive approach and the incremental approach of FTSDT learner under different patch sizes, numbers of patches, and input maximum neighborhood sizes. All experiments were conducted in C language on a Dell workstation with Quad-core Intel Xeon CPU E5630 @ 2.53GHz, and 12 GB RAM.

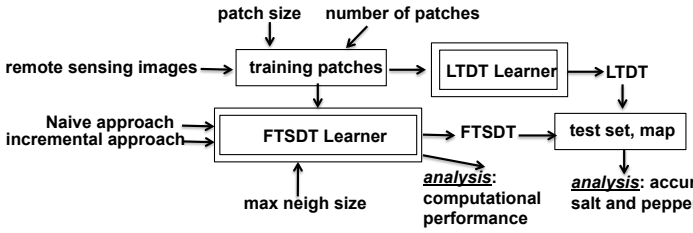


Fig. 11. Experiment Design

Dataset description: We used high resolution (3m*3m) remote sensing imagery collected from the city of Chanhassen, MN, by the National Agricultural Imagery Program and Markhurd Incorporation. There were 12 continuous explanatory features including multi-temporal spectral information (R,G,B,NIR bands), and Normalized Difference Vegetation Index (NDVI) for the years 2003, 2005, and 2008. Class labels (i.e., wet land, dry land) are created by a field crew and photo interpreters.

We further selected three different scenes from the city. On each scene, we use stratified clustered sampling. More specifically, we randomly selected a number of wetland and dryland circular contiguous clusters of pixels (whose radius is around 7 pixel) to create training set. The remaining pixels in the scenes were used as test sets. Contiguous clusters instead of isolated pixels were used in order to learn focal test rules in decision nodes. More details are given in Table I.

TABLE I. DESCRIPTION OF DATASETS

Scene	Size	Training samples
1	476*396	2319 (dryland class); 1883 (wetland class)
2	858*759	2963 (dryland class); 3140 (wetland class)
3	482*341	4804 (dryland class); 3829 (wetland class)

Choice of focal test functions: For the focal-test-based spatial decision tree, we used the specific focal test $(f \leq \delta) \oplus (\Gamma_I^S < 0)$ described earlier in Section II.

B. Does the focal-test-based spatial decision tree reduce salt-and-pepper noise?

Evaluation metric: We compared the amount of salt-and-pepper noise in prediction through a quantitative spatial autocorrelation measure, i.e., the gamma index (introduced earlier in Section II) with a queen neighborhood.

Parameter settings: The maximum neighborhood size was set to 11 pixels by 11 pixels. Minimum tree node size was 50 pixels.

Result: The last columns of Table II, Table III, and Table IV show the gamma index autocorrelation levels of DT and FTSDT predictions on three datasets. Gamma index value ranges between negative one and positive one. A larger gamma index means less salt-and-pepper noise. As can be seen, FTSDT improves the spatial autocorrelation by mostly over 10%. This confirms that FTSDT reduces salt-and-pepper noise compared to DT.

C. Does the focal-test-based spatial decision tree improve classification accuracy?

Parameter setting: The maximum neighborhood size was 11 by 11, minimum tree node size was 50 pixels.

Classification accuracy: We compared the classification performance of the proposed FTSDT classifier with DT in terms of confusion matrices, precision & recall, and overall accuracy on test set. The results are in Table II, Table III, and Table IV. In the confusion matrix, the columns are test samples classified as dryland and wetland respectively, and the two rows are test samples whose true class labels are dryland and wetland respectively. Precision and recall were computed on the wetland class. As can be seen, the FTSDT improves the overall classification accuracy of DT by around 3 to 4 percents. Improvements are also shown in the precision and recall. The reason of accuracy improvement is that the focal test in tree node helps correct salt-n-pepper noise errors during classification process.

D. Does the focal-test-based spatial decision tree have smaller tree size?

Parameter setting: The maximum neighborhood size was 11 by 11, minimum tree node size was 50 pixels.

Size of trees: We compared the number of nodes in learned FTSDT and DT models. A smaller tree is often desirable since it is more compact and easier to interpret. The results are shown in the "node number" columns of Table II, Table III, and Table IV. As can be seen, the FTSDT model has relatively a smaller number of nodes. This is due to the fact that the traditional decision tree has the i.i.d. assumption and can only discriminate different classes according to local feature values in each node. In contrast, focal-test-based decision tree can exploit the spatial context of a sample to help discriminate classes, especially for samples with potential salt-n-pepper noise errors. Thus, focal-test-based spatial decision tree requires fewer nodes to fit the training samples.

E. Parameter Sensitivity Analysis: maximum neighborhood size S_{max}

Parameter setting: The minimum tree node size was 50 pixels. Dataset scene 3 was used.

