**Development of Automated Methods for Using Satellite Imagery to Monitor Changes in Vegetative Communities**

Report for Quarter 1/Task 2 in 2021

Submitted to: St. Johns River Water Management District

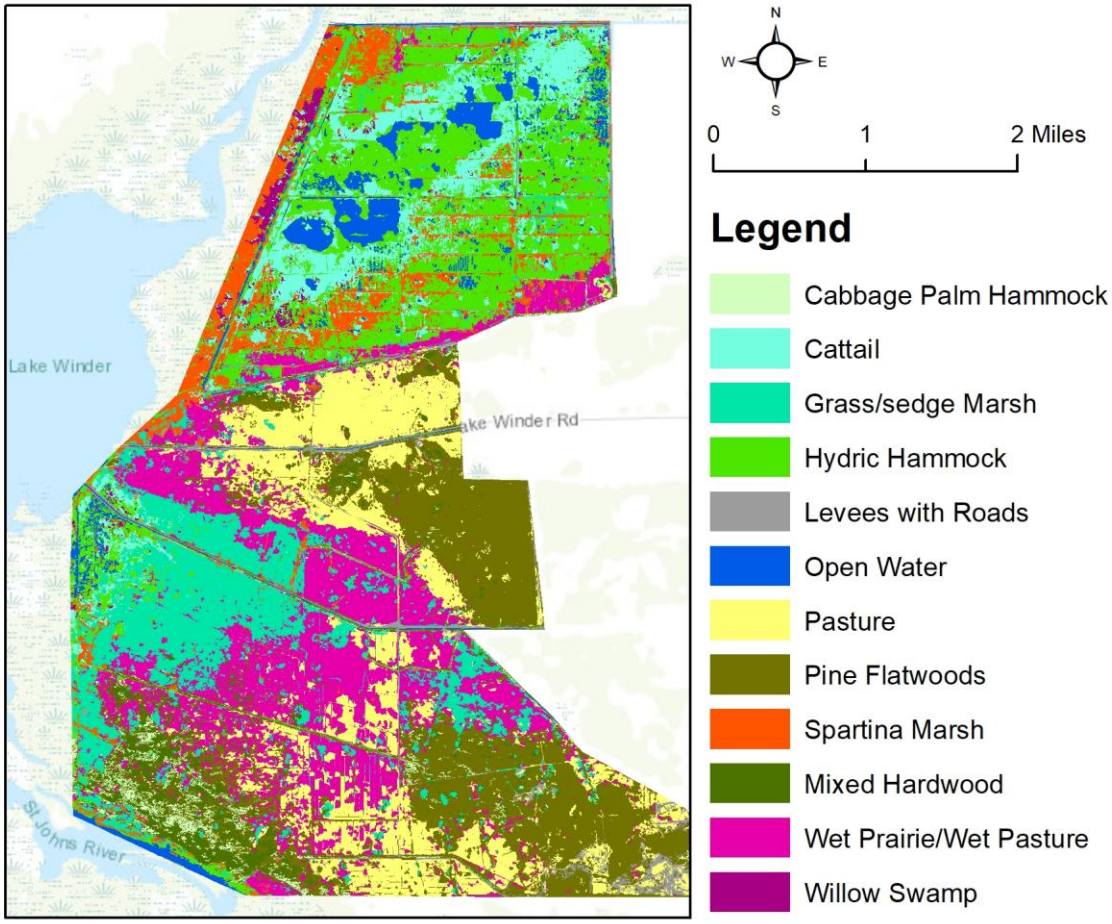
By

Caiyun Zhang, PhD

Department of Geosciences, Florida Atlantic University

777 Glades Road, Boca Raton, FL 33431, USA, Tel: 561-297-2648

Email: czhang3@fau.edu




Classified Vegetation Map for Moccasin Island from Machine Learning Ensemble Analysis

## Contents

## 1. Technical Procedures in Mapping

### 1.1 Data quality check

The St. Johns River Water Management District (District) provided non-pansharpened/pansharpened WorldView (WV)-2 imagery collected on different dates in 2017, aerial photography from 2017 used by District to create historical vegetation maps, 2017 vegetation maps, lidar digital elevation models (DEM), and some ground truth field data on January 21, 2021. The data were organized in PlantCommMap_FAU.mxd file and stored on a thumb drive by the District and sent to Florida Atlantic University (FAU). A screenshot of this mxd file is shown in Figure 1.

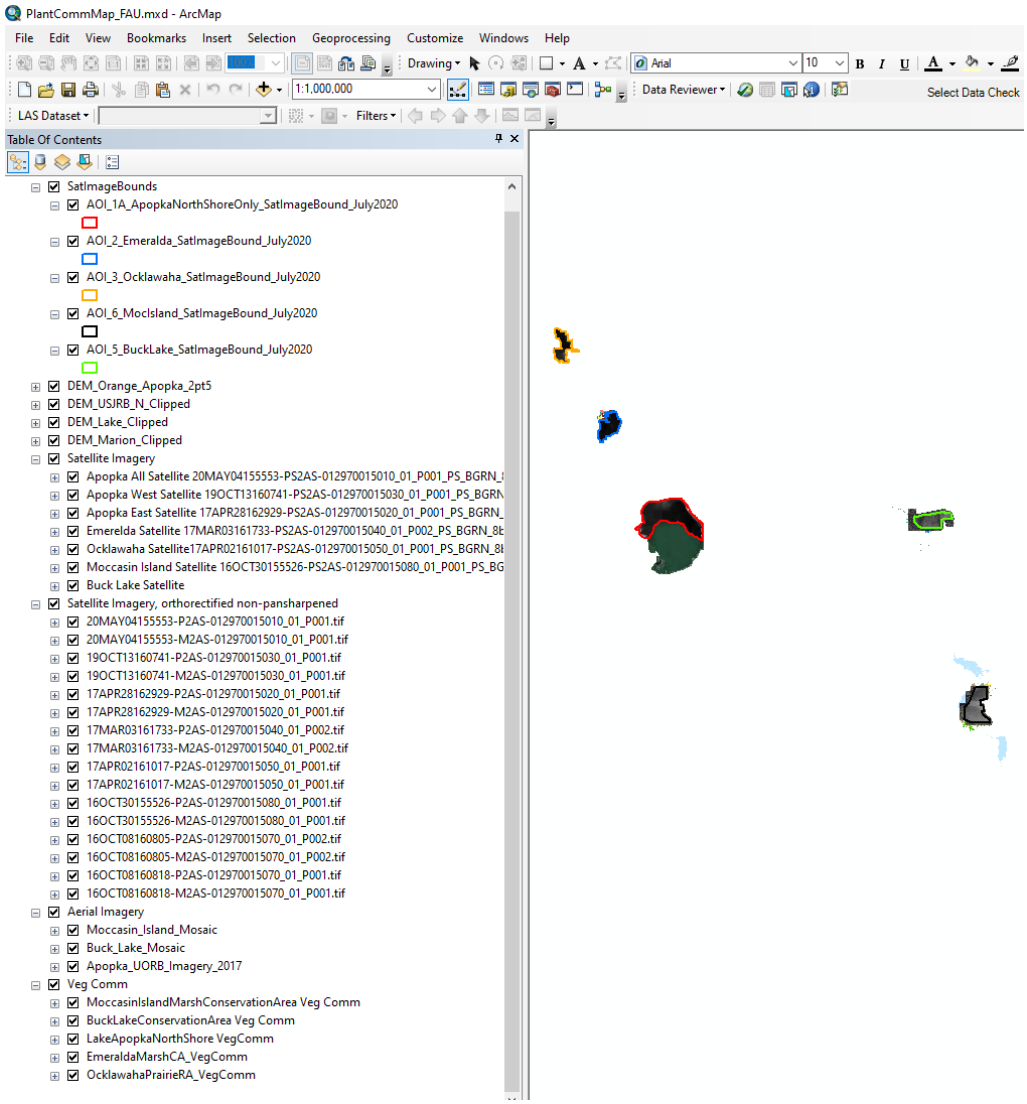



Figure 1. Screenshot of PlantCommMap_FAU.mxd: AOIs are boundaries of each mapping area; DEM are DEMs of each area; Satellite Imagery is pansharpened imagery of each area; Satellite Imagery, orthorectified non-pansharpened group is WV-2 multispectral and panchromatic imagery; Aerial Imagery group is aerial photography; Veg Comm is vegetation community maps produced from aerial photography.

In this project, we used pansharpened WV imagery to map vegetation communities. The pansharpened WV imagery has a spatial resolution of 0.3 meter and 4 spectral bands (Blue, Green, Red, and Near Infrared). In this quarter, the 4-band pansharpened imagery was used. The District recently has provided 8-band pansharpened imagery to see whether 8-band can improve future mapping results. We will focus on the application of 8-band WV image analysis in quarter 2, 2021. The pansharpened and WV multispectral imagery had been orthorectified and aligned well with the aerial photography by visual check. The imagery had also been clipped to the project areas by District and thus image preprocessing was unnecessary.

Further examination of satellite imagery and aerial photography revealed some changes due to water coverage expansion/reduction during the time period between satellite and aerial imagery collection date. This needed to be considered during training/reference sample selection.

### 1.2 Mapping technical procedures

We used pansharpened 4-band WV imagery, lidar DEM, and vegetation community map data to map vegetation communities using an object-based machine learning classification procedure. We used vegetation community data (existing maps) to help select training and testing data. WV-2 and lidar DEM were the inputs to generate the final vegetation maps. The detailed procedures are provided in the subsections below.

#### 1.2.1 Editing community names required by District

The District contract requires mapping communities based on the names in Table 1. Some names used in the existing community maps were not consistent with the required names. We thus need to edit the attribute tables of the existing community maps to match the names in Table 1. A tutorial for this is provided (Appendix 1). An example of a vegetation map with the revised community names for the Moccasin Island area is shown in Figure 2. For the classification, we needed to use a number (e.g., 1, 2, 3, ...) to indicate a specific community to facilitate later procedures. The number code used for Moccasin Island is displayed in Figure 3. The number codes were saved in a txt file so that they could be converted into the text community names in the final mapping procedure.

Table 1. Plant communities to be mapped (revised from Appendix 2 of the Statement of Work (SOW).

| **Community Names** | **Abbreviation** |
|---|---|
| Anthropogenic (AN) | AN |
| Bayhead/gall (BG) | BG |
| Bare Soil (BS) | BS |
| Cabbage Palm Hammock (CP) | CP |
| Cattail (CT) | CT |
| Cypress (CY) | CY |
| Deep Marsh (DM) | DM |
| Dry Prairie (DP) | DP |
| Forested Flatwood Depressions (FD) | FD |
| Grass/Sedge Marsh (GS) | GS |
| Hydric Hammock (HH) | HH |
| Mixed Herbaceous Marsh (HM) | HM |
| Hardwood Swamp (HS) | HS |
| Levee/Road (LR) | LR |
| Water (OW) | OW |

| | |
|---|---|
| Pasture (PA) | PA |
| Pine Flatwoods (PF) | PF |
| Phragmites (PH) | PH |
| Pine Plantation (PI) | PI |
| Palmetto Prairie (PP) | PP |
| Submerged Aquatic Beds (SAB) | SAB |
| Scrub (SC) | SC |
| Sawgrass (SG) | SG |
| Sandhill (SH) | SH |
| Salt Marsh/Salt flat (SM) | SM |
| Spartina (SP) | SP |
| Shrub Swamp (SS) | SS |
| Upland Hardwood (UH) | UH |
| Wet prairie/wet pasture (WP) | WP |
| Willow Swamp (WS) | WS |
| Anthropogenic (AN) | AN |

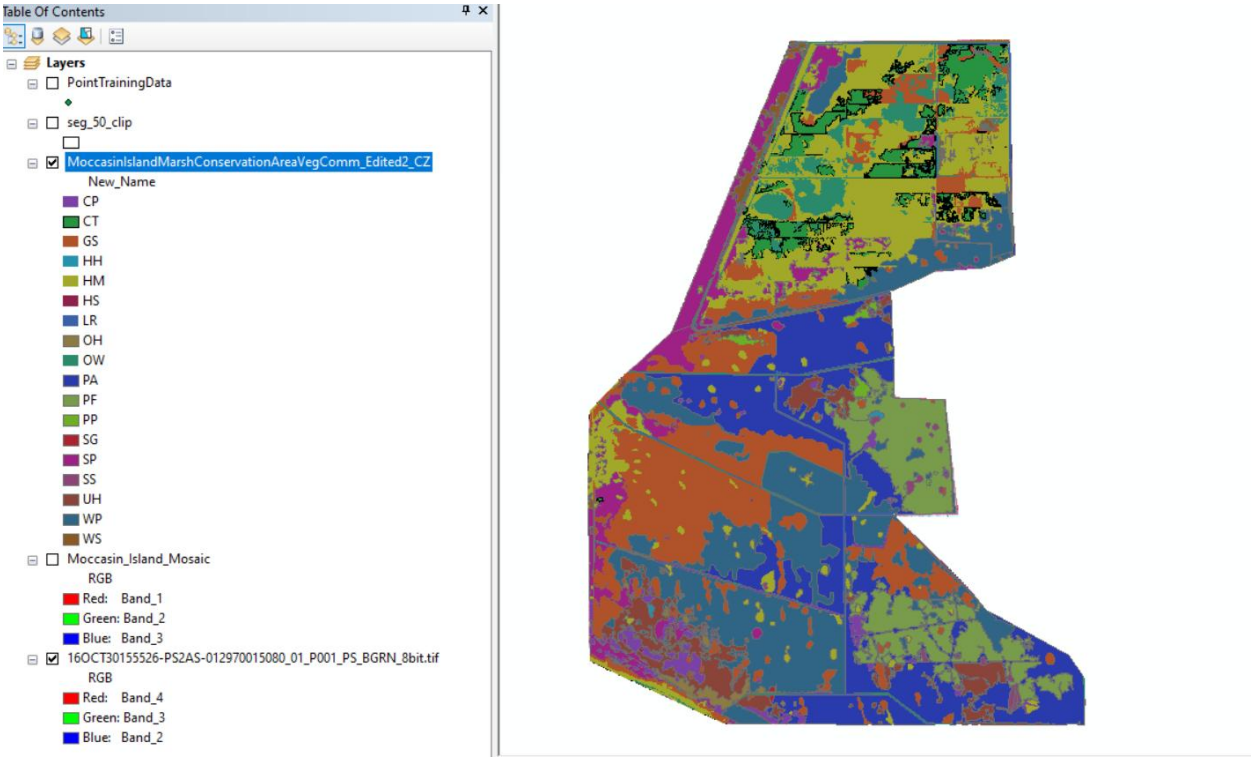



Figure 2. Community map for Moccasin Island with revised community names required by Appendix 2 in the SOW (Table 1).

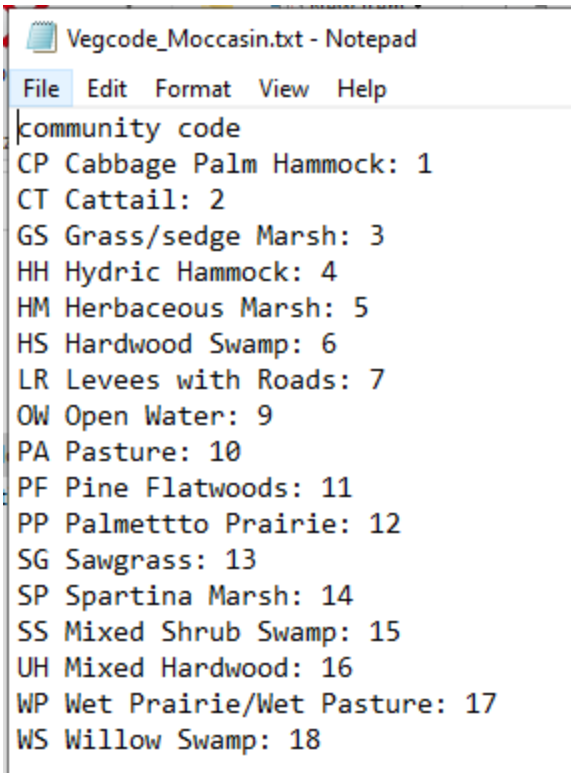



Figure 3. Communities in Moccasin Island coded as numbers for easier use in classification and labeling. There are 17 communities in the Moccasin Island area, that needed to be mapped from satellite imagery.

### 1.2.2 Segmentation

To conduct object-based classification, image objects should be created first. We segmented the pansharpened imagery using the multiresolution segmentation algorithm in eCognition with scale set to 100, and others as default, and exported all the segments as ArcGIS shape files. Features/attributes included mean, standard deviation, and texture, as shown in Figure 4. This can take hours in eCognition. To reduce the time, we split the imagery into smaller tiles and then segmented each tile one by one, and then merged the segmentation results in ArcGIS. The final segmentation file is an ArcGIS shape file with spectral and spatial attributes. After segmentation, we clipped the segmentation file to the AOI area in ArcGIS.

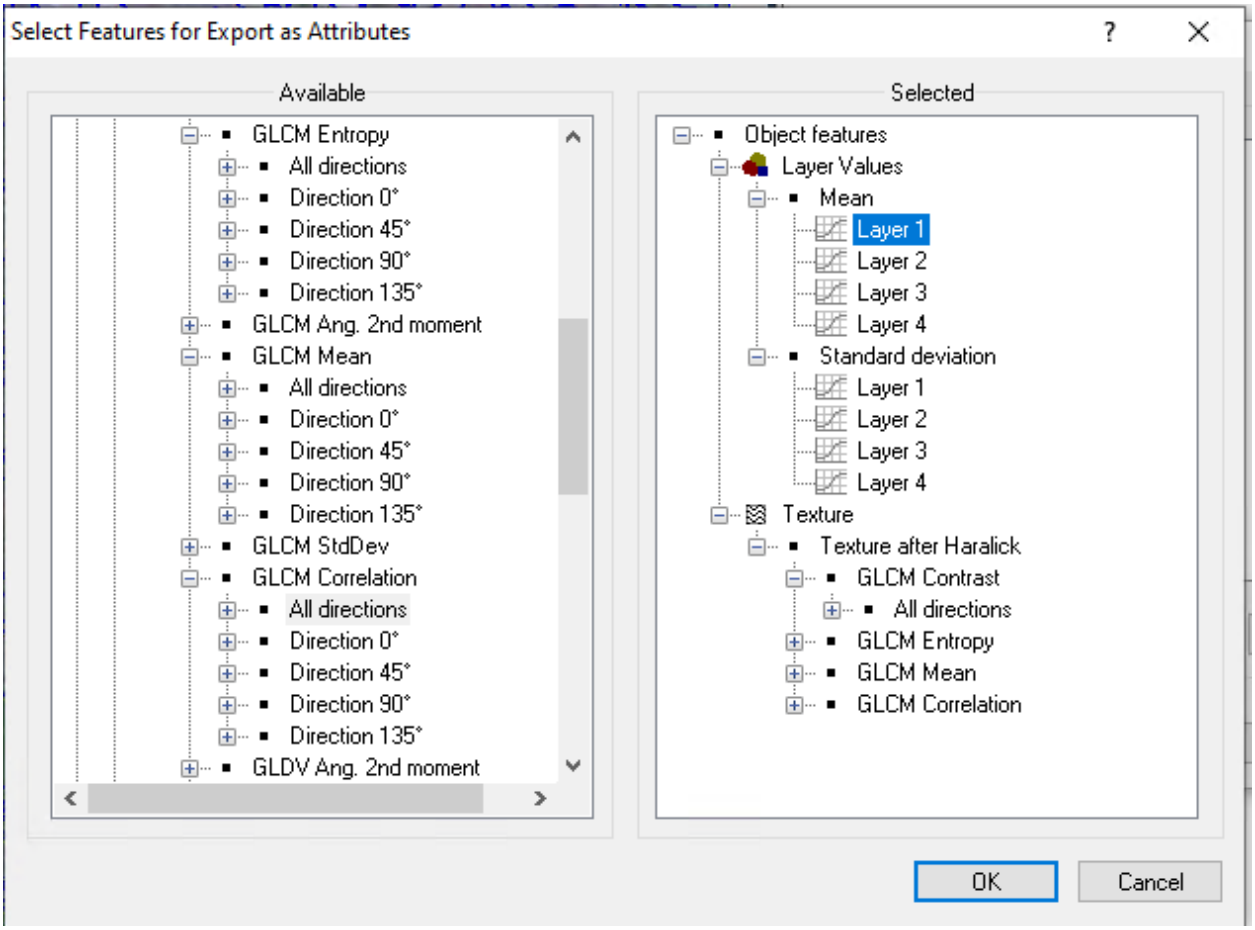
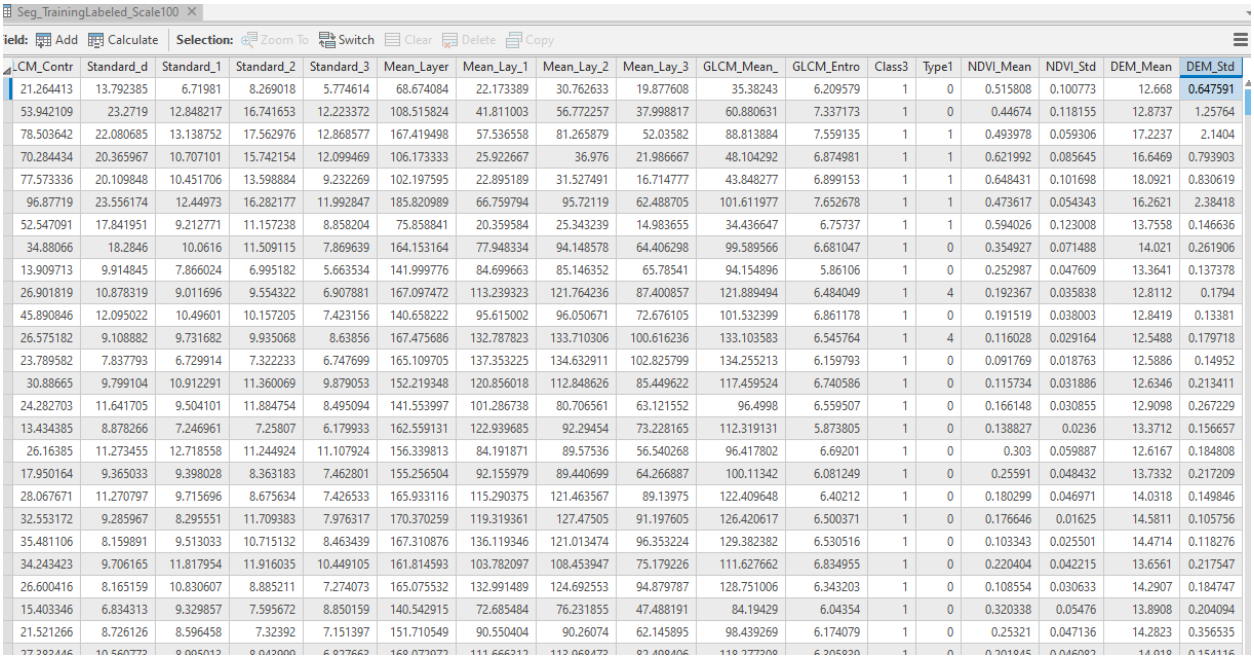


Figure 4. Segmented objects with features exported from eCognition.

Unfortunately, the District is unwilling to purchase new software (eCognition), at this time, and contractually requires the use of ArcGIS, ENVI or open-source software. Therefore, FAU will find open-source packages to replace the image segmentation part of the mapping procedure. This is discussed in Section 6 Identified Issues and Potential Solutions.

1.2.3 Extracting NDVI and lidar DEM

Using the Normalized Difference Vegetation Index (NDVI) and lidar DEM helped improve classification. In ArcGIS Pro, you can calculate NDVI using band 4 (NIR) and band 3(Red) from the 4-band pansharpened imagery. If 8-band is used, Band 7 (NIR) and 5 (Red) should be used for NDVI calculation. In ArcGIS Pro, once the imagery is loaded, the NDVI calculation can be accessed by going to Imagery->Raster Functions. After creating the NDVI imagery, the segmentation file, which is a vector polygon file, can be loaded, and then Zonal Statistics as Table can be used to get the statistical table of NDVI for each segment/polygon. You can then create two new fields as float numerical type in the segmentation file (NDVI_Mean, NDVI_Std), and join the statistical table with the segmentation results and calculate NDVI_Mean, and NDVI_Std from the statistical table. The DEM file can then be loaded and similar procedures can be used to get the mean and standard deviation of the DEM for each segment (DEM_Mean and DEM_Std). The screenshot in Figure 5 shows the attribute table of the Moccasin Island segmentation file as an example.



Seg_TrainingLabeled_Scale100

ield: Add Calculate Selection: Zoom To Switch Clear Delete Copy

| _CM_Contr | Standard_d | Standard_1 | Standard_2 | Standard_3 | Mean_Layer | Mean_Lay_1 | Mean_Lay_2 | Mean_Lay_3 | GLCM_Mean_ | GLCM_Entro | Class3 | Type1 | NDVI_Mean | NDVI_Std | DEM_Mean | DEM_Std |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 21.264413 | 13.792385 | 6.71981 | 8.269018 | 5.774614 | 68.674084 | 22.173389 | 30.762633 | 19.877608 | 35.38243 | 6.209579 | 1 | 0 | 0.515808 | 0.100773 | 12.668 | 0.647591 |
| 53.942109 | 23.2719 | 12.848217 | 16.741653 | 12.223372 | 108.515824 | 41.811003 | 56.772257 | 37.998817 | 60.880631 | 7.337173 | 1 | 0 | 0.44674 | 0.118155 | 12.8737 | 1.25764 |
| 78.503642 | 22.080685 | 13.138752 | 17.562976 | 12.868577 | 167.419498 | 57.536558 | 81.265879 | 52.03582 | 88.813884 | 7.559135 | 1 | 1 | 0.493978 | 0.059306 | 17.2237 | 2.1404 |
| 70.284434 | 20.365967 | 10.707101 | 15.742154 | 12.099469 | 106.173333 | 25.922667 | 36.976 | 21.986667 | 48.104292 | 6.874981 | 1 | 1 | 0.621992 | 0.085645 | 16.6469 | 0.793903 |
| 77.573336 | 20.109848 | 10.451706 | 13.598884 | 9.232269 | 102.197595 | 22.895189 | 31.527491 | 16.714777 | 43.848277 | 6.899153 | 1 | 1 | 0.648431 | 0.101698 | 18.0921 | 0.830619 |
| 96.87719 | 23.556174 | 12.44973 | 16.282177 | 11.992847 | 185.820989 | 66.759794 | 95.72119 | 62.488705 | 101.611977 | 7.652678 | 1 | 1 | 0.473617 | 0.054343 | 16.2621 | 2.38418 |
| 52.547091 | 17.841951 | 9.212771 | 11.157238 | 8.858204 | 75.858841 | 20.359584 | 25.343239 | 14.983655 | 34.436647 | 6.75737 | 1 | 1 | 0.594026 | 0.123008 | 13.7558 | 0.146636 |
| 34.88066 | 18.2846 | 10.0616 | 11.509115 | 7.869639 | 164.153164 | 77.948334 | 94.148578 | 64.406298 | 99.589566 | 6.681047 | 1 | 0 | 0.354927 | 0.071488 | 14.021 | 0.261906 |
| 13.909713 | 9.914845 | 7.866024 | 6.995182 | 5.663534 | 141.999776 | 84.699663 | 85.146352 | 65.78541 | 94.154896 | 5.86106 | 1 | 0 | 0.252987 | 0.047609 | 13.3641 | 0.137378 |
| 26.901819 | 10.878319 | 9.011696 | 9.554322 | 6.907881 | 167.097472 | 113.239323 | 121.764236 | 87.400857 | 121.889494 | 6.484049 | 1 | 4 | 0.192367 | 0.035838 | 12.8112 | 0.1794 |
| 45.890846 | 12.095022 | 10.49601 | 10.157205 | 7.423156 | 140.658222 | 95.615002 | 96.050671 | 72.676105 | 101.532399 | 6.861178 | 1 | 0 | 0.191519 | 0.038003 | 12.8419 | 0.13381 |
| 26.575182 | 9.108882 | 9.731682 | 9.935068 | 8.63856 | 167.475686 | 132.787823 | 133.710306 | 100.616236 | 133.103583 | 6.545764 | 1 | 4 | 0.116028 | 0.029164 | 12.5488 | 0.179718 |
| 23.789582 | 7.837793 | 6.729914 | 7.322233 | 6.747699 | 165.109705 | 137.353225 | 134.632911 | 102.825799 | 134.255213 | 6.159793 | 1 | 0 | 0.091769 | 0.018763 | 12.5886 | 0.14952 |
| 30.88665 | 9.799104 | 10.912291 | 11.360069 | 9.879053 | 152.219348 | 120.856018 | 112.848626 | 85.449622 | 117.459524 | 6.740586 | 1 | 0 | 0.115734 | 0.031886 | 12.6346 | 0.213411 |
| 24.282703 | 11.641705 | 9.504101 | 11.884754 | 8.495094 | 141.553997 | 101.286738 | 80.706561 | 63.121552 | 96.4998 | 6.559507 | 1 | 0 | 0.166148 | 0.030855 | 12.9098 | 0.267229 |
| 13.434385 | 8.878266 | 7.246961 | 7.25807 | 6.179933 | 162.559131 | 122.939685 | 92.29454 | 73.228165 | 112.319131 | 5.873805 | 1 | 0 | 0.138827 | 0.0236 | 13.3712 | 0.156657 |
| 26.16385 | 11.273455 | 12.718558 | 11.244924 | 11.107924 | 156.339813 | 84.191871 | 89.57536 | 56.540268 | 96.417802 | 6.69201 | 1 | 0 | 0.303 | 0.059887 | 12.6167 | 0.184808 |
| 17.950164 | 9.365033 | 9.398028 | 8.363183 | 7.462801 | 155.256504 | 92.155979 | 89.440699 | 64.266887 | 100.11342 | 6.081249 | 1 | 0 | 0.25591 | 0.048432 | 13.7332 | 0.217209 |
| 28.067671 | 11.270797 | 9.715696 | 8.675634 | 7.426533 | 165.933116 | 115.290375 | 121.463567 | 89.13975 | 122.409648 | 6.40212 | 1 | 0 | 0.180299 | 0.046971 | 14.0318 | 0.149846 |
| 32.553172 | 9.285967 | 8.295551 | 11.709383 | 7.976317 | 170.370259 | 119.319361 | 127.47505 | 91.197605 | 126.420617 | 6.500371 | 1 | 0 | 0.176646 | 0.01625 | 14.5811 | 0.105756 |
| 35.481106 | 8.159891 | 9.513033 | 10.715132 | 8.463439 | 167.310876 | 136.119346 | 121.013474 | 96.353224 | 129.382382 | 6.530516 | 1 | 0 | 0.103343 | 0.025501 | 14.4714 | 0.118276 |
| 34.243423 | 9.706165 | 11.817954 | 11.916035 | 10.449105 | 161.814593 | 103.782097 | 108.453947 | 75.179226 | 111.627662 | 6.834955 | 1 | 0 | 0.220404 | 0.042215 | 13.6561 | 0.217547 |
| 26.600416 | 8.165159 | 10.830607 | 8.885211 | 7.274073 | 165.075532 | 132.991489 | 124.692553 | 94.879787 | 128.751006 | 6.343203 | 1 | 0 | 0.108554 | 0.030633 | 14.2907 | 0.184747 |
| 15.403346 | 6.834313 | 9.329857 | 7.595672 | 8.850159 | 140.542915 | 72.685484 | 76.231855 | 47.488191 | 84.19429 | 6.04354 | 1 | 0 | 0.320338 | 0.05476 | 13.8908 | 0.204094 |
| 21.521266 | 8.726126 | 8.596458 | 7.32392 | 7.151397 | 151.710549 | 90.550404 | 90.26074 | 62.145895 | 98.439269 | 6.174079 | 1 | 0 | 0.25321 | 0.047136 | 14.2823 | 0.356535 |

Figure 5. Attribute table of the segmentation file of Moccasin Island in ArcGIS with NDVI and DEM mean and standard deviation value extracted for each segment.

### 1.2.4 Training/testing sample selection

The District did not have adequate ground truth point data from mapping the aerial imagery to train the classification model, thus we needed to collect training/testing samples from the satellite imagery. The training/testing/reference data was selected using the existing vegetation maps and satellite imagery. This is an important step in the entire classification. Incorrect reference samples will result in unacceptable classification and final maps. We can select training objects from the segments in ArcGIS Pro. Here are steps for reference sample selection in ArcGIS.

1) Create an integer field named Class in the segmentation file, which will be used to label the reference objects.
2) Label the selected samples for each class (1, 2, 3…) following the numbers you were using in Figure 3.
   - Set the segmentation file as the only selectable layer in ArcGIS Pro
   - For each community reference sample selection, first look at the existing vegetation map labeled as classes 1, 2, 3…and get general idea about its distribution and its size
   - Select reference samples class by class. For example, for class 1, look at the satellite and aerial imagery and determine what class 1 looks like on the imagery, and then select segments which are located within the class 1 community and not the edge of this community based on the reference map. Also don't select segments which have changed due to the time gap between the collection of the aerial and satellite imagery (e.g., Herbaceous Marsh that has become Open Water due to flooding). Set the Interactive Selection Method (Selection->Interactive Selection Method) to Add to Current Selection. This setting will continue the selection for the

same community unless reset. If you accidently select a wrong sample, use Remove From Current Selection to drop the wrong sample.

- Once you have collected adequate training samples for class 1 (see *Tip*), you then label these selected segments with the vegetation code (e.g., 1) using the Field Calculator function in the attribute table. After labeling class 1 training samples, clear the selection.
- Repeat the above procedure and complete the reference sample selections for all communities. Training samples should be selected for all the targeted communities

*Tip:* Training samples should be well distributed in the project area to catch the variation of each community (for example, pine density can be different). Samples should be visually easier to identify on the imagery (e.g., forest should be very different from marshes and the selected samples make sense); Each training object should be located within the community rather than at the edge of each community based on the reference map. Adequate training samples should be collected for each community. The number of training samples depends on the coverage of each community in the AOI based on the existing map, and the variation within each community. For example, variation within water is usually small and thus a small number of water samples can be selected, even though water may have a large coverage. Communities covering larger areas should contain more training samples. Communities covering only a small portion of the total area should have fewer reference samples. However, adequate sampling of all communities is necessary for calculating the error matrix for statistical analysis. Very minor communities in the AOI may have to be ignored if they cannot produce adequate training samples. Visually check each training object, comparing between the satellite imagery, aerial photography, and ArcGIS base imagery, and label each object with the number code you are using for each community. If training samples are not correctly collected, results will be unacceptable. Thus this step is crucial. Also note that due to the large volume of segments, selecting training samples from segments directly is challenging. The segments can be split into smaller parts to reduce the data size using ArcGIS selection function and make it easier to handle. However, training sample selection is a manual procedure. Consequently, GPU or parallel computation will not make this step more efficient.

### 1.2.5 Exporting attribute table data for machine learning model training

ArcGIS Pro only utilizes Support Vector Machine (SVM) and Random Tree machine learning classifiers and only supports its own data format and wizard for classification. Random Forest (RF), Artificial Neural Network (ANN), Support Vector Machine (SVM), and k-Nearest Neighbor (k-NN) algorithms have all been shown to be effective classifiers. Therefore, open source machine learning algorithms for training and classification using these four classification procedures will be used to produce the final maps. After the training sample selection in ArcGIS, select and export the training data in ArcGIS as an ASCII file which will be edited and used in R, Python, or WEKA for the machine learning training and testing. Only export features/attributes to be used for classification and be sure to include the Class field which is the reference class labeled during the training sample selection. In the machine learning training procedure, Random Forest (RF), Artificial Neural Network (ANN), Support Vector Machine (SVM), and k-Nearest Neighbor (k-NN) algorithms, error matrices, Kappa statistics, and producer's and user's accuracies can be created using k-fold cross validation. If the overall accuracy (OA) is above 85%, then the model can be used to predict the plant communities.

1.2.6 Final classification and mapping

For final classification, export the segmentation attribute table as an ASCII file and edit it to the required data format for use in R, Python, or WEKA, and then use that data as the input for the community classification prediction. Using the trained machine learning model, and input attributes of the entire project area, a final classification is determined for each object input attribute in R, Python or WEKA. Join the classification to the segmentation file using the FID or other unique ID you create, and calculate the prediction class for each segment in ArcGIS from each classifier, then dissolve the class and generate a smaller shapefile to be used for mapping.

1.2.7 Ensemble analysis

If classifiers produce a comparable accuracy, an ensemble analysis (EA) can be conducted to give a more robust classification map. The idea is that if a majority of classifiers vote that a polygon should be in a given class, then the polygon should be labeled as this class; if all classifiers vote for completely different classes, then the final class should be the class which has the highest producer's accuracy, which was obtained during the analysis of the training/testing samples in the training/testing procedure. In this project, three classification results were combined because combining four classifications was too difficult.

## 2. Results for Moccasin Island Area

This section provides the results for Moccasin Island area using segmentation results from eCognition software.

### 2.1 Communities to be mapped

For this area, MS and TS were mapped as SS; MH and OH were mapped as UH; and CA and FF were mapped as OW. After editing the existing vegetation map for Moccasin Island, there were 17 communities for this area. Classes Hydric Hammock (HH), Hardwood Swamp (HS), Palmetto Prairie (PP), Sawgrass (SG) and Mixed Shrub Swamp (SS) are very minor communities (as shown in Figure 6) and difficult to classify because adequate training samples could not be collected for these communities. Preliminary testing results showed that an inclusion of training samples from these minor communities produced a very low accuracy. Thus, they were eliminated from the classification.

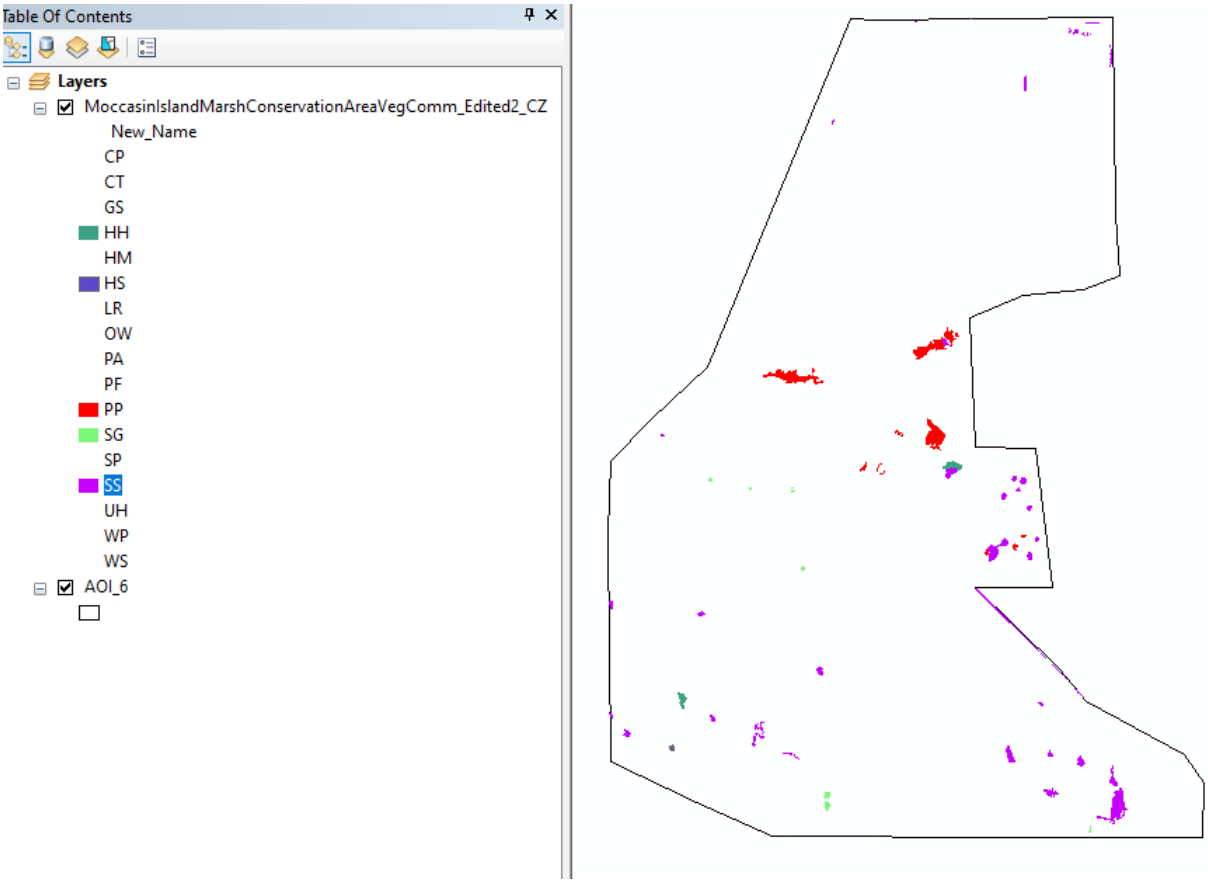



Figure 6. Minor communities HH, HS, PP, SG, and SS in Moccasin Island area. These communities were difficult to classify due to inadequate training samples and were therefore eliminated from the classification.

## 2.2 Segmentation

In eCognition, a total of 88,471 image segments were generated from the multiresolution segmentation algorithm. The size of segmentation is manageable in ArcGIS and the segmentation results are reasonable based on visual check. Figure 7 shows the segmentation results from the multiresolution segmentation algorithm.