We observed the effectiveness (i.e. accuracy, spatial autocorrelation) of proposed approach under different values of parameter S_{max} (0 represents local test based decision tree). As can be seen in Figure 12(a), as S_{max} value increases, the autocorrelation levels of prediction first increase quickly and then increase slowly or remain relatively stable. This is due to the fact that salt-n-pepper error patterns are in relatively small spatial scales. When S_{max} is large enough to capture the salt-n-pepper noise patterns, increasing S_{max} does not improve

TABLE II. COMPARISON OF DT AND FTSDT ON SCENE 1

Models	Confusion Matrix		Precision	Recall	Overall Accuracy	Node Number	Autocorrelation Level
DT	99060	20172	0.711	0.762	0.807	33	0.838
	15456	49606					
FTSDT	109104	10128	0.822	0.717	0.845	31	0.957
	18406	46656					

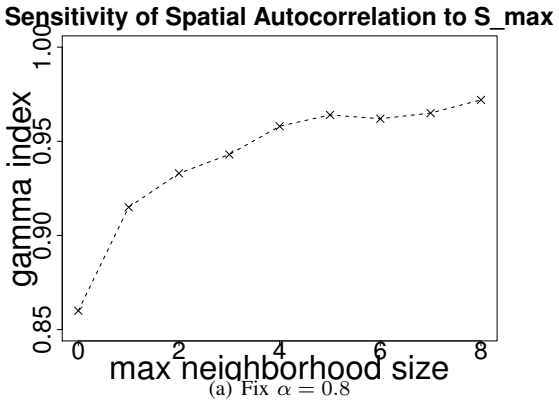
TABLE III. COMPARISON OF DT AND FTSDT ON SCENE 2

Models	Confusion Matrix		Precision	Recall	Overall Accuracy	Node Number	Autocorrelation Level
DT	383044	29749	0.844	0.694	0.844	21	0.917
	71134	161192					
FTSDT	384631	28162	0.868	0.799	0.884	21	0.976
	46673	185653					

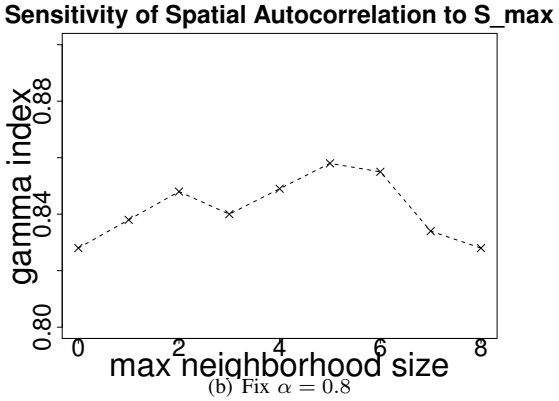
TABLE IV. COMPARISON OF DT AND FTSDT ON SCENE 3

Models	Confusion Matrix		Precision	Recall	Overall Accuracy	Node Number	Autocorrelation Level
DT	99723	18234	0.613	0.766	0.826	55	0.86
	8837	28935					
FTSDT	103913	14044	0.679	0.788	0.858	35	0.964
	8008	29764					

autocorrelation a lot. Figure 12 (b) shows the sensitivity of test accuracy on different S_{max}. As S_{max} increases, the test accuracy first increases and then decreases. A very large S_{max} may lead to overfitting. In practice, S_{max} can be set according to the resolution of image and the scale of autocorrelation on class labels.



(a) Fix $\alpha = 0.8$



(b) Fix $\alpha = 0.8$

Fig. 12. Single parameter sensitivity analysis

F. Evaluation of Computational Scalability

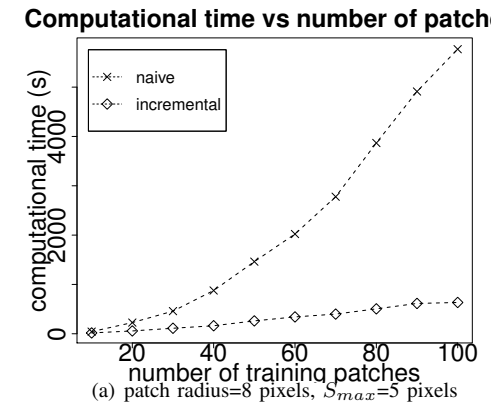
We compared the Naive FTSDT learning algorithm and the Incremental algorithm on different training patch sizes, different numbers of training patches, and different maximum neighborhood sizes S_{max}.

Different numbers of training patches: Figure 13(a) shows the effect of the number of input training patches. In this experiment, we fixed S_{max} to 5, and training patch size $radius$ to 8; and increased the number of training patches. As can be seen, the computational time of the incremental approach grows much slower than the naive approach as the number of training patches increases.