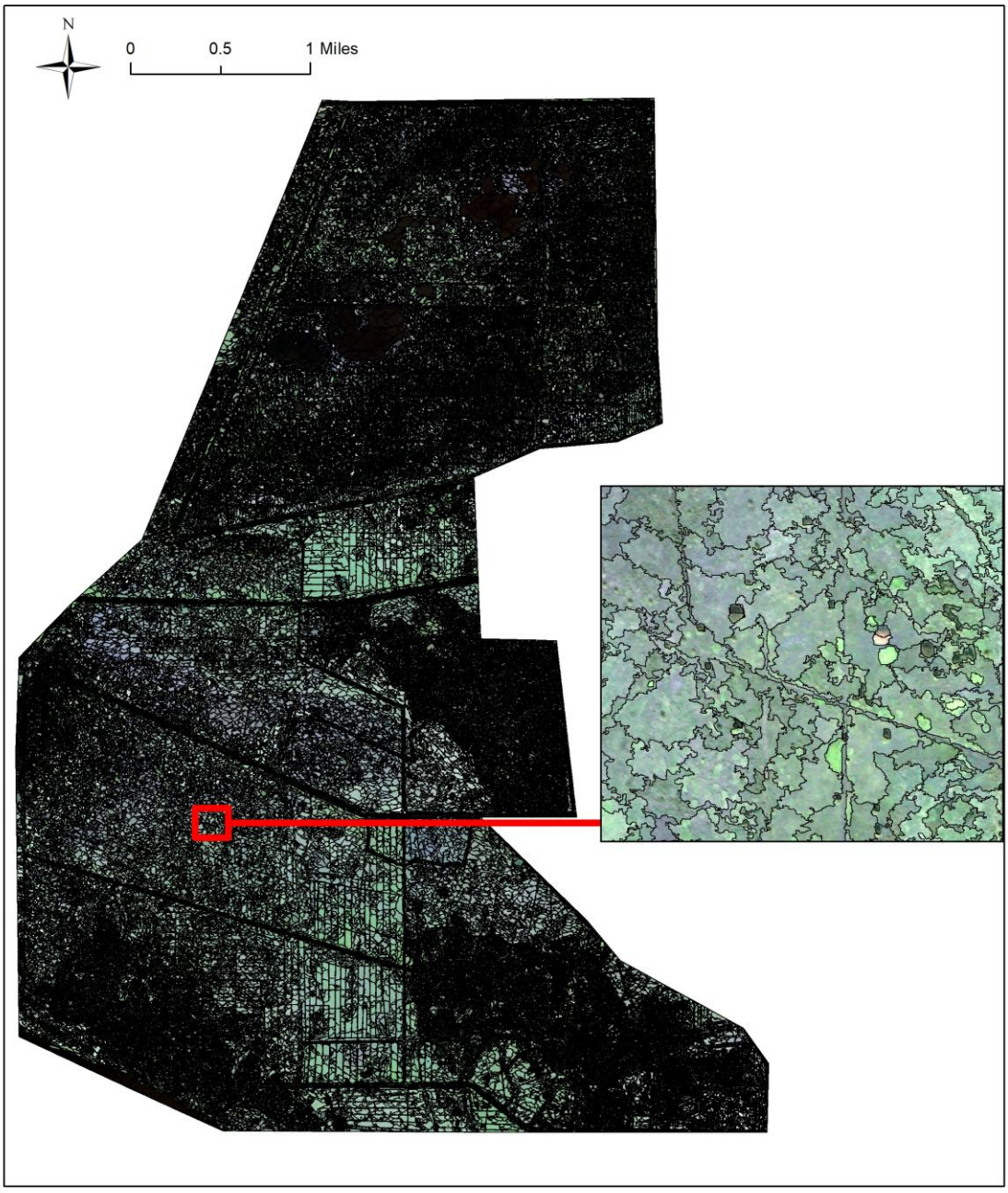



Figure 7. Multiresolution segmentation results from eCognition (scale=100, and others set as default).

### 2.3 Selected reference samples

A total of 3720 reference samples was collected for 12 communities to be used as training and testing samples for classification, as shown in Figure 8.

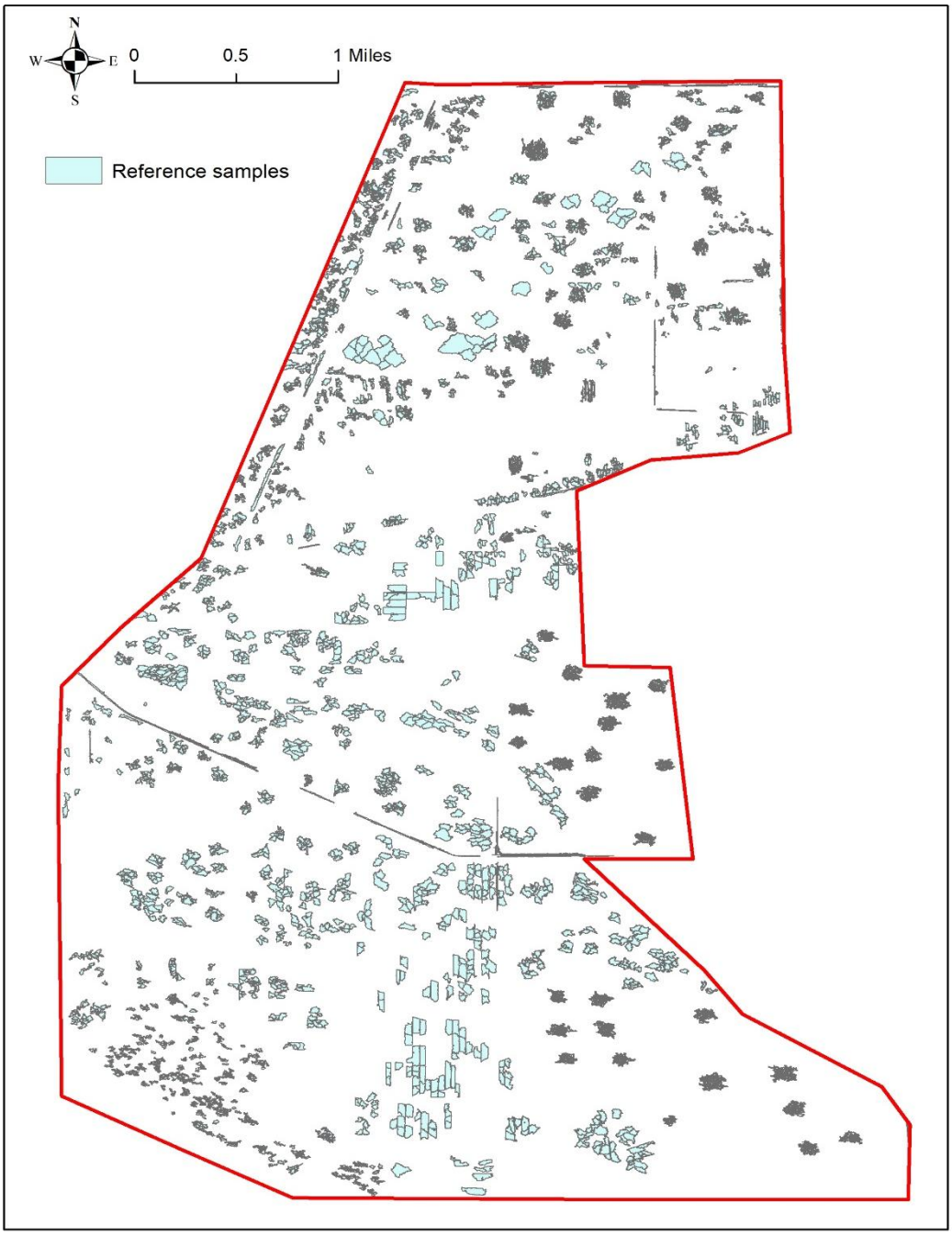



Figure 8. Selected reference samples for Moccasin Island area classification.

### 2.4 Accuracy assessment

The performance of classifiers ANN, kNN, SVM, RF based on 10-fold cross validation is displayed in Table 2. All achieved an overall accuracy (OA) above 80%, and SVM achieved an accuracy above 85%. McNemar tests show that there is no significant difference between ANN and kNN (Table 3). Ensemble analysis (EA) was conducted to integrate outputs from ANN, SVM, and RF, and produced an OA of 84.8% and Kappa value of 0.83. Comparison with an inclusion/exclusion of lidar DEM in the classification is also displayed in Table 2. An addition of lidar DEM increased the OA around 10%. This is important in the mapping procedure. Thus in the final classification for mapping, lidar DEM was included. The error matrix from ANN, SVM, RF and ensemble analysis are provided in Tables 4-7. Producer's Accuracy (PA), User's Accuracy (UA), OA, and Kappa values are also provided in these tables.

Table 2. Performance of each machine learning classifier and ensemble analysis with/without lidar DEM. EA includes lidar DEM only.

| | ANN | kNN | SVM | RF | EA |
|---|---|---|---|---|---|
| Overall Accuracy (%) | 80.0/71.0 | 80.1/69.3 | 85.2/75.8 | 82.6/72.4 | 84.8 |
| Kappa Value | 0.77/0.67 | 0.78/0.65 | 0.83/0.73 | 0.80/0.69 | 0.83 |

Table 3. McNemar test between different classifiers (* significance with 95% confidence when z-Score is greater than 1.96). ANN and kNN are not significantly different. Thus the ensemble analysis was conducted using only ANN, SVM, and RF.

| | ANN/kNN | ANN/SVM | ANN/RF | SVM/RF |
|---|---|---|---|---|
| z-Score | 0.6 | 9.6* | 4.3* | 4.6* |

Table 4. Error matrix of ANN classification. Column represents reference and row represents classification result.

| | CP | CT | GS | HM | LR | OW | PA | PF | SP | UH | WP | WS | Column Total | UA (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CP | 69 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 1 | 0 | 0 | 81 | 85.2 |
| CT | 0 | 311 | 4 | 51 | 0 | 1 | 0 | 0 | 14 | 0 | 0 | 14 | 395 | 78.7 |
| GS | 0 | 8 | 236 | 20 | 2 | 0 | 15 | 0 | 23 | 1 | 67 | 8 | 380 | 62.1 |
| HM | 0 | 79 | 7 | 393 | 2 | 2 | 0 | 0 | 26 | 1 | 3 | 11 | 524 | 75.0 |
| LR | 2 | 2 | 1 | 0 | 61 | 2 | 3 | 9 | 5 | 0 | 7 | 0 | 92 | 66.3 |
| OW | 0 | 0 | 0 | 5 | 0 | 101 | 0 | 0 | 0 | 0 | 0 | 1 | 107 | 94.4 |
| PA | 0 | 0 | 0 | 0 | 0 | 0 | 307 | 1 | 0 | 0 | 64 | 0 | 372 | 82.5 |
| PF | 2 | 0 | 1 | 1 | 3 | 0 | 3 | 614 | 0 | 23 | 1 | 0 | 648 | 94.8 |
| SP | 0 | 15 | 13 | 19 | 5 | 0 | 1 | 0 | 222 | 0 | 14 | 6 | 295 | 75.3 |
| UH | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 19 | 0 | 170 | 0 | 0 | 191 | 89.0 |
| WP | 0 | 2 | 28 | 10 | 4 | 0 | 74 | 0 | 4 | 0 | 417 | 0 | 539 | 77.4 |
| WS | 0 | 5 | 0 | 15 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 74 | 96 | 77.1 |
| Row Total | 81 | 443 | 360 | 499 | 90 | 104 | 380 | 643 | 287 | 191 | 550 | 92 | 3720 | |
| PA (%) | 93.2 | 73.7 | 81.1 | 76.5 | 79.2 | 95.3 | 76.2 | 93.9 | 75.0 | 86.7 | 72.8 | 64.9 | OA: 80.0% Kappa: 0.77 | |

Table 5. Error matrix of SVM classification. Column represents reference and row represents classification result.

| | CP | CT | GS | HM | LR | OW | PA | PF | SP | UH | WP | WS | Column Total | UA (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CP | 76 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 81 | 93.8 |
| CT | 0 | 335 | 1 | 41 | 0 | 1 | 0 | 0 | 12 | 0 | 1 | 4 | 395 | 84.8 |
| GS | 0 | 8 | 295 | 17 | 2 | 0 | 0 | 0 | 10 | 0 | 46 | 2 | 380 | 77.6 |
| HM | 0 | 78 | 7 | 412 | 0 | 1 | 0 | 0 | 20 | 0 | 1 | 5 | 524 | 78.6 |
| LR | 0 | 0 | 2 | 2 | 81 | 1 | 0 | 3 | 1 | 0 | 2 | 0 | 92 | 88.0 |
| OW | 0 | 1 | 0 | 4 | 1 | 101 | 0 | 0 | 0 | 0 | 0 | 0 | 107 | 94.4 |
| PA | 0 | 0 | 2 | 0 | 0 | 0 | 321 | 1 | 0 | 0 | 48 | 0 | 372 | 86.3 |
| PF | 5 | 0 | 2 | 0 | 4 | 0 | 2 | 619 | 0 | 15 | 1 | 0 | 648 | 95.5 |
| SP | 0 | 12 | 17 | 15 | 0 | 0 | 0 | 0 | 237 | 0 | 12 | 2 | 295 | 80.3 |
| UH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 15 | 0 | 176 | 0 | 0 | 191 | 92.1 |
| WP | 0 | 2 | 33 | 2 | 2 | 0 | 57 | 0 | 4 | 0 | 439 | 0 | 539 | 81.4 |
| WS | 0 | 7 | 1 | 6 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 79 | 96 | 82.3 |
| Row Total | 81 | 443 | 360 | 499 | 90 | 104 | 380 | 643 | 287 | 191 | 550 | 92 | 3720 | |
| PA (%) | 93.8 | 75.6 | 81.9 | 82.6 | 90.0 | 97.1 | 84.5 | 96.3 | 82.6 | 92.1 | 79.8 | 85.9 | OA: 85.2% Kappa: 0.83 | |

Table 6. Error matrix of RF classification. Column represents reference and row represents classification result.

| | CP | CT | GS | HM | LR | OW | PA | PF | SP | UH | WP | WS | Column Total | UA (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CP | 70 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 0 | 81 | 86.4 |
| CT | 0 | 313 | 1 | 48 | 0 | 0 | 0 | 0 | 21 | 0 | 0 | 12 | 395 | 79.2 |
| GS | 0 | 8 | 288 | 17 | 0 | 0 | 1 | 1 | 17 | 1 | 43 | 4 | 380 | 75.8 |
| HM | 0 | 68 | 9 | 415 | 0 | 2 | 0 | 0 | 25 | 0 | 2 | 3 | 524 | 79.2 |
| LR | 0 | 0 | 4 | 3 | 61 | 1 | 4 | 6 | 4 | 0 | 9 | 0 | 92 | 66.3 |
| OW | 0 | 0 | 0 | 6 | 0 | 101 | 0 | 0 | 0 | 0 | 0 | 0 | 107 | 94.4 |
| PA | 0 | 0 | 1 | 0 | 0 | 0 | 297 | 0 | 0 | 0 | 74 | 0 | 372 | 79.8 |
| PF | 2 | 0 | 1 | 1 | 1 | 0 | 1 | 628 | 0 | 12 | 2 | 0 | 648 | 96.9 |
| SP | 0 | 13 | 16 | 16 | 0 | 0 | 0 | 0 | 233 | 0 | 16 | 1 | 295 | 79.0 |
| UH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 41 | 0 | 150 | 0 | 0 | 191 | 78.5 |
| WP | 0 | 0 | 22 | 7 | 1 | 0 | 42 | 0 | 7 | 0 | 460 | 0 | 539 | 85.3 |
| WS | 0 | 23 | 0 | 13 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 58 | 96 | 60.4 |
| Row Total | 72 | 425 | 342 | 526 | 63 | 104 | 356 | 676 | 309 | 163 | 606 | 78 | 3720 | |
| PA (%) | 97.2 | 73.6 | 84.2 | 78.9 | 96.8 | 97.1 | 86.1 | 91.4 | 75.4 | 92.0 | 75.9 | 74.4 | OA: 82.6% Kappa: 0.8 | |

Table 7. Error matrix of ensemble analysis. Column represents reference and row represents classification result.

| | CP | CT | GS | SP | LR | UH | LR | WS | OW | PA | PF | WP | Column Total | UA (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CP | 73 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 81 | 90.1 |
| CT | 0 | 328 | 2 | 12 | 44 | 0 | 0 | 7 | 1 | 0 | 0 | 1 | 395 | 83.0 |
| GS | 0 | 7 | 290 | 15 | 20 | 0 | 1 | 3 | 0 | 1 | 1 | 42 | 380 | 76.3 |
| SP | 0 | 12 | 15 | 237 | 16 | 0 | 0 | 2 | 0 | 0 | 0 | 13 | 295 | 80.3 |
| LR | 0 | 76 | 7 | 23 | 409 | 0 | 0 | 5 | 2 | 0 | 0 | 2 | 524 | 78.1 |
| UH | 0 | 0 | 0 | 0 | 0 | 171 | 0 | 0 | 0 | 0 | 20 | 0 | 191 | 89.5 |
| LR | 0 | 0 | 2 | 4 | 0 | 0 | 77 | 0 | 2 | 0 | 3 | 4 | 92 | 83.7 |
| WS | 0 | 6 | 1 | 3 | 8 | 0 | 0 | 78 | 0 | 0 | 0 | 0 | 96 | 81.3 |
| OW | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 101 | 0 | 0 | 0 | 107 | 94.4 |
| PA | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 318 | 0 | 52 | 372 | 85.5 |
| PF | 2 | 0 | 2 | 0 | 1 | 15 | 2 | 0 | 0 | 1 | 624 | 1 | 648 | 96.3 |
| WP | 0 | 1 | 22 | 3 | 8 | 0 | 2 | 0 | 0 | 52 | 0 | 451 | 539 | 83.7 |
| Row Total | 75 | 430 | 343 | 297 | 511 | 186 | 83 | 95 | 106 | 372 | 656 | 566 | 3720 | |
| PA (%) | 97.3 | 76.3 | 84.6 | 79.8 | 80.0 | 91.9 | 92.8 | 82.1 | 95.3 | 85.5 | 95.1 | 79.7 | OA: 84.8% Kappa: 0.83 | |

### 2.5 Vegetation community mapping

Classification maps from machine learning algorithms, and ensemble analysis are displayed in Figure 9. In general, all classifiers produced a consistent spatial pattern with the existing map, but classification maps provide a more realistic/natural edge of each patch and presence of small patches within large communities. For most regions, three classifiers generated consistent classification, as shown in green in the uncertainty map in Figure 9(G). Red regions show a "warning sign" where classification from three classifiers are completely inconsistent.

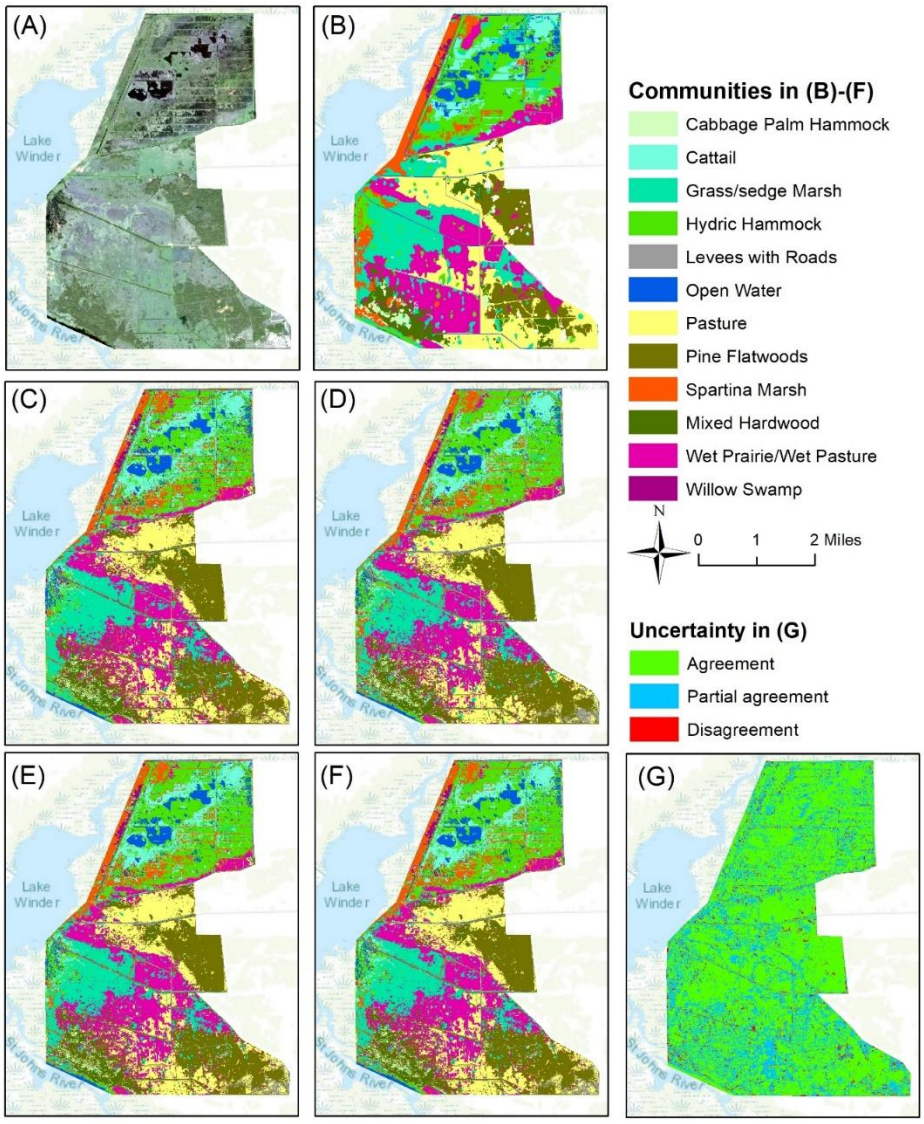



Figure 9. (A) WV-2 pansharpened imagery; (B) reference map from aerial photography; Object-based classifications from (C) ANN, (D) SVM, (E) RF, and (F) Ensemble analysis. (G) Uncertainty from ANN, SVM, and RF. Agreement: achieved full agreement from three classifiers; partial agreement: achieved a majority vote from three classifiers (two classifiers vote the same class); Disagreement: three classifiers vote completely different classes and no consensus.

## 3. Results for Emeralda Marsh Area

This section provides the results for Emeralda Marsh area using segmentation algorithm in eCognition software.

### 3.1 Communities to be mapped

For this area, BD, RE, and PL were mapped as AN; BH was mapped as BG; BA was mapped as BS; DM-N and DM-LS were mapped as DM; RD and LV were mapped as LR; FF and W were mapped as OW; AB was mapped as SAB; TS was mapped as SS; and U was mapped as UH. Shallow Marsh was mapped as Mixed Herbaceous Marsh (HM). After editing, there were 16 communities to be mapped.

### 3.2 Segmentation

A total of 9,361 image segments was generated from the multiresolution segmentation algorithm in eCognition. This size is manageable in ArcGIS and the segmentation results are reasonable based on visual check. Figure 10 shows the segmentation results.

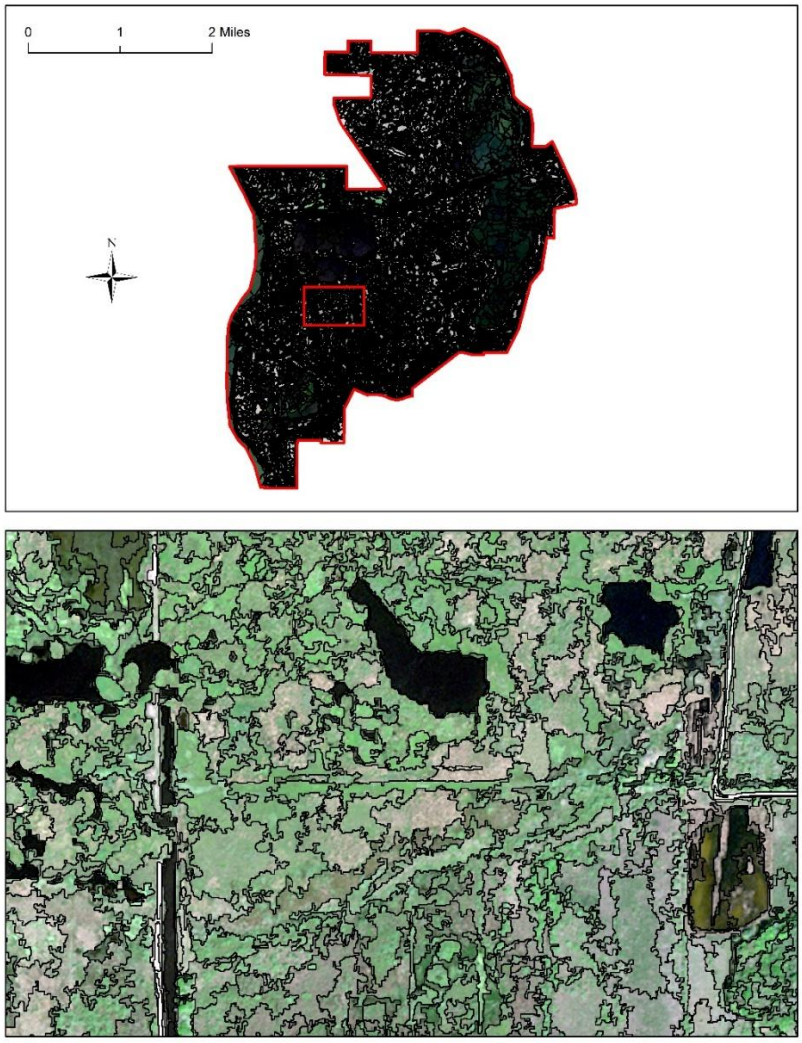



Figure 10. Multiresolution segmentation results from eCognition (scale=100 and others set as default).

### 3.3 Selected reference samples

Training/reference sample selection is a labor intensive procedure that necessitates directly picking training objects from the segmentation vector polygon file which has a large volume. Consequently, we developed an easier method to select training samples using point features. The steps are below.

- Create a new point vector layer in ArcGIS->View->Catalog Pane, and make sure the projection of this point layer is consistent with the current layers (i.e. the existing veg maps, using NAD_1983_HARN_UTM_Zone_17N), and then add a new attribute "Class" as an integer for this point layer, which will be used to label the community.
- Open this point layer in ArcGIS Pro and start adding points for each community based on the existing vegetation map (the one you have edited with new community names), and the satellite imagery. Similar to the polygon-based training sample selection, highlight the community where you want to put training points and note its distribution and the coverage of this community, then start adding training points in Edit->Create. Make sure each point is within each community by changing each community symbol (the existing vegetation map) to be hollow, and checking the underlying imagery, to see whether each training sample point added looks like the intended community. After creating points for each community, label the points with the number code developed for each community as shown in Figure 3. Also make sure these training points are well spread over the AOI where the community is located. Figure 11 displays an example of the points we added for the Emeralda Marsh area.

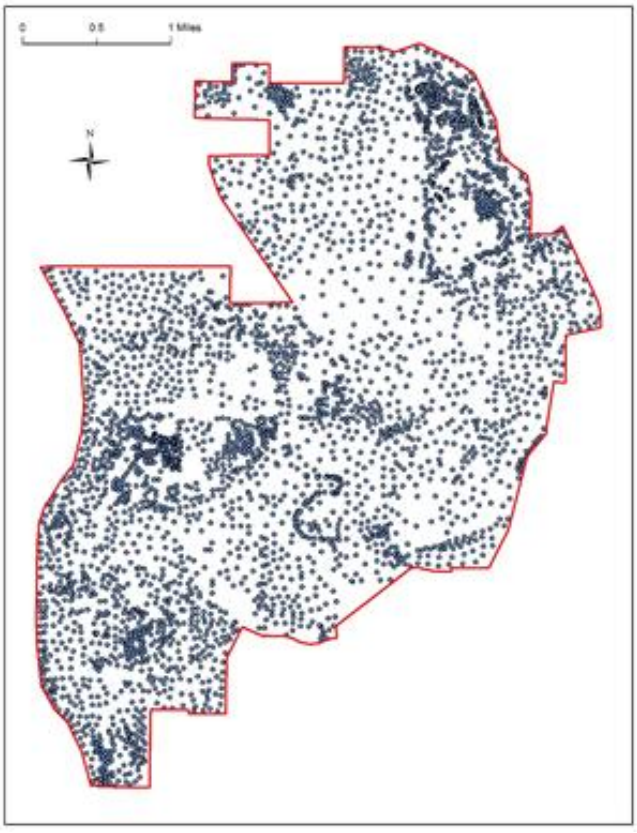


Figure 11. Labeled points for Emeralda Marsh area to be used for collecting training data.

- After editing the point training data layer, the point layer can then be spatially joined with the segmentation polygon layer, and then exported. In this way, the training objects/segments can be identified. The interface for the spatial join between the point layer and segmentation polygon layer is displayed in Figure 12, and the final training objects for the Emeralda Marsh area are shown in Figure 13.

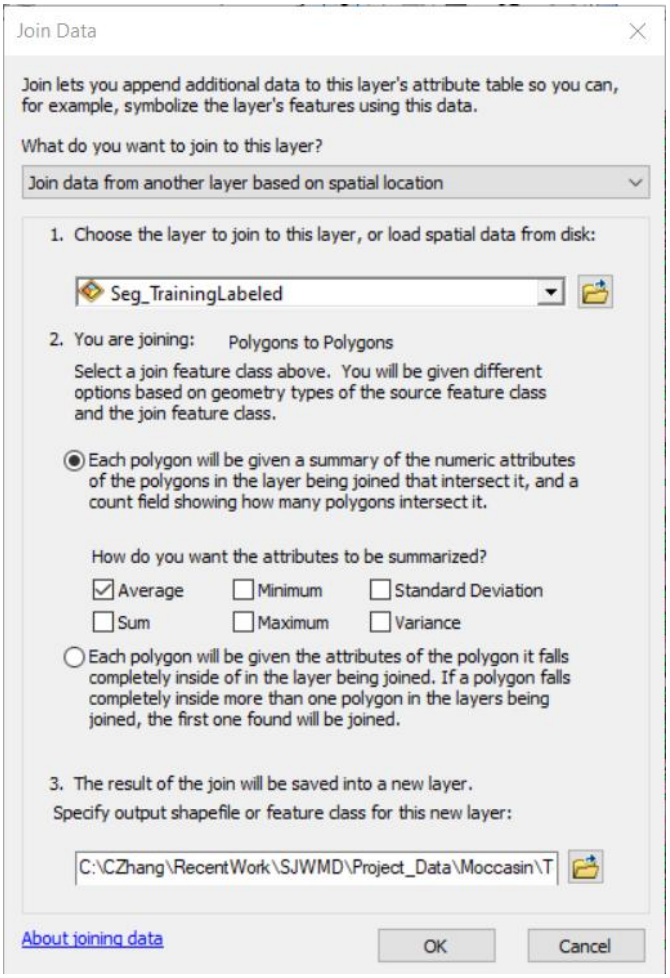



Figure 12. Spatially join the segmentation polygon file with the labeled point layer to identify the training polygons.

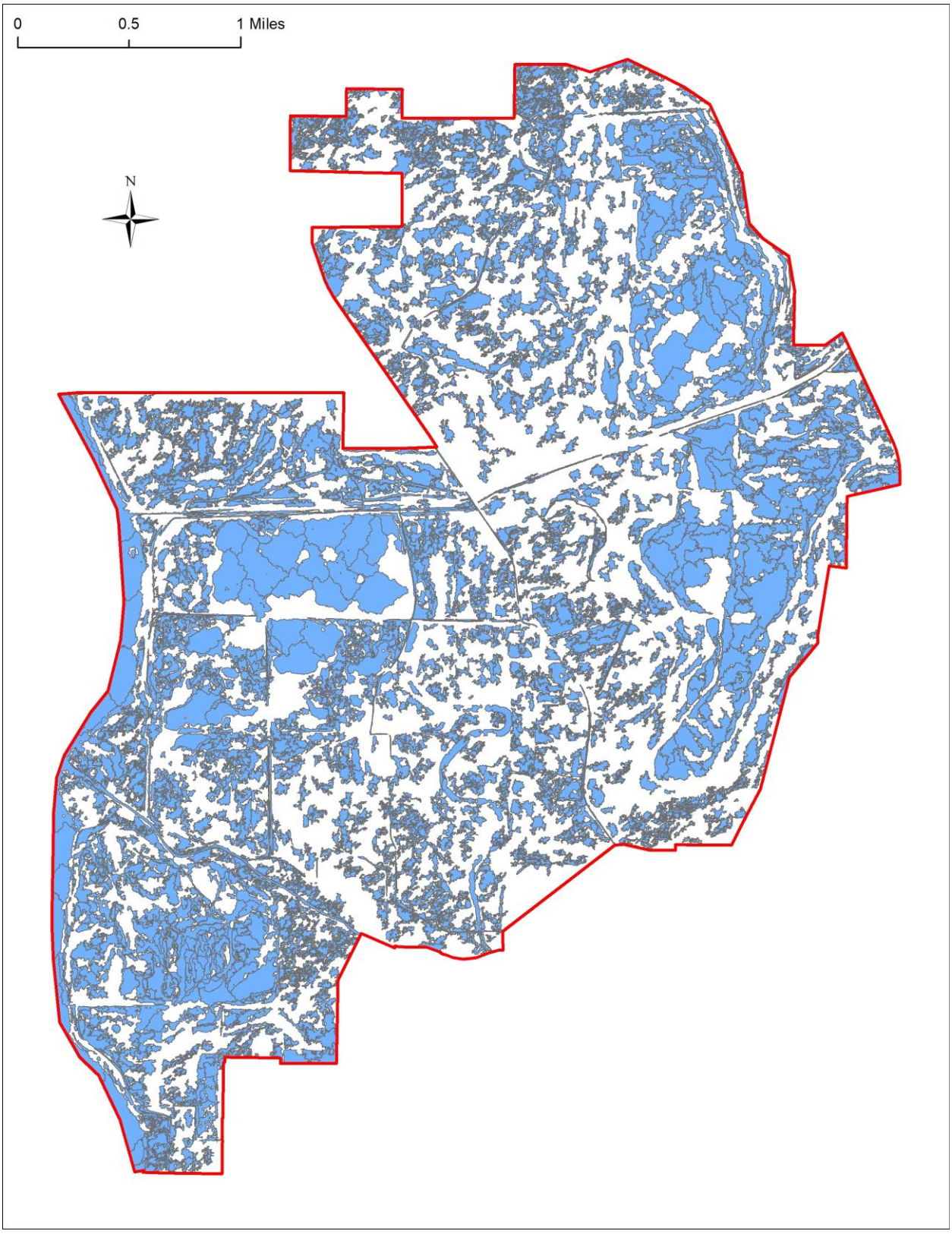



Figure 13. Identified training objects used for Emeralda Marsh area classification.

### 3.4 Accuracy assessment

The performance of classifiers ANN, kNN, SVM, RF based on 10-fold cross validation is displayed in Table 8. All achieved an overall accuracy (OA) above 80%, and RF achieved an accuracy above 85%. Results from McNemar tests are displayed in Table 9. kNN was dropped due to it having the lowest accuracy. RF was significantly better than ANN, kNN, and SVM, and ANN and SVM generated similar accuracy. Ensemble analysis of ANN, RF, and SVM achieved an OA of 84.4% with Kappa value of 0.82. Comparison with an inclusion of lidar DEM in the classification is also displayed in Table 8. An addition of lidar DEM increased the OA around 5-8% and thus is important in the mapping procedure. The error matrix is large due to the classification of 16 communities in this area and has been submitted separately as an Excel file.

Table 8. Performance of each machine learning classifier and ensemble analysis with/without lidar DEM; EA is ensemble analysis of ANN, SVM, and RF.

| | ANN | kNN | SVM | RF | EA |
|---|---|---|---|---|---|
| Overall Accuracy (%) | 81.6/78.0 | 80.8/76.7 | 82.0/77.9 | 85.3/82.5 | 84.4 |
| Kappa Value | 0.79/0.75 | 0.78/0.73 | 0.79/0.74 | 0.83/0.80 | 0.82 |

Table 9. McNemar test between different classifiers (* significance with 95% confidence when z-Score is greater than 1.96). ANN and kNN are not significantly different. Thus the ensemble analysis was conducted using only ANN, SVM, and RF.

| | ANN/kNN | ANN/SVM | ANN/RF | SVM/RF |
|---|---|---|---|---|
| z-Score | 1.2 | 0.7 | 6.4* | 6.2* |

### 3.5 Vegetation community mapping

Classification maps from machine learning algorithms, and ensemble analysis are displayed in Figure 14. In general, all classifiers produced a consistent spatial pattern with the existing map, but classification maps provide a more realistic/natural edge of each patch and presence of small patches within large communities. For most regions, three classifiers generated consistent classification, as shown in green in the uncertainty map in Figure 14 (G). Red regions show a "warning sign" where classification from three classifiers are completely inconsistent.

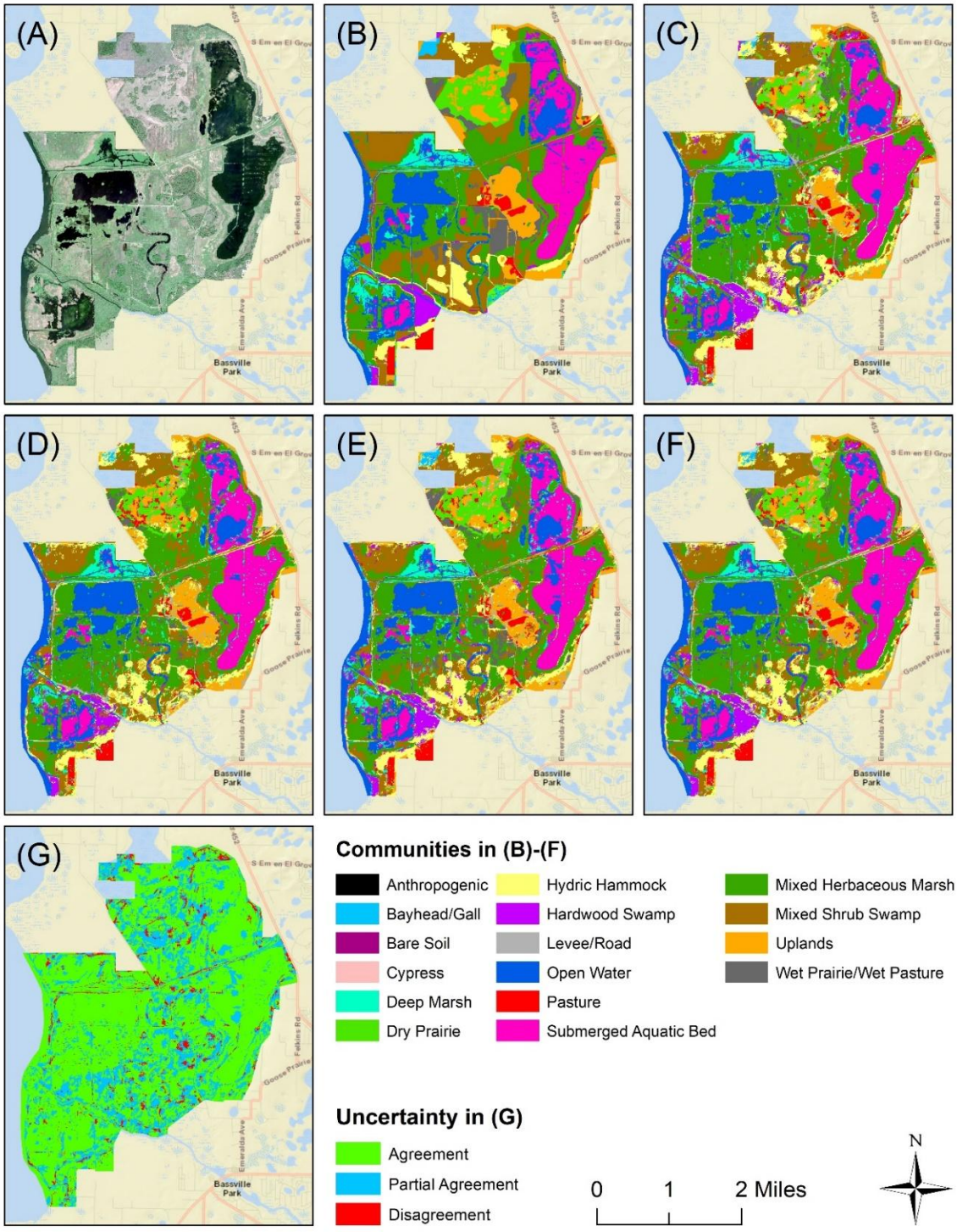



Figure 14. (A) WV-2 pansharpened imagery; (B) reference map from aerial photography; Object-based classifications from (C) ANN, (D) SVM, (E) RF, and (F) Ensemble analysis. (G) Uncertainty from ANN, SVM, and RF.

### 3.6 Analysis

For this area, classification of BG, DP, SS, and WP were confounded. Class BG is Bayhead/gall, class DP and class WP are dry/wet prairies and pastures, and class SS is swamp. They were difficult to differentiate on the imagery. Class BG(Bayhead/gall) was confused with class HH (Hydric Hammock); class DP (Dry Prairie) was confused with class PA (Pasture); class SS (Shrub Swamp) was confused with class HM (Mixed Herbaceous Marsh); and class WP (Wet Prairie) was confused with class HM. The aerial coverage of communities AN (Anthropogenic), BS (Bare Soil), and CY (Cypress) were very small and isolated and generated a limited amount of reference samples. These small communities generated very low accuracy and complicated the results produced by the machine learning classifiers. Thus, these three were dropped in the machine learning classifiers (Table 8 and Table 9) and the classification maps (Figure 14 C-G) which contained 13 out of the original 16 total classes.