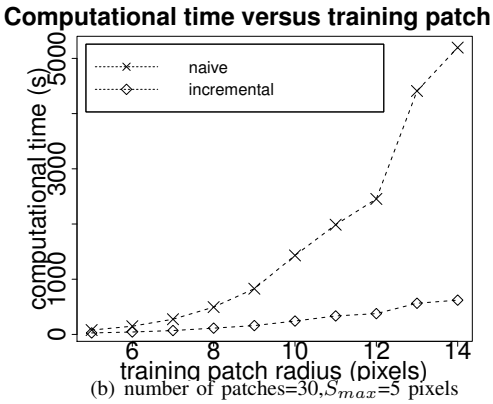
Different training patch sizes: Figure 13(b) shows the effect of input training patch size. In this experiment, we fixed S_{max} to 5, and the number of training patches to 30; and increased the training patch size. As can be seen, the computational time of the incremental approach grows more slowly than the naive approach as the training patch size increases.

Different max neighborhood sizes: Figure 13(c) shows the effect of parameter max size S_{max}. In this experiment, we fixed the number of training patches to 30 and patch size to 10; and increased S_{max}. As can be seen, the computational time of the incremental approach grows more slowly than the naive approach as S_{max} increases.

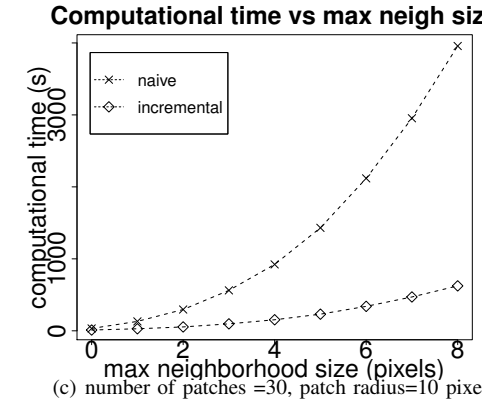
Distribution of cost on features: We further sliced the computational time on different features and drew a bar chart. Results in Figure 13(d) show that computational time of the naive and the incremental approaches only differs in the first two features. Further investigation shows that these two features have real values, unlike the other features which have integer values (0 to 255). The computational gain of incremental updates is greater for features with real values. The gain can be explained by the fact that the incremental algorithm avoids a large number of redundant focal function computations in such cases.



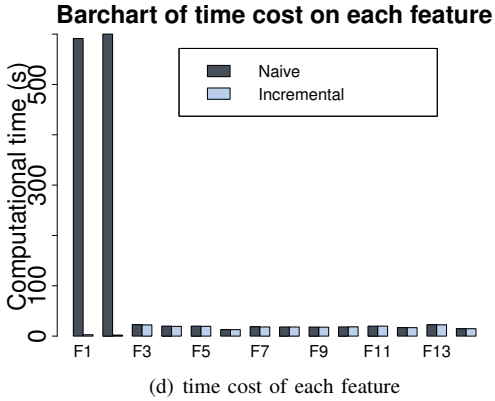
(a) patch radius=8 pixels, S_{max}=5 pixels



(b) number of patches=30,S_{max}=5 pixels



(c) number of patches =30, patch radius=10 pixels



(d) time cost of each feature

Fig. 13. Scalability of the Naive Algorithm and the Incremental Algorithm

V. Discussions

We propose a focal test based spatial decision tree to automatically address the salt-n-pepper noise issue of decision tree for raster image classification. There are some other relevant techniques to reduce salt-n-pepper noise, including pre-processing [22]), post-processing [23], and adding additional contextual varables into the input features [24]. These techniques can help reduce salt-n-pepper noise, but they are mostly non-automatic, and require extra efforts by users beyond model learning and prediction. Another increasingly popular technique is image segmentation, especially Geographic-Object-Based Image Analysis (GEOBIA) [25]. This approach also requires manual tuning by domain experts during segmentation process.

VI. Conclusions and Future Work

This paper proposed a novel approach for the spatial decision tree learning (SDTL) problem. The SDTL problem is important due to many societal applications but is challenging due to spatial autocorrelation, spatial heterogeneity, and large computational cost. We proposed an FTSDT model, which tests not only local but also focal properties of a location in decision nodes. Experimental results on a real world remote sensing dataset show that the proposed FTSDT reduces salt-and-pepper noise and improves classification accuracy, compared with local-test-based decision tree.

In future work, we plan to investigate other focal test functions, and adaptive neighborhood sizes at different localtions to better model spatial heterogeneity. We also plan to compare FTSDT with DT in tree ensembles, such as bagging and boosting.

References

[1] J. R. Quinlan, *C4.5: programs for machine learning*. Morgan kaufmann, 1993.