## 4. Results for Buck Lake Area

This section provides results for Buck Lake area using segmentation in eCognition software.

### 4.1 Communities to be mapped

For this area, CA was mapped as OW; WL was mapped as DM: CTSG was mapped as HM; MS and TS were mapped as SS; and MH and OH were mapped as UH. In total, there were 20 communities to be mapped making it a challenge due to the large number of communities.

### 4.2 Segmentation

A total of 25,362 image segments was generated from the multiresolution segmentation algorithm in eCognition. This size is manageable in ArcGIS and the segmentation results are reasonable based on visual check. Figure 15 shows the segmentation results.

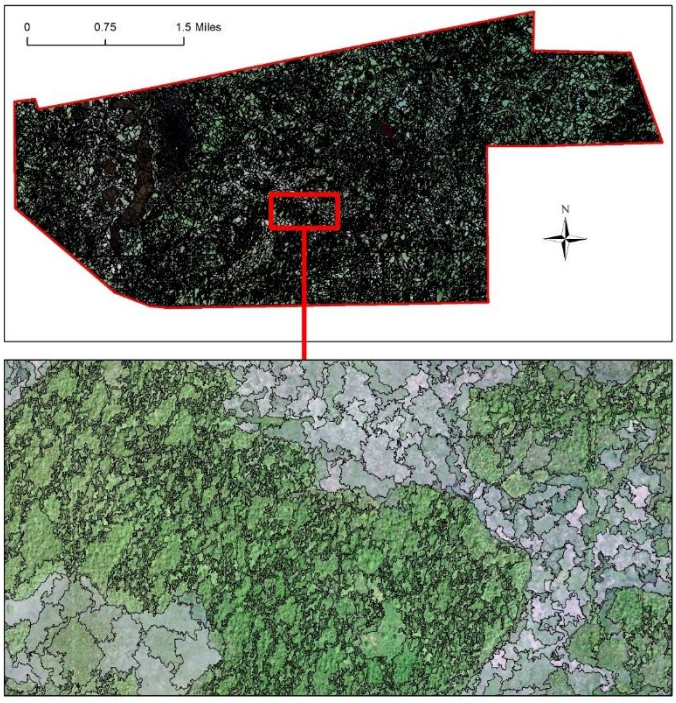



Figure 15. Segmentation results for Buck Lake area.

### 4.3 Reference sample selection

We collected a total of 2266 reference samples to be used as training and testing samples for classification of the 20 communities, as shown in Figure 16.

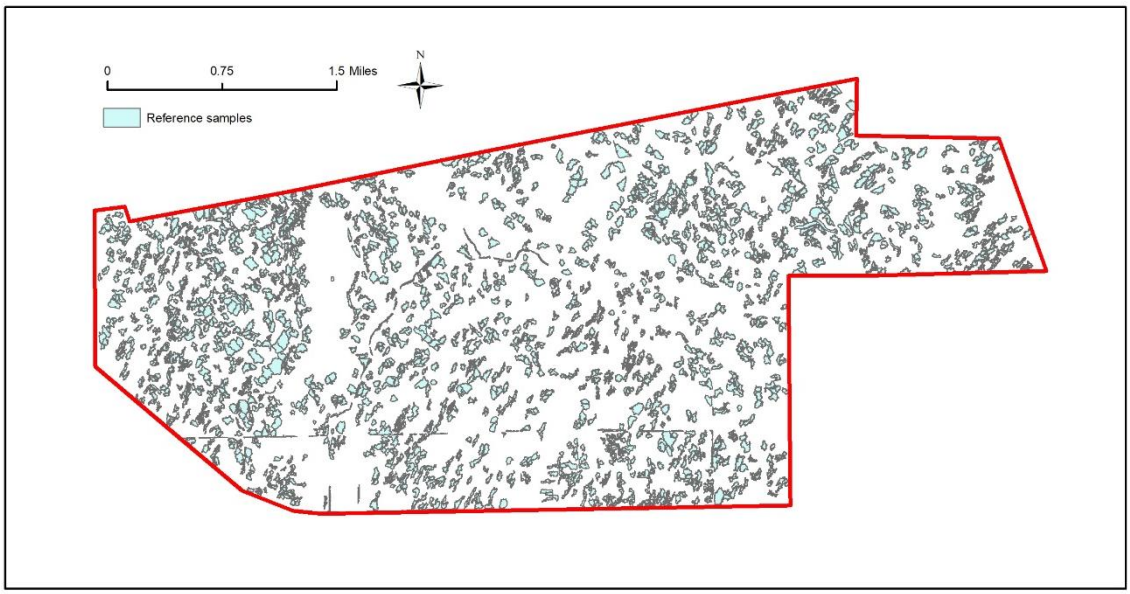



Figure 16. Selected reference samples for Buck Lake area.

### 4.4 Accuracy assessment

The performance of classifiers ANN, kNN, SVM, RF based on 10-fold cross validation is displayed in Table 10. All achieved an overall accuracy (OA) above 65%, and SVM achieved an accuracy above 70%. Results from McNemar tests are displayed in Table 11. kNN was dropped due to it having the lowest accuracy. SVM was significantly better than ANN and RF. RF and ANN generated similar accuracies. Ensemble analysis of ANN, RF, and SVM achieved an OA of 71.2% with Kappa value of 0.68. Comparison with an inclusion of lidar DEM in the classification is also displayed in Table 10. The addition of lidar DEM increased the OA around 5-8% and thus is important in the mapping procedure. The error matrix is large due to classification of 20 communities in this area and has been submitted separately as an Excel file.

Table 10. Performance of each machine learning classifier and ensemble analysis with/without lidar DEM; EA is ensemble analysis of ANN, SVM, and RF. kNN was not used due to the difficulty using four classifiers versus three.

| | ANN | kNN | SVM | RF | EA |
|---|---|---|---|---|---|
| Overall Accuracy (%) | 68.0/64.2 | 65.0/57.8 | 71.3/63.0 | 68.0/62.3 | 71.2 |
| Kappa Value | 0.65/0.60 | 0.61/0.53 | 0.68/0.58 | 0.65/0.58 | 0.68 |

Table 11. McNemar test between different classifiers (* significance with 95% confidence when z-Score is greater than 1.96). Ensemble analysis was done using ANN, SVM, and RF.

| | ANN/kNN | ANN/SVM | ANN/RF | SVM/RF |
|---|---|---|---|---|
| z-Score | 3.1* | 4.4* | 0.1 | 4.0* |

### 4.5 Vegetation community mapping

Classification maps from machine learning algorithms, and ensemble analysis are displayed in Figure 17. In general, all classifiers produced a consistent spatial pattern with the existing map, but classification maps provide a more realistic/natural edge of each patch and presence of small patches within large communities. For most regions, three classifiers generated consistent classification, as shown in green in the uncertainty map in Figure 17(G). Red regions show a "warning sign" where classification from three classifiers are completely inconsistent.

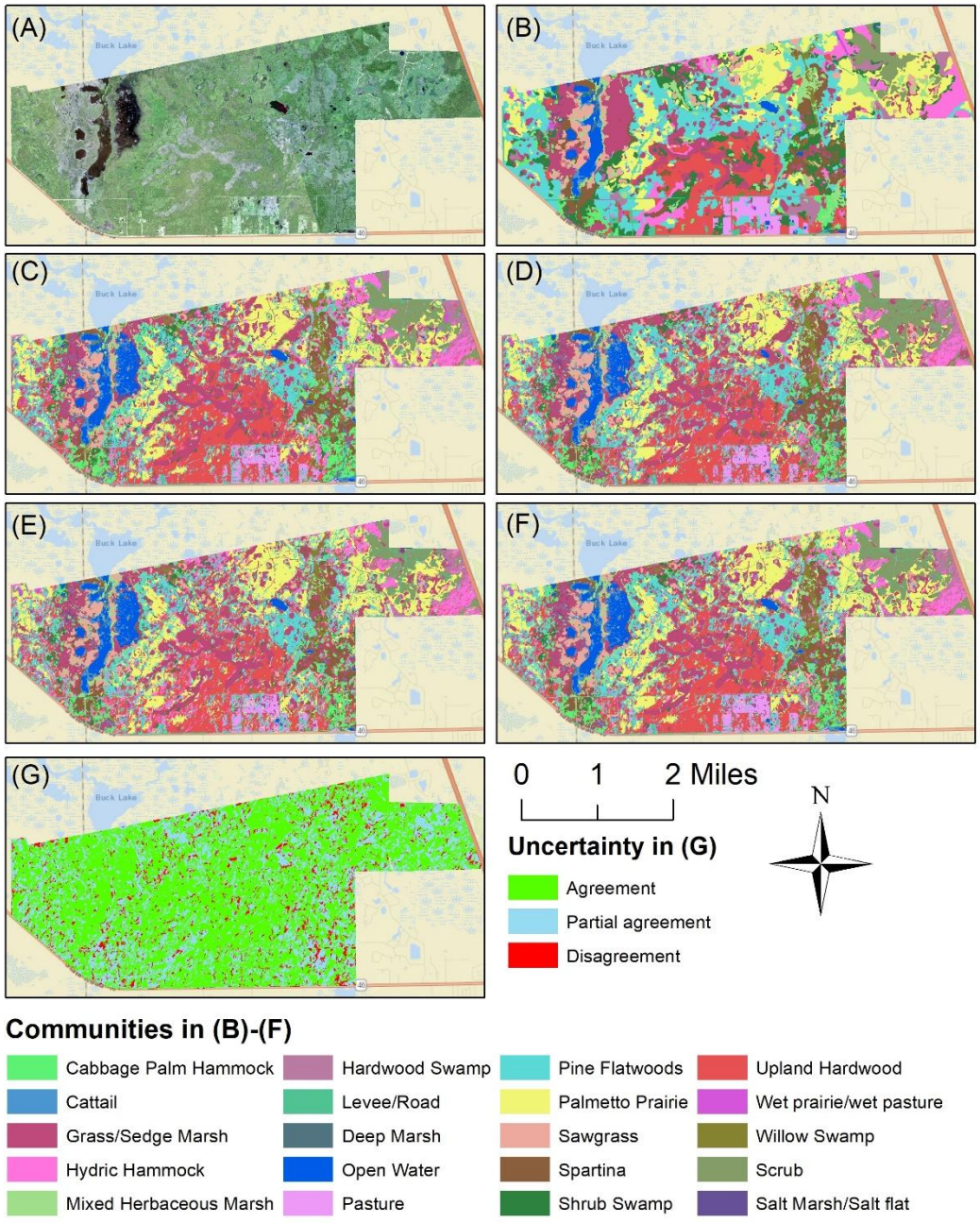


Figure 17. (A) WorldView-2 pansharpened imagery; (B) reference map from aerial photography; Object-based classifications from (C) ANN, (D) SVM, (E) RF, and (F) Ensemble analysis. (G) Uncertainty from ANN, SVM, and RF.

### 4.6 Analysis

A lower accuracy was produced for this area. Further analysis shows that classes CP (Cabbage Palm Hammock), PF (Pine Flatwoods), UH (Upland Hardwood), and SS (Shrub Swamp) were hard to differentiate for this area. All are forests and difficult to differentiate on imagery itself. In addition, class HM (Mixed Herbaceous Marsh) was confused with GS (Grass/Sedge Marsh); class SS (Shrub Swamp) was confused with PF (Pine Flatwoods); HM and WP (Wet Prairie) were confused with one another; and WS (Willow Swamp) and HM were confused with one another.

One potential factor for the lower accuracy for this area maybe the incorrect labeling of training samples caused by the existing vegetation map, which is not consistent with the 2014 LCLU map. For example, the UH (Upland Hardwood) areas shown on the existing map have multiple LCLU community labels including pine flatwoods (4110) and Hydric Pine Flatwoods (6250), which are class PF (pine flatwoods); and Shrub and Brushland (3200), which is class SS (Shrub Swamp). Based on these potential differences, it was not surprising that classes such as SS, PF, UH, and CP are confused during classification, resulting in mixed labeling for these classes. It appears that the selection of training samples for these classes, based on potentially wrong identification of communities on the original map, led to a lower classification accuracy for these classes over this area. Figure 18 shows an example of this labeling confusion between the District vegetation map and LCLU map.

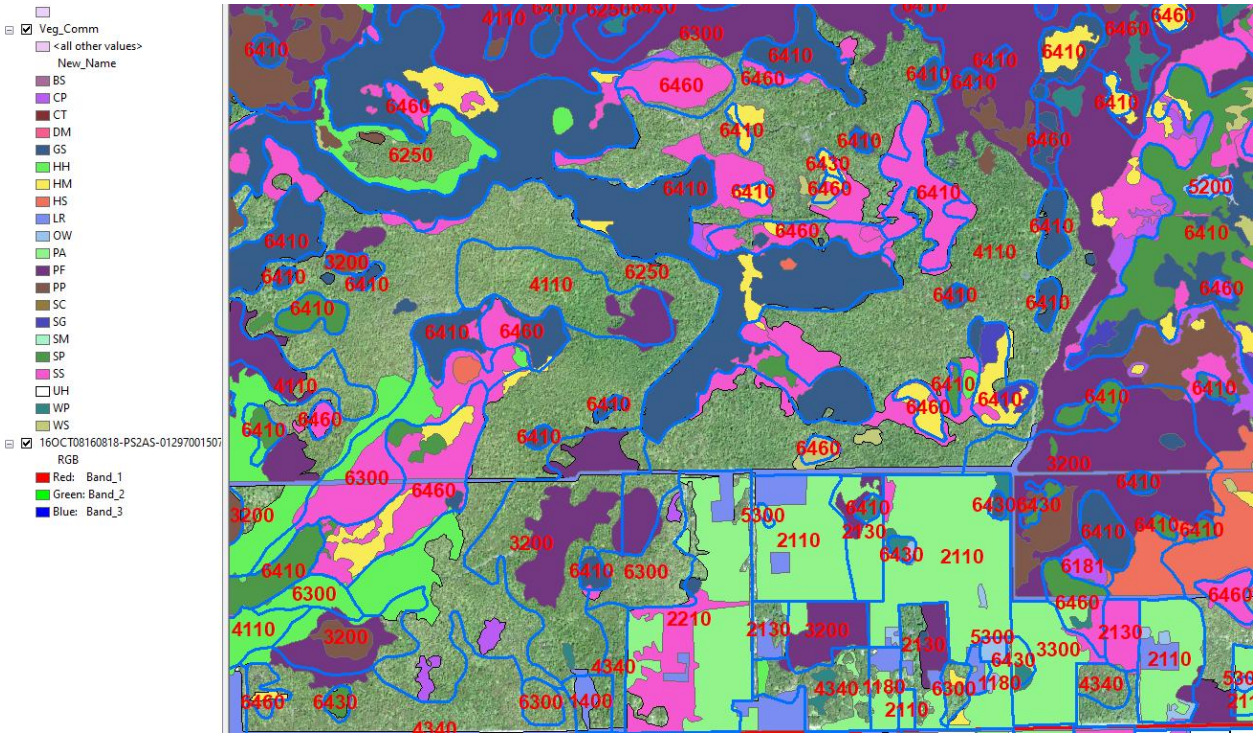



Figure 18. The Upland Hardwood communities on the District map (areas of no color with imagery showing through) have been labelled as pine flatwoods (4110) in LCLU map. Are they hardwood or pines?

## 5. Application of WV Multispectral imagery

We compared 0.3-m pansharpened imagery with 1.2-m multispectral imagery for classification for Moccasin Island area. We found that pansharpened imagery could achieved an overall accuracy (OA) of 85% while application of multispectral imagery produced an OA around 50%. Thus analysis of pansharpened imagery is suitable and thus was used for mapping. Testing of other areas were not conducted.

## 6. Identified Issues and Potential Solutions

### 6.1 Image segmentation

The multiresolution segmentation algorithm integrated in eCognition is commonly used for object-based remote sensing image classification, but unfortunately the District prefers to use open-source software or software it already owns license for, such as ArcGIS and ENVI, as stated in their contract with FAU. Consequently, an alternative to the multiresolution segmentation in eCognition must be found. ArcGIS utilizes a Mean Shift image segmentation algorithm and ENVI utilizes an edge-based watershed image segmentation. However, neither could produce a segmentation result comparable to eCognition. Figure 19 shows the segmentation results for part of the Buck Lake area using ArcGIS Mean Shift and ENVI edge-based segmentation algorithm. For ArcGIS Pro, this was produced by running the segmentation and then converting the segmented imagery into a polygon vector layer.

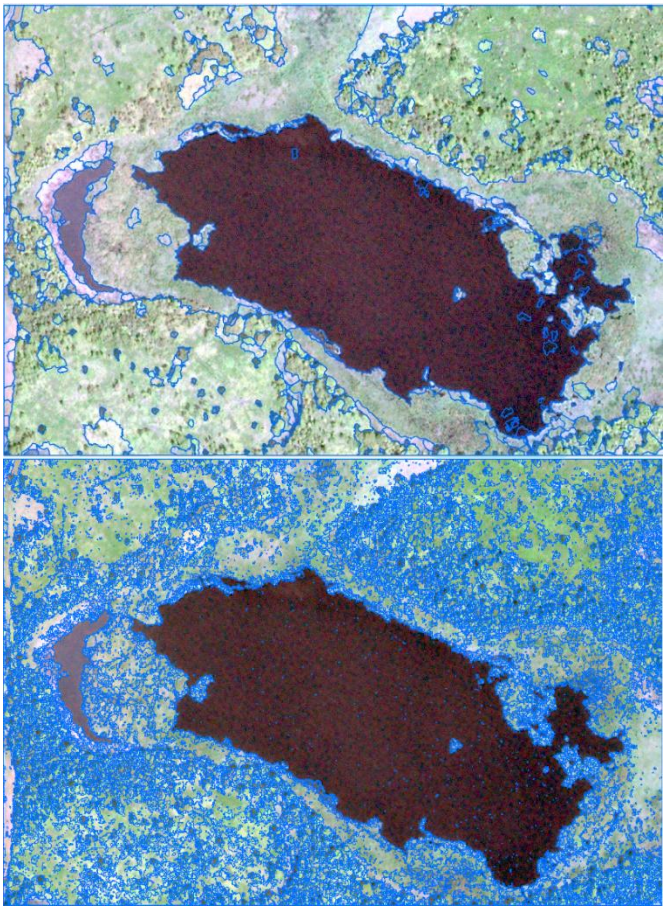


Figure 19. Segmentation results using ArcGIS Mean Shift (top), and ENVI edge-based (bottom) segmentation algorithm.

The open source remote sensing image processing package OTB (https://www.orfeo-toolbox.org/) has integrated a segmentation algorithm Large Scale Mean Shift which is described as the same algorithm used in ArcGIS but can split the large size imagery into small tiles and automatically stitch them after segmentation. This algorithm produced a similar result as the multiresolution segmentation in eCognition (an example is shown in Figure 20). A subsequent meeting with ESRI and District staff on March 23, 2021was not helpful as ESRI technical staff could not explain the difference in the ArcGIS and OTB results and could not say when ESRI would integrate the multiresolution segmentation algorithm into ArcGIS.

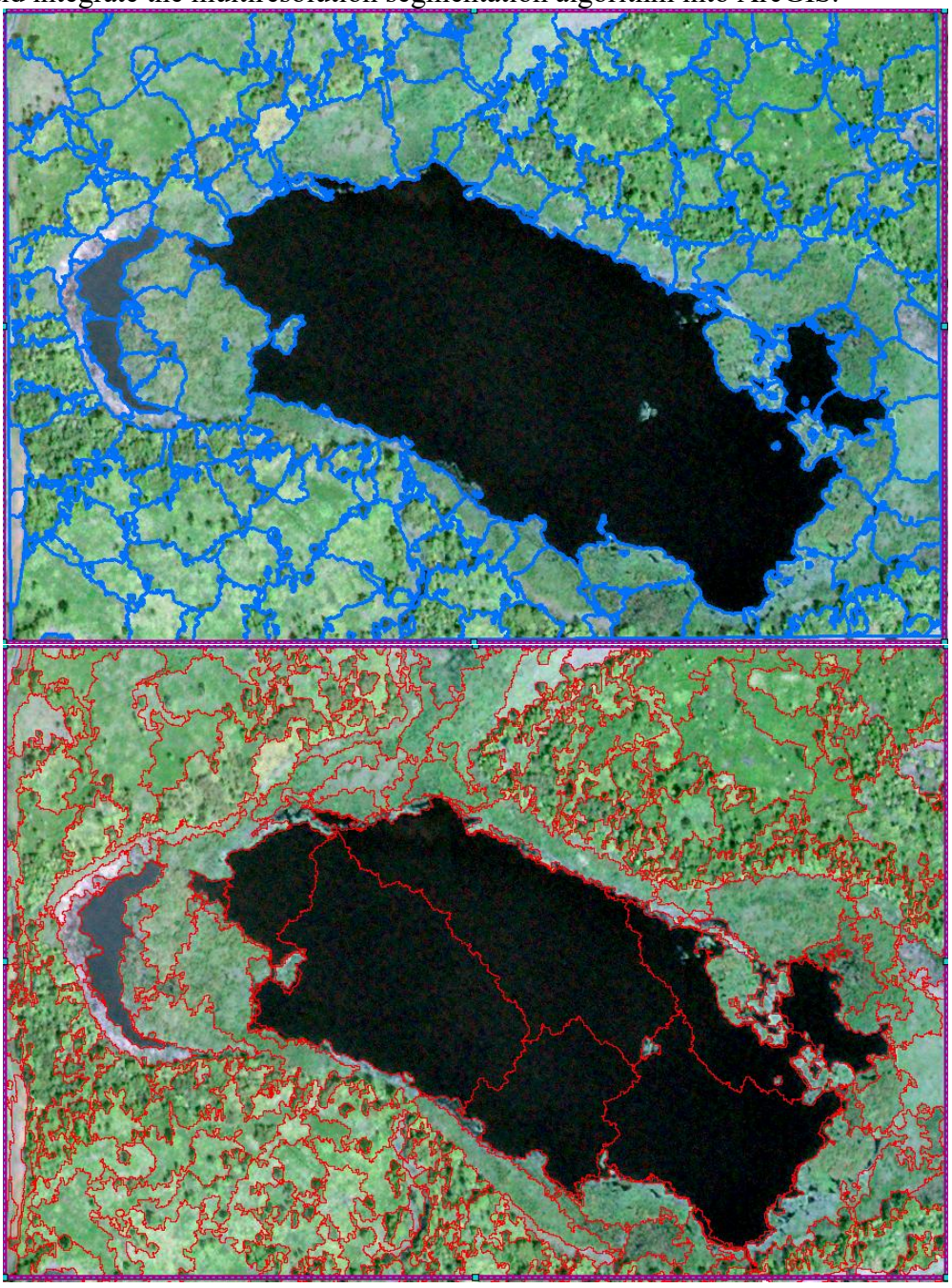


Figure 20. Segmentation from OTB Large Scale Mean Shift (top), and eCognition (bottom) from eCognition.

OTB Large Scale Mean Shift segmentation can be implemented using QGIS (https://qgis.org/en/site/), an open source GIS software. We have developed a tutorial using Large Scale Mean Shift segmentation of OTB in QGIS (See Appendix 2). However, the segmentation is very slow in QGIS, which is mainly caused by the procedure called mean shift imagery smoothing. The Large Scale Mean Shift segmentation in OTB goes through four procedures and the first step is the mean shift smoothing. However, we could not find any mention of this procedure in a literature survey of this algorithm. This procedure may be the reason for the difference between ArcGIS Mean Shift segmentation and OTB Large Scale Mean Shift segmentation. The OTB Large Scale Mean Shift segmentation can also run in Python, but must be in the Python 3.5 environment and with the correct environmental variables set up on PC (Appendix 3). This may speed up the segmentation procedure. If so, we will figure out how to parallel process this procedure in Python and further speed up the procedure. We also tested other open source segmentation such as SAGA's watershed segmentation. So far the best solution is the application of OTB Large Scale Mean Shift segmentation.

### 6.2 Labeling confusion and other issues

In the mapping procedure, training/testing sample selection and labeling are the most important steps. Incorrect/confusing samples will lead to a lower accuracy. We found pine flatwoods, pine plantation, upland hardwood, and cabbage palm hammock are labeled alternatively in LCLU maps for Buck Lake, as shown in Figure 18. In addition, buildings in the Buck Lake area were mapped as levees/roads in the original map based on aerial photography. LCLU maps may help to more precisely classify the Uplands mapped in UORB using aerial photography. Shadows do not seem to be a problem as we have not seen big shadows on the satellite imagery and the OBIA techniques can help reduce small shadows in the classification.

## 7. Files for Quarter 1, 2021

Products in 2021 quarter 1 are summarized in Table 12, and are distributed to District via Google Drive.

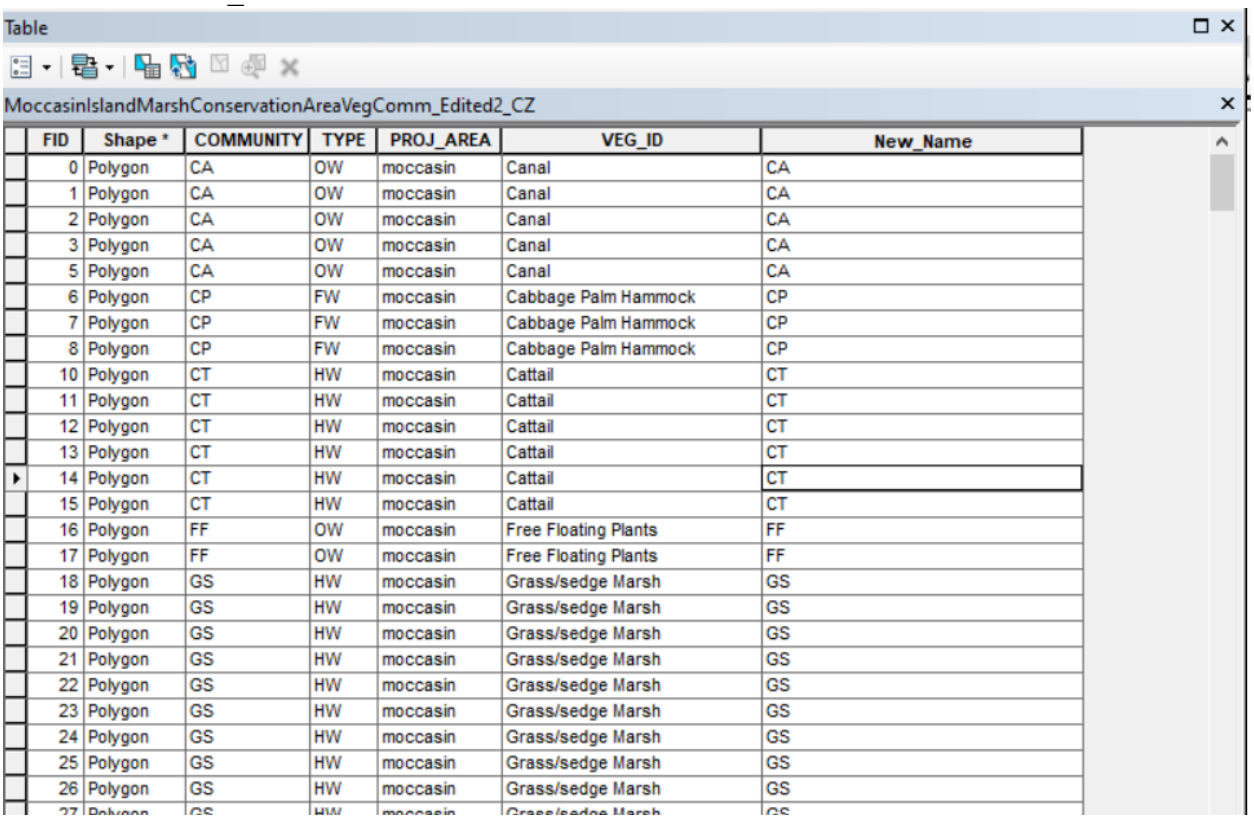
Table 12. Products in Quarter 1 provided by FAU.

| Areas | Products |
|---|---|
| Moccasin Island | ArcGIS shape files of ANN, SVM, RF, Ensemble Analysis, Uncertainty; Excel file of error matrix |
| Emeralda Marsh | |
| Buck Lake | |

## Appendix 1. Editing Community Names

This tutorial is used to revise the old community name into the required new community name in the SOW Appendix 2. The tutorial is based on Moccasin Island Vegetation Community Map (MoccasinIslandMarshConservationAreaVegComm.shp). Use the PlantCommunityCrosswalk.xlsx file and the Appendix 2 in SOW to revise the name

1. Open the vegetation shape file in ArcGIS, and clip it to your AOI
2. Add a new text attribute as New_Name, as shown below, and copy the old COMMUNITY attribute to the New_Name attribute; you can start editing/revising the name on the New_Name attribute.



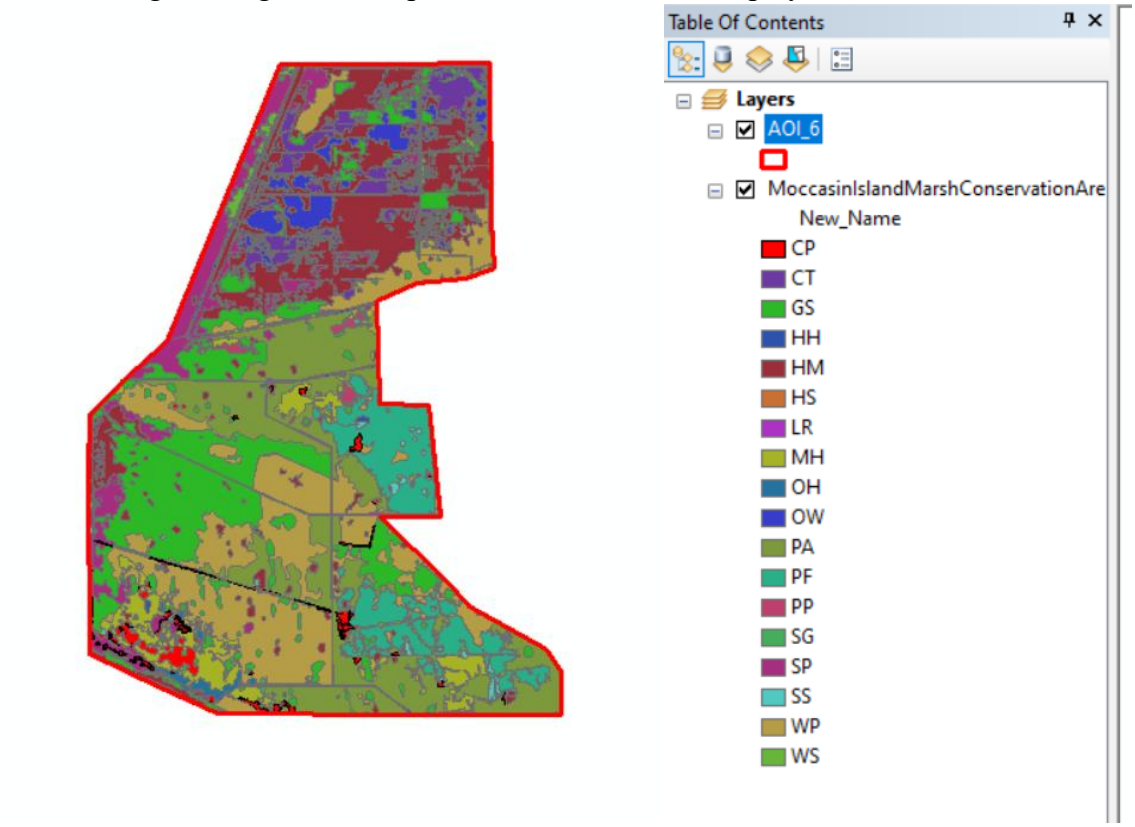
Table

MoccasinIslandMarshConservationAreaVegComm_Edited2_CZ

| FID | Shape * | COMMUNITY | TYPE | PROJ_AREA | VEG_ID | New_Name |
|---|---|---|---|---|---|---|
| 0 | Polygon | CA | OW | moccasin | Canal | CA |
| 1 | Polygon | CA | OW | moccasin | Canal | CA |
| 2 | Polygon | CA | OW | moccasin | Canal | CA |
| 3 | Polygon | CA | OW | moccasin | Canal | CA |
| 5 | Polygon | CA | OW | moccasin | Canal | CA |
| 6 | Polygon | CP | FW | moccasin | Cabbage Palm Hammock | CP |
| 7 | Polygon | CP | FW | moccasin | Cabbage Palm Hammock | CP |
| 8 | Polygon | CP | FW | moccasin | Cabbage Palm Hammock | CP |
| 10 | Polygon | CT | HW | moccasin | Cattail | CT |
| 11 | Polygon | CT | HW | moccasin | Cattail | CT |
| 12 | Polygon | CT | HW | moccasin | Cattail | CT |
| 13 | Polygon | CT | HW | moccasin | Cattail | CT |
| 14 | Polygon | CT | HW | moccasin | Cattail | CT |
| 15 | Polygon | CT | HW | moccasin | Cattail | CT |
| 16 | Polygon | FF | OW | moccasin | Free Floating Plants | FF |
| 17 | Polygon | FF | OW | moccasin | Free Floating Plants | FF |
| 18 | Polygon | GS | HW | moccasin | Grass/sedge Marsh | GS |
| 19 | Polygon | GS | HW | moccasin | Grass/sedge Marsh | GS |
| 20 | Polygon | GS | HW | moccasin | Grass/sedge Marsh | GS |
| 21 | Polygon | GS | HW | moccasin | Grass/sedge Marsh | GS |
| 22 | Polygon | GS | HW | moccasin | Grass/sedge Marsh | GS |
| 23 | Polygon | GS | HW | moccasin | Grass/sedge Marsh | GS |
| 24 | Polygon | GS | HW | moccasin | Grass/sedge Marsh | GS |
| 25 | Polygon | GS | HW | moccasin | Grass/sedge Marsh | GS |
| 26 | Polygon | GS | HW | moccasin | Grass/sedge Marsh | GS |

3. Look at the crosswalk excel file and see which name should be changed. For example, I am working on Moccasin Island, it is in the USJRB area, you can see areas labeled as open water, canals, ditches or free floating plants are all now as Water, use OW label as suggested in Appendix 2 in SOW. The original COMMUNITY attribute listed as CA, FF, OW, should be changed into OW, the TYPE attribute can help to select them. We will eventually need to map COMMUNITY, rather than type. The aim is to only use names appearing in Appendix 2- Plant Community in the New_Name attribute. After checking the Moccasin Island Vegetation shape file, only CC and FF need to change to OW; MS and TS need to change to SS, and all the other COMMUNITY designations do not need to change.
4. Once you finish the editing, double check that all communities in the New_Name can be found in Appendix 2 of the SOW. Any communities you cannot find in Appendix 2 should be renamed.

5. We will classify the satellite imagery based on Appendix 2 names, which I edited and listed in the crosswalk excel file.
6. After editing, the vegetation map with the new name is displayed below




We will use the new name attribute and this vegetation map to select training samples for the satellite imagery classification. Make sure the names shown in the legend can all be found in Appendix 2 of the SOW, which I also list in the Excel file.

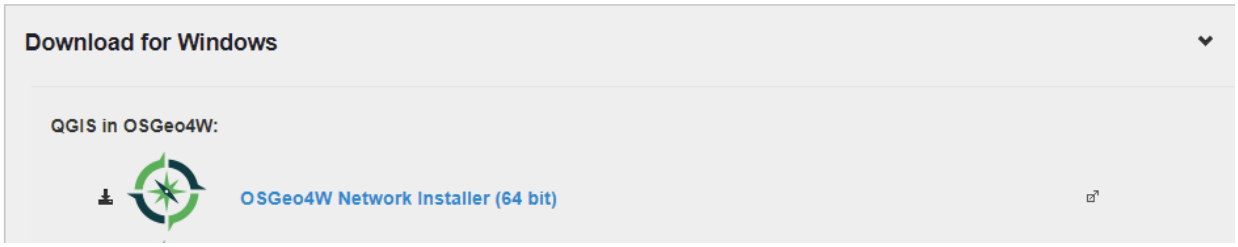
## Appendix 2. Using OTB/QGIS Open Source for Image Segmentation

### Background

Ideally, the multiresolution segmentation algorithm in Trimble eCognition Developer 10.0 software package should be used for object-based vegetation mapping because it can delineate better boundaries for each vegetation patch than other segmentation algorithms such as the edge-based segmentation algorithms in ENVI Feature Extraction module. However, the District prefers to use open-source software, ArcGIS or ENVI, thus alternatives to the multiresolution segmentation algorithm utilized by eCognition must be found. The following is the tutorial for using the open source OTB Large Scale Mean Shift segmentation algorithm as an alternative for the eCognition multiresolution segmentation procedure.

**1. Download QGIS** from https://qgis.org/en/site/forusers/download.html

Make sure to download the **3.16 or higher version of QGIS** to be compatible to OTB.




Screenshot of downloading QGIS for Win 10, 64 bit

**2. Install QGIS**; the package folder will be C:\OSGeo4W64

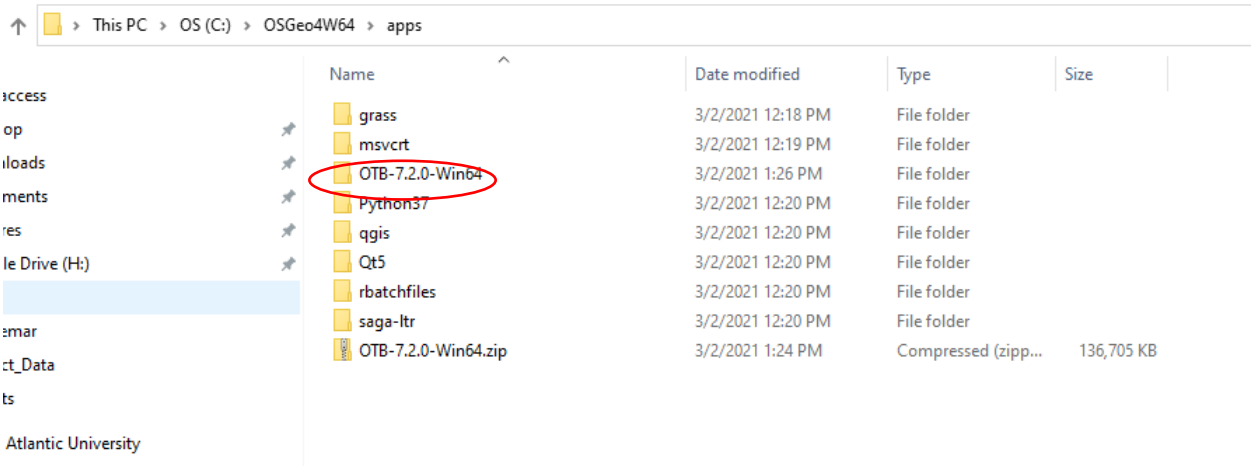
**3. Download Orfeo ToolBox** from https://www.orfeo-toolbox.org/download/

The current version is OTB 7.2.0 for Windows 64 bit.


Screenshot of downloading OTB for win 10 64 bit

**4. Extract Orfeo Toolbox folder to C:\OSGeo4W64\apps**

After downloading the OTB, extract OTB to C:\OSGeo4W64\apps, as shown below.




Extracting OTB-7.2.0-Win64 zip package to “C:\OSGeo4W64\apps” folder.

## 5. Configuring OTB in QGIS

Based on https://docs.qgis.org/3.16/en/docs/user_manual/processing/3rdParty.html

OTB applications are fully supported within the QGIS Processing framework.

OTB (Orfeo ToolBox) is an image processing library for remote sensing data. It also provides applications that provide image processing functionalities. The list of applications and their documentation are available in OTB CookBook

**Note**

Note that OTB is not distributed with QGIS and needs to be installed separately. Binary packages for OTB can be found on the download page.

To configure QGIS processing to find the OTB library:

1. Open the processing settings: **Settings ► Options ► Processing** (left panel)*
2. You can see OTB under “Providers”:
    1. Expand the **OTB** tab
    2. Tick the **Activate** option
    3. Set the **OTB folder**. This is the location of your OTB installation.
    4. Set the **OTB application folder**. This is the location of your OTB applications ( <PATH_TO_OTB_INSTALLATION>/lib/otb/applications )
    5. Click “ok” to save the settings and close the dialog.