[2] M. A. Friedl and C. E. Brodley, "Decision tree classification of land cover from remotely sensed data," *Remote sensing of environment*, vol. 61, no. 3, pp. 399–409, Elsevier, 1997.

[3] F. Yuan, K. E. Sawaya, B. C. Loeffelholz, and M. E. Bauer, "Land cover classification and change analysis of the twin cities (minnesota) metropolitan area by multitemporal landsat remote sensing," *Remote sensing of Environment*, vol. 98, no. 2, pp. 317–328, 2005.

[4] A. Deschamps, D. Greenlee, T. Pultz, and R. Saper, "Geospatial data integration for applications in flood prediction and management in the red river basin," in *International Geoscience and Remote Sensing Symposium, Toronto, Canada. Symposium, Geomatics in the Era of RADARSAT (GER'97), Ottawa, Canada*, 2002.

[5] R. Hearne, "Evolving water management institutions in the red river basin," *Environmental Management*, vol. 40, no. 6, pp. 842–852, Springer, 2007.

[6] C. Van Westen, "Remote sensing for natural disaster management," *International Archives of Photogrammetry and Remote Sensing*, vol. 33, no. B7/4; PART 7, pp. 1609–1617, 2000.

[7] A. Akselrod-Ballin, M. Galun, R. Basri, A. Brandt, M. Gomori, M. Filippi, and P. Valsasina, "An integrated segmentation and classification approach applied to multiple sclerosis analysis," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 1122–1129.

[8] M. Celebi, H. Kingravi, Y. Aslandogan, and W. Stoecker, "Detection of blue-white veil areas in dermoscopy images using machine learning techniques," in *Proc. of SPIE Vol*, vol. 6144. Citeseer, 2006, pp. 61 445T–1.

[9] D. Bazell and D. W. Aha, "Ensembles of classifiers for morphological galaxy classification," *The Astrophysical Journal*, vol. 548, no. 1, p. 219, 2001.

[10] T. Yuan and W. Kuo, "Spatial defect pattern recognition on semiconductor wafers using model-based clustering and bayesian inference," *European Journal of Operational Research*, vol. 190, no. 1, pp. 228–240, 2008.

[11] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 21, no. 3, pp. 660–674, 1991.

[12] J. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, Springer, 1986.

[13] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[14] X. Li and C. Claramunt, "A spatial Entropy-Based decision tree for classification of geographical information," *Transactions in GIS*, vol. 10, no. 3, pp. 451–467, Blackwell Publishing Ltd, 2006.

[15] D. Stojanova, M. Ceci, A. Appice, D. Malerba, and S. Džeroski, "Global and local spatial autocorrelation in predictive clustering trees," in *Discovery Science*. Springer, 2011, pp. 307–322.

[16] D. Stojanova, M. Ceci, A. Appice, D. Malerba, and S. Dzeroski, "Dealing with spatial autocorrelation when learning predictive clustering trees," *Ecological Informatics, Elsevier*, 2012.

[17] Z. Jiang, S. Shekhar, P. Mohan, J. Knight, and J. Corcoran, "Learning spatial decision tree for geographical classification: a summary of results," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 2012, pp. 390–393.

[18] A. H. Solberg, T. Taxt, and A. K. Jain, "A markov random field model for classification of multisource satellite imagery," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 34, no. 1, pp. 100–113, 1996.

[19] M. Celik, B. M. Kazar, S. Shekhar, D. Boley, and D. J. Lilja, "Spatial dependency modeling using spatial auto-regression," in *Workshop on Geospatial Analysis and Modeling with Geoinformation Connecting Societies (GICON), International Cartography Association (ICA)*, 2006.

[20] L. Anselin, "Local indicators of spatial association-lisa," *Geographical analysis*, vol. 27, no. 2, pp. 93–115, 1995.

[21] J. R. Quinlan, "Learning decision tree classifiers," *ACM Computing Surveys (CSUR)*, vol. 28, no. 1, pp. 71–72, 1996.

[22] D. Brownrigg, "The weighted median filter," *Communications of the ACM*, vol. 27, no. 8, pp. 807–818, 1984.

[23] J. Wijnant and T. Steenberghen, "Per-parcel classification of urban ikonos imagery," in *Proceedings of 7th AGILE Conference on Geographic Information Science*, 2004, pp. 447–455.

[24] A. Puissant, J. Hirsch, and C. Weber, "The utility of texture analysis to improve per-pixel classification for high to very high spatial resolution imagery," *International Journal of Remote Sensing*, vol. 26, no. 4, pp. 733–745, 2005.

[25] G. Hay and G. Castilla, "Geographic object-based image analysis (geobia): A new name for a new discipline," in *Object-based image analysis*. Springer, 2008, pp. 75–89.