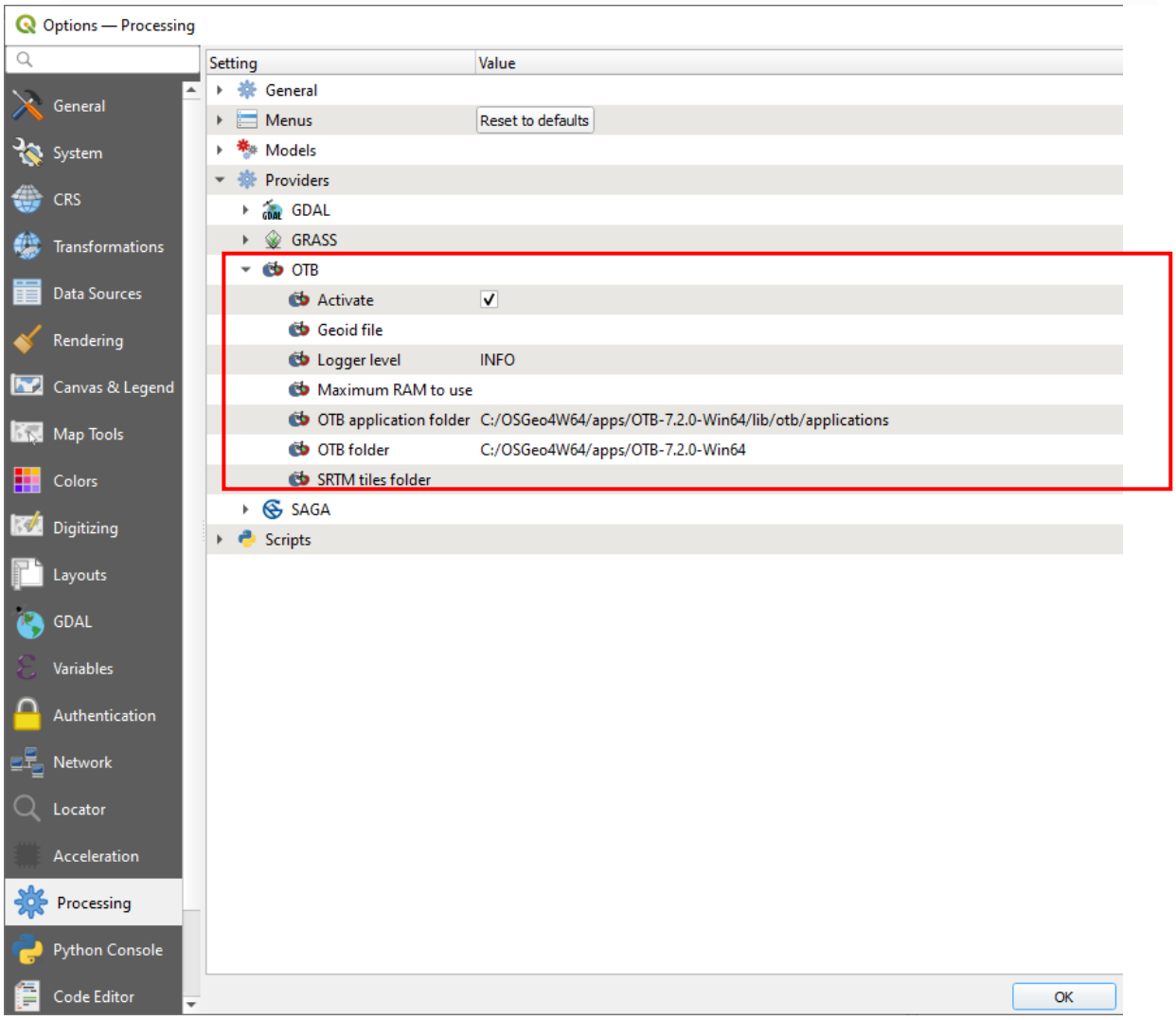
If settings are correct, OTB algorithms will be available in the **Processing Toolbox**.

- **Activate**: This is a checkbox to activate or deactivate the OTB provider. An invalid OTB setting will uncheck this when saved.
- **OTB folder**: This is the directory where OTB is available.
- **OTB application folder**: This is the location(s) of OTB applications.
  Multiple paths are allowed.
- **Logger level** (optional): Level of logger to use by OTB applications.

The level of logging controls the amount of detail printed during algorithm execution. Possible values for logger level are `INFO`, `WARNING`, `CRITICAL`, `DEBUG`. This value is `INFO` by default. This is an advanced user configuration.

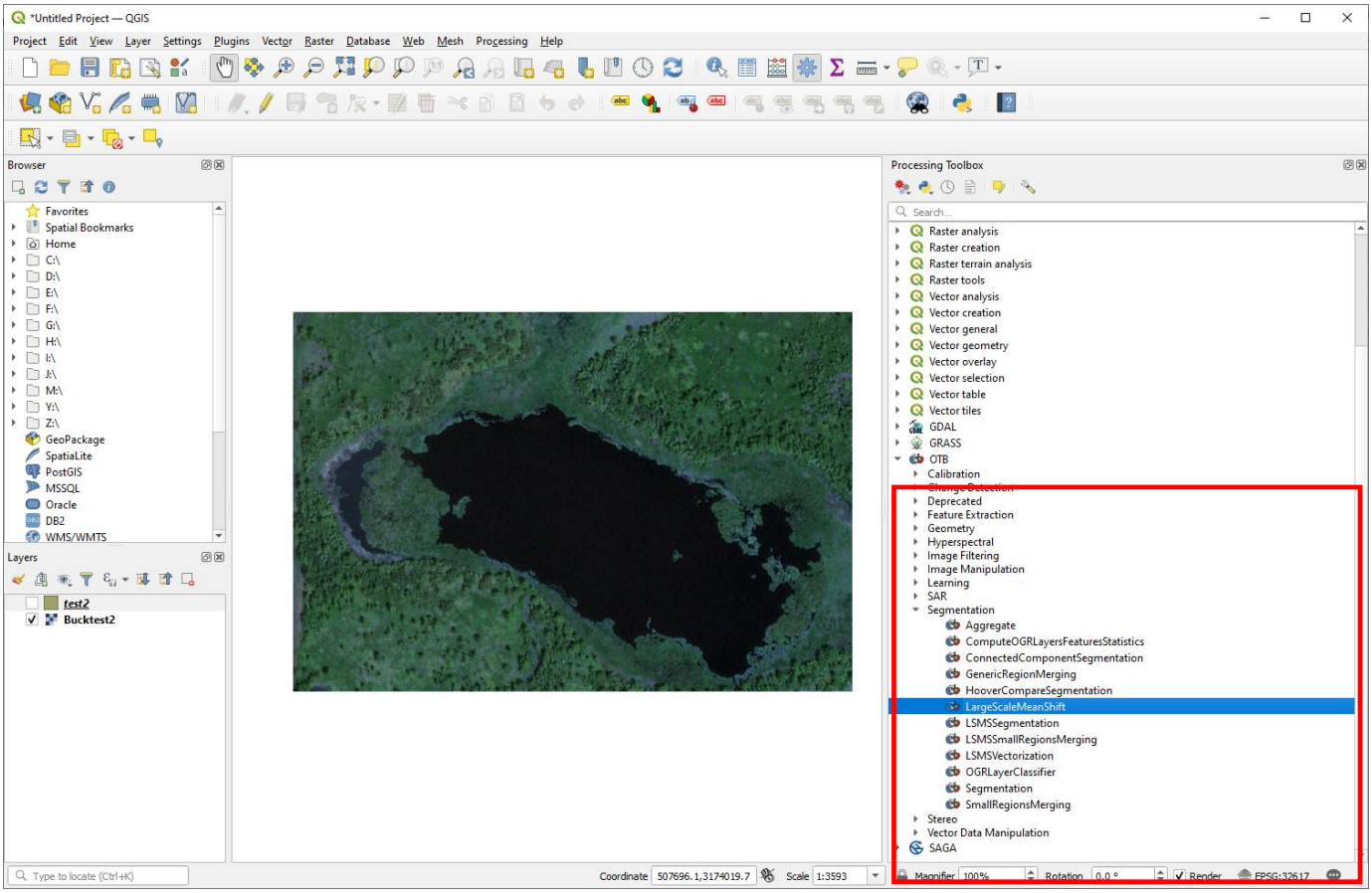
- **Maximum RAM to use** (optional): by default, OTB applications use all available system RAM. You can, however, instruct OTB to use a specific amount of RAM (in MB) using this option. A value of 256 is ignored by the OTB processing provider. This is an advanced user configuration.
- **Geoid file** (optional): Path to the geoid file.
  This option sets the value of the elev.dem.geoid and elev.geoid parameters in OTB applications. Setting this value globally enables users to share it across multiple processing algorithms. Empty by default.
- **SRTM tiles folder** (optional): Directory where SRTM tiles are available.
  SRTM data can be stored locally to avoid downloading of files during processing. This option sets the value of elev.dem.path and elev.dem parameters in OTB applications. Setting this value globally enables users to share it across multiple processing algorithms. Empty by default.

All OTB versions (from OTB 6.6.1) are compatible with the latest QGIS version.




Screenshot of settings for OTB in QGIS.

After setting, OTB tools will appear in QGIS, Processing->Toolbox, as shown below.

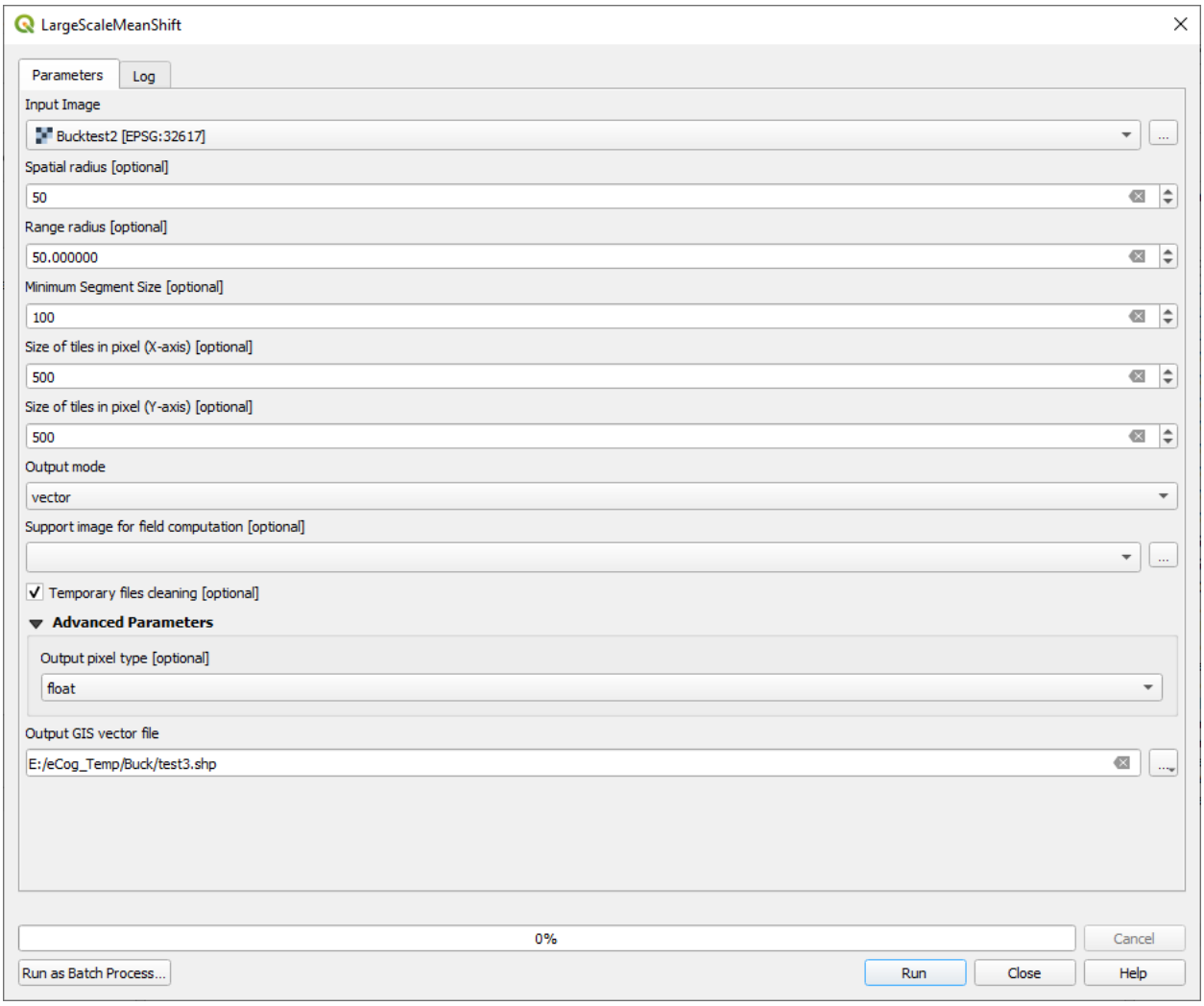



OTB tools as they appear in QGIS Processing->Toolbox.

## 6. Running LargeScaleMeanShift segmentation

In QGIS, Processing Toolbox->OTB->Segmentation->LargeScaleMeanShift, the interface is shown below. Details of this segmentation algorithm can be found at

https://www.orfeo-toolbox.org/CookBook/recipes/improc.html#segmentation




LargeScaleMeanShift segmentation interface in QGIS->OTB.
Setting spatial radius: 50; Range radius: 50, minimum segment size: 100

## Appendix 3. Using OTB in Python 3.5

### Background

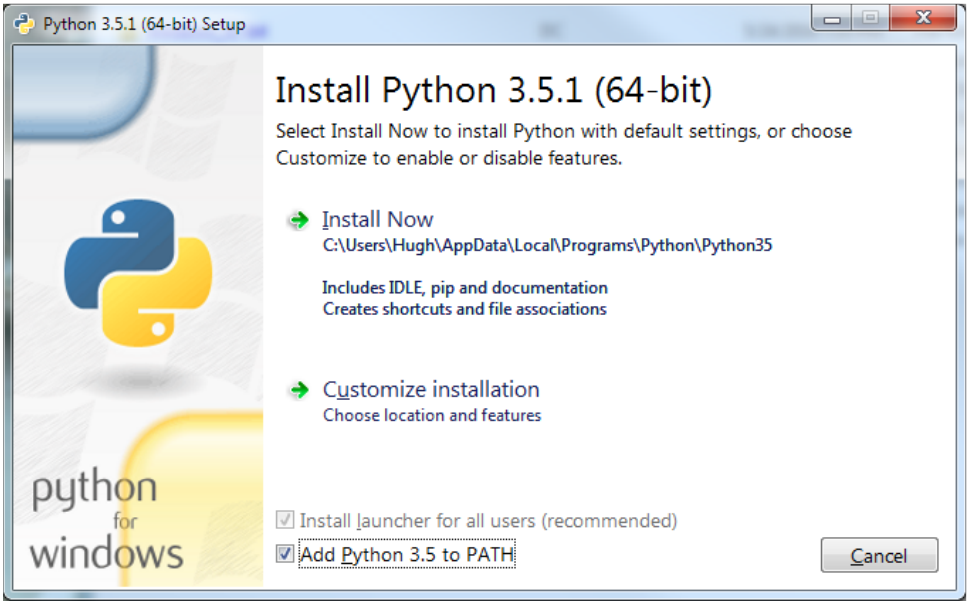
OTB (https://www.orfeo-toolbox.org/), an open source remote sensing data processing package, may run faster in Python platform than QGIS. Current Python 3.9 is not compatible with OTB, thus Python 3.5 needs to be installed first. This tutorial is developed for setting up OTB 7.2 in Python 3.5.

### 1. Download Python 3.5.1

Python 3.5 is available at http://web.cs.wpi.edu/~cs1004/a16/Resources/. I have download it and saved it at "J:\SJWMD\Project_Data\Doc\SoftwareUsingOTB\python-3.5.1-amd64.exe" (also shared in Google Drive in SJWMD\Software) with District. Python 3.5 is end of life and thus is not maintained anymore. It is better to save the Python 3.5 installation file. The Python 3.5 for window 64 bit should be downloaded and installed because almost all Windows PCs sold over the past few years are 64-bit systems.

### 2. Install Python 3.5

*Right-click* on the file **python-3.5.1-amd64.exe** and select *Run as Administrator* to start the installation. You should be greeted by a dialog box resembling the following:




Make sure Add Python 3.5 to Path is checked.

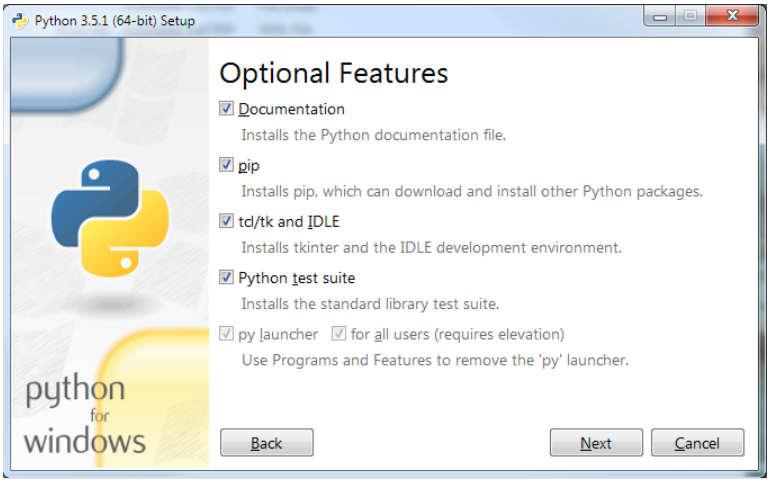
I selected Customize installation and installed Python 3.5 to folder "C:\Users\czhang3\Python". This directory/folder will be used in Environment Variable setup.

Tips:

*If* there is an earlier version of *Python 3.x* installed on your computer (for any value of *x*), you should **Cancel** this installation and remove (i.e., uninstall) the previous version before installing this one.

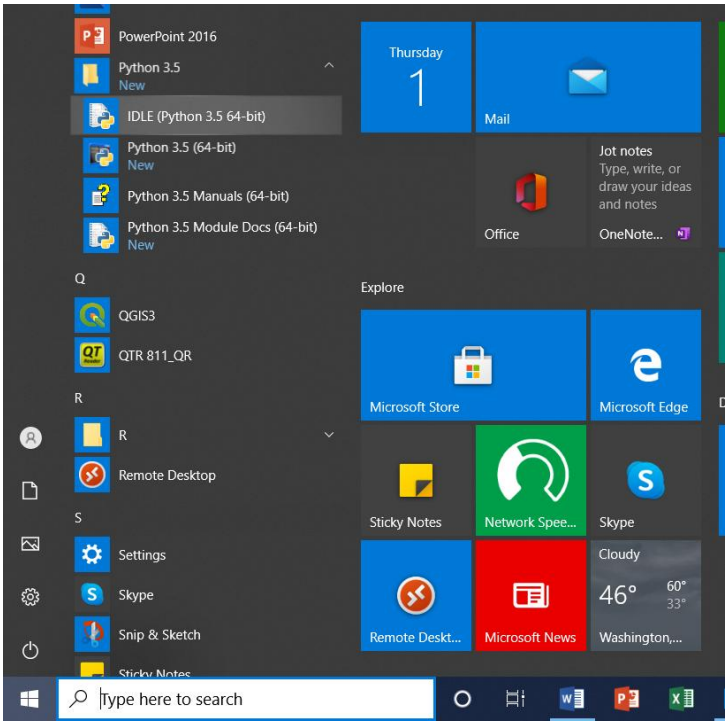
*If* you have a 32-bit installation of Windows (an unlike event nowadays), you should cancel this installation and download and install python-3.5.1.exe for 32-bit system instead.

*If*, however, your computer is used by multiple people with different user names, then click on *Customize Installation* to install *Python* in a more generic place.




Be sure all of the boxes are checked. After installation, Python 3.5 appears in Window 10 Start.

A detailed tutorial for installing Python 3.5 is at http://web.cs.wpi.edu/~cs1004/a16/Resources/ in PDF.




Python 3.5 in Windows 10 Start menu.

3. Testing Python 3.5

Start Python 3.5 (64 bit) from the Start menu. This will start the Python Shell console. Type in:

Print ("hello, Python 3.5")

Hello, Python 3.5 will appear as the output.

**4. Install numpy**

OTB depends on a package called "numpy" which is a commonly used library for Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. To run OTB package, numpy needs to be installed first. To install a package for python, pip is often used in windows command mode by typing command:

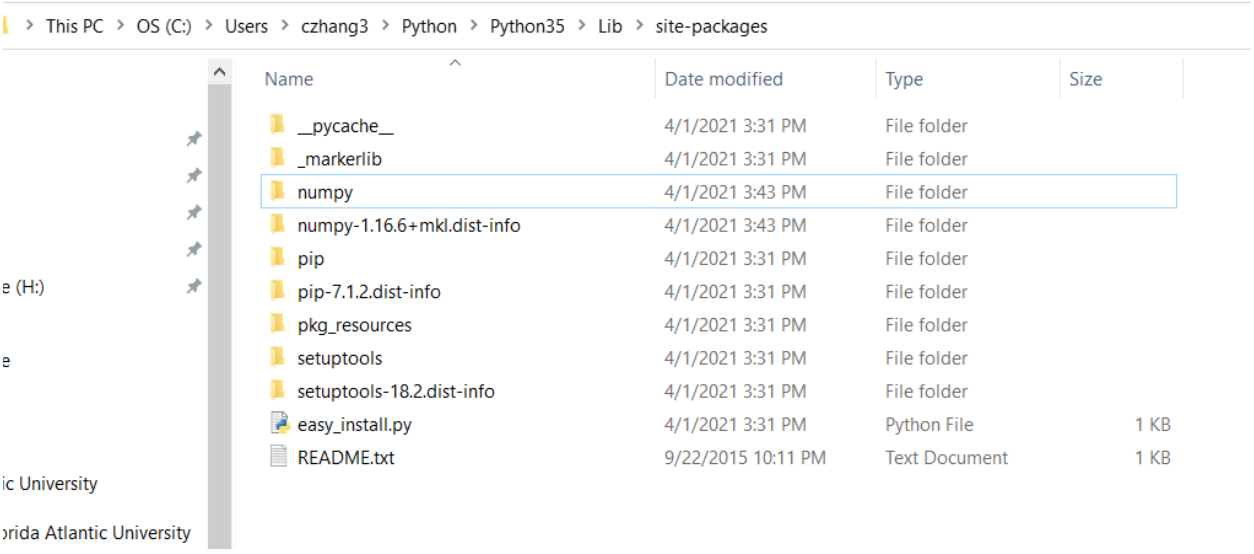
Pip install numpy

Pip does not work for Python 3.5 though pip has been installed in scripts folder of Python3.5 (C:\Users\czhang3\Python\Python35\Scripts). Here is the solution to solve the numpy installation.

Download the correct version of numpy from:
https://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy,
numpy-1.16.6+vanilla-cp35-cp35m-win_amd64.whl is the correct version compatible with

Python 3.5 and for windows 64-bit. Again, I have downloaded and shared it on Google Drive "\SJRWM\Software". Other versions do not work for Python 3.5.

Unzip/extract numpy-1.16.6+vanilla-cp35-cp35m-win_amd64.whl using 7-zip (https://www.7-zip.org/). If you have not installed 7-zip, a free software to compress/extract files/folder, you can install 7-zip. Two folders are unzipped from this file, and then copy these two folders to "C:\Users\czhang3\Python\Python35\Lib\site-packages" (the Python3.5 folder), shown as below.




You can test numpy by type in "import numpy as np" in the Python 3.5 console. An error message should not appear if it is correctly installed.

### 5. Download OTB

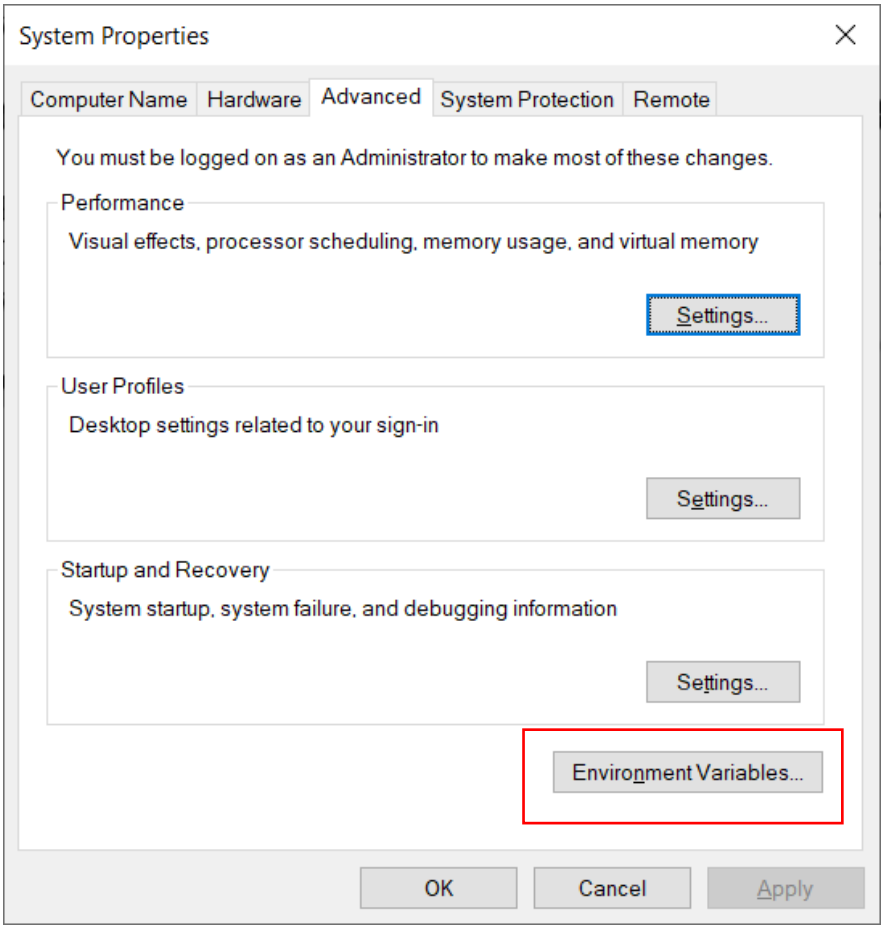
Download OTB from https://www.orfeo-toolbox.org/download/, select Win64 bits, and use 7-zip to extract the file to "C:\Users\czhang3\OTB-7.2.0-Win64" or your own working folder such as "C:\\users***\ OTB-7.2.0-Win64". Again this file is shared in Google Drive \SJRWM\Software.

Run otbenv.dat file in "C:\Users\czhang3\OTB-7.2.0-Win64\" by directly clicking this file. This will set up the OTB variable environment.

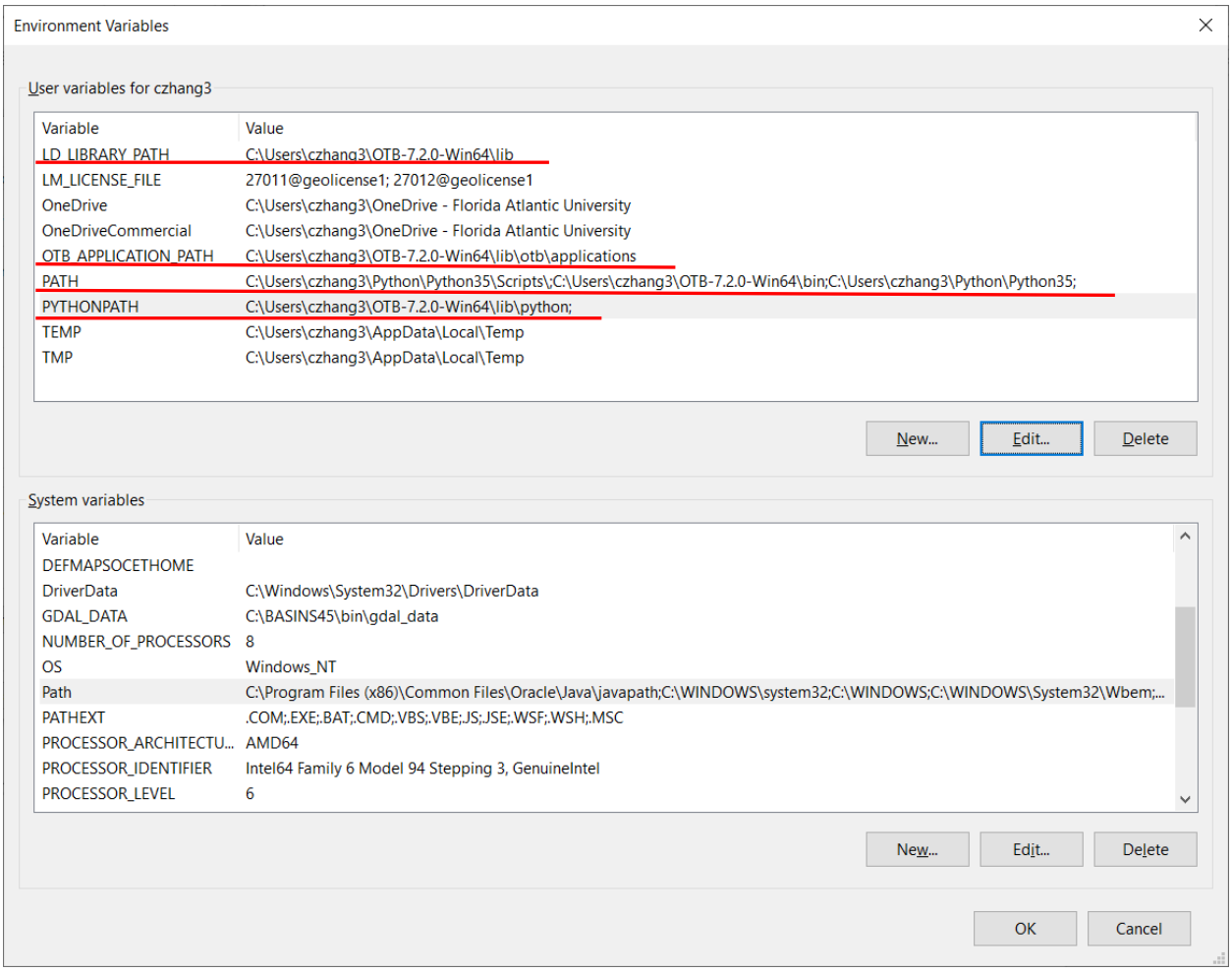
### 6. Set up path and environment for using OTB in Python 3.5

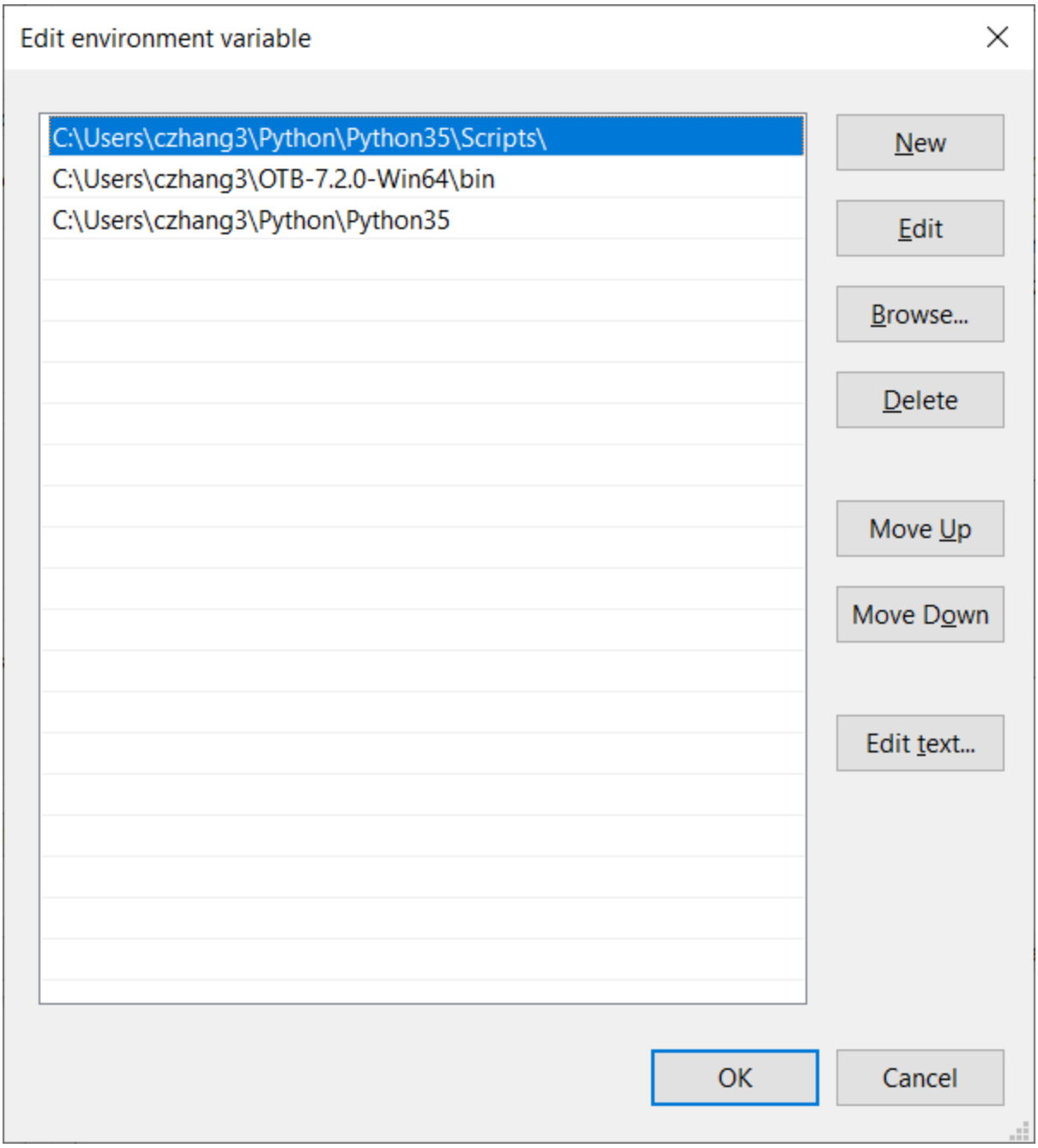
This part is critical in the setup. Both Python 3.5 and OTB have been downloaded and installed, now you need to connect these two. This is a critical step to link OTB with Python 3.5, and requires manually setting up environmental variables in the system.

Go to Control panel->System->Advance system settings->Environmental Variables


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

Manually add LD_LIBRARY_PATH, PATH, PYTHONPATH, and OTB_APPLICATION_PATH, and direct each variable to the relevant OTB and Python 3.5 folders , an example is shown below. The newly added PATH variables






This is the PATH variable editing window for me.

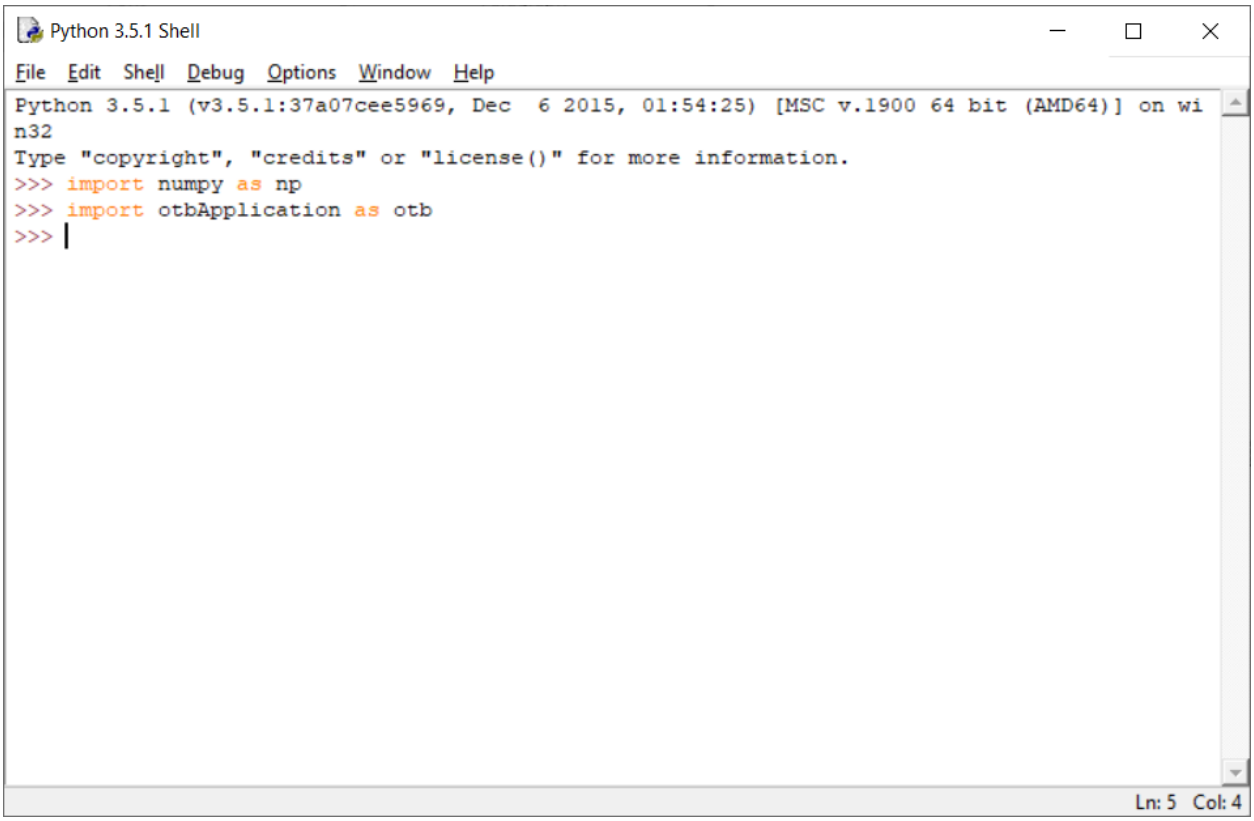
**7. Testing otbApplication**

Start IDLE (Python 3.5 64 bit)

Type in

Import numpy as np

Import otbApplication as otb




Testing of otbApplication as otb. There should be no error message when you import these two libraries.

**8. Testing Large Scale Mean Shift Segmentation in Python**

This part is extracted from

https://www.orfeo-toolbox.org/CookBook/Applications/app_LargeScaleMeanShift.html

LargeScaleMeanShift

Description

This application chains together the 4 steps of the MeanShift framework, that is the MeanShiftSmoothing, the LSMSSegmentation, the LSMSSmallRegionsMerging and the LSMSVectorization.

This application can be a preliminary step for an object-based analysis.

It generates a vector data file containing the regions extracted with the MeanShift algorithm. The spatial and range radius parameters allow for adapting the sensitivity of the algorithm depending on the image dynamics and resolution. There is a step to remove small regions whose size (in pixels) is less than the given ‘minsize’ parameter. These regions are merged to a similar neighbor region. In the output vectors, there are additional fields to describe each region. In particular the mean and standard deviation (for each band) is computed for each region using the input image as support. If an optional ‘imfield’ image is given, it will be used as support image instead.

Parameters

Segmentation as vector output options

Standard segmentation with labeled raster output options

**Input Image** `-in image` *Mandatory*

The input image can be any single or multiband image. Beware of potential imbalance between bands ranges as it may alter euclidean distance.

**Spatial radius** `-spatialr int` *Default value: 5*

Radius of the spatial neighborhood for averaging. Higher values will result in more smoothing and higher processing time.

**Range radius** `-ranger float` *Default value: 15*

Threshold on spectral signature euclidean distance (expressed in radiometry unit) to consider neighborhood pixel for averaging. Higher values will be less edge-preserving (more similar to simple average in neighborhood), whereas lower values will result in less noise smoothing. Note that this parameter has no effect on processing time.

**Minimum Segment Size** `-minsize int` *Default value: 50*

If, after the segmentation, a segment is smaller than this criterion, the segment is merged with the segment that has the closest spectral signature.

**Size of tiles in pixel (X-axis)** `-tilesizex int` *Default value: 500*

Size of tiles along the X-axis for tile-wise processing.

**Size of tiles in pixel (Y-axis)** `-tilesizey int` *Default value: 500*

Size of tiles along the Y-axis for tile-wise processing.

**Output mode** `-mode [vector|raster]` *Default value: vector*

Type of segmented output

**Segmentation as vector output**

In this mode, the application will produce a vector file or database and compute field values for each region

**Standard segmentation with labeled raster output**
In this mode, the application will produce a standard labeled raster.

Segmentation as vector output options

**Support image for field computation** `-mode.vector.imfield image`
This is an optional support image that can be used to compute field values in each region. Otherwise, the input image is used as support.

**Output GIS vector file** `-mode.vector.out filename [dtype]` *Mandatory*
The output GIS vector file, representing the vectorized version of the segmented image where the features of the polygons are the radiometric means and variances.

Standard segmentation with labeled raster output options

**The output raster image** `-mode.raster.out image [dtype]` *Mandatory*
This corresponds to the output of the small region merging step.

---

**Temporary files cleaning** `-cleanup bool` *Default value: true*
If activated, the application will try to clean all temporary files it created

**Available RAM (MB)** `-ram int` *Default value: 256*
Available memory for processing (in MB).

Examples From Python:

```
import otbApplication

app = otbApplication.Registry.CreateApplication("LargeScaleMeanShift")

app.SetParameterString("in", "QB_1_ortho.tif")

app.SetParameterInt("spatialr", 4)

app.SetParameterFloat("ranger", 80)

app.SetParameterInt("minsize", 16)

app.SetParameterString("mode.vector.out", "regions.shp")

app.ExecuteAndWriteOutput()
```

Here is my Python script for segmenting part of Buck Lake area. You can revise it to your sample imagery

```
# this script is written by: Caiyun Zhang
# this script is used to run Large Scale Mean Shift Segmentation in OTB
# 4/2/2021
import numpy as np
import otbApplication as otb
inFile="E:\\eCog_Temp\\Buck\\Bucktest2.dat" # define input imagery for segmentation
outFile="E:\\eCog_Temp\\Buck\\new\\seg2.shp" # define output ArcGIS shape file of
nSpatialr=50 # define spatialr input, integer
fRanger=50.0 # define ranger input, float
nMinsize=100 # define minimum segment size, integer
nTilesizeX=1000 # define tile size X, integer, default=500
nTilesizeY=1000 # define tile size Y, integer, default=500
# Create the application with codename "LargeScaleMeanShift"
app = otb.Registry.CreateApplication("LargeScaleMeanShift")
# Set up parameters
app.SetParameterString("in", inFile)
app.SetParameterInt("spatialr", nSpatialr)
app.SetParameterFloat("ranger", fRanger)
app.SetParameterInt("minsize", nMinsize)
app.SetParameterInt("tilesizex", nTilesizeX)
app.SetParameterInt("tilesizey", nTilesizeY)
app.SetParameterString("mode.vector.out", outFile)
# This will execute the application and save the output file
app.ExecuteAndWriteOutput()
print ("game over")
```