Semi-Supervised Water Boundary Detection Using Drone Imagery

Jack Howard
ECE Department
Stevens Institute of Technology
Hoboken, USA
jhoward1@stevens.edu

Saurabh Raman Parkar
ECE Department
Stevens Institute of Technology
Hoboken, USA
sparkar@stevens.edu

Ian Day
Department of Civil, Environmental and Ocean Eng
Stevens Institute of Technology
Hoboken, USA
iday1@stevens.edu

***Abstract*—Water boundary detection is an essential component in coastal research, especially in the fields of ocean engineering and oceanography. Detecting the boundary between water and land on coasts gives insight into the tidal patterns of the ocean. The task has inherent difficulties due to changing weather patterns and light levels over the water, leading to different colors of water in images. Similarly, drone imagery has a large volume of data without proper labelling, resulting in significant time delays and human intervention to accomplish the task. In this paper, a semi-supervised model using drone imagery is proposed to minimize the level of human interaction and increase the accuracy of water boundary detection. Experimental analysis on k-means clustering, k-nearest neighbors classification, logistic regression, support vector machine classification, and c-support vector machine classification are presented.**

I. Introduction

The task of determining water boundaries presents a challenging task in coastline research. Specifically, determining the specific boundary between land and water in a tidal environment is difficult, as the tide fluctuates the water line with time and can extend onto the land. Coastal boundary detection has significant applications such as developing a Digital Elevation Model (DEM), environmental surveillance, urban planning and ecological Research. Historically, water boundary detection was done through the manual effort of cropping images accordingly [1]. With current advancements in machine learning and deep learning, this task can be automated with supervised machine learning models that require large quantities of labeled data, which makes the computational requirements expensive.

To address the growing need of large volumes of labelled data, semi-supervised approaches were developed, where only a selection of the data requires labels [2]. These approaches have been applied to a variety of topics including wireless security [3] and ice-water classification [4]. Ultimately, this approach to data supervision dramatically reduces the amount of human time spent labelling these datasets. In extension of this, models aimed to specifically generate labels on data have grown in popularity [5]. Typically, these models rely on a statistical correlation existing within the data, thereby creating well-defined regions associated with each class. However, semi-supervised autolabellers address can address overlapping data through manually labelling boundary data with human supervision [6]. These models can be paired with supervised machine learning techniques, resulting in powerful models that rely on minimal human interaction.

In water boundary detection, image segmentation is the primary methodology utilized by machine learning and deep learning models [7]. This approach splits the image into regions corresponding to labelled objects in order to detect edges between [8]. As a general approach to edge detection, this method is reliable and widely applicable with corresponding labelled data. However, these typically require large volumes of labelled data of all objects typically found in the application-specific environment. To address this, region-based classification methods are utilized to group outlying data together as one class of data [9].

With these concepts, it becomes clear that a combination of a semi-supervised autolabeller and a region-based classifier may be powerful in water boundary detection. In this paper, we propose a semi-supervised machine learning classifier which can determine the water limits by classifying wet and dry sand on the coastline found with drone imagery through the following procedure:

1) Identify small windows where water and land meet through a semi-supervised autolabeller
2) Classify land and water within the smaller image through a supervised classifier
3) Extract the decision region for water boundary detection.

The rest of the paper is organized as follows. Section II discusses existing related works surrounding water boundary detection and the two main approaches to do so. Section III explicates our model with results and analysis. In Section III, there are three subsections A, B, and C corresponding to the description of the dataset, the machine learning algorithms used in the model, and the details of implementation including performance evaluation. Section IV outlines comparison between our approaches and other existing approaches. Section V discusses the future improvements that can be made to our model. In Section VI, we conclude this paper.

II. Related Work

In this section, we analyze previous approaches taken and separate them into two general categories: image pre-processing and satellite imaging.

A. Image Pre-Processing

Over the past few decades, advances in power efficiency of small devices has enabled the use of real-time detection in simple cases. Specifically, Unmanned Surface Vehicles (USV) are able to use texture entropy to extract low-brightness areas from a camera in real-time, thereby detecting a body of water [10]. Similarly, probabilistic models using stochastic relaxation on maximum a posteriori estimators are used to detect general boundaries in images by segmenting the boundary from the rest of the image [11].

Notably, these approaches rely exclusively on statistical analysis to either extract or trace the boundary, resulting in a disparity between human and machine performance [12]. Similarly, land with high surface reflection or inconsistent exposure to sunlight are known to disorient these models due to the lack of contrast between the two regions [13]. By consequence, these models may rely on human interaction for the entire process or may provide erroneous results. However, the complexity of these models are extremely small, meaning that most (if not all) modern devices can complete the task.

B. Satellite Imaging

Contemporary machine and deep learning models require large volumes of data to achieve high performance, prompting the use of satellite images to train models [14]. Specifically, the widespread availability of satellite images combined with the advancements of convolutional neural networks (CNN) have prompted investigations into deep networks for image segmentation [15]. Primarily, these methods rely on a large data volume to generate a large model to accomplish edge detection from a high altitude perspective.

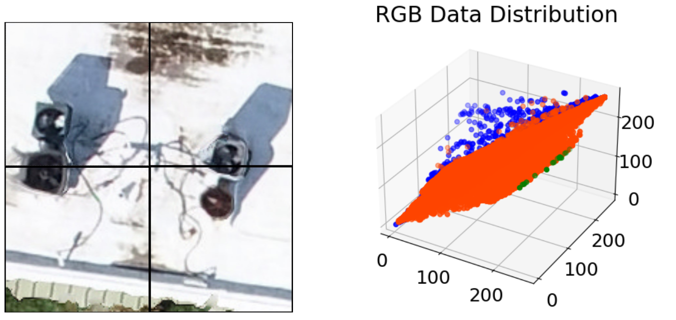
Generally, the drawback to using satellite images as a foundation for the data collection is the abstract viewpoint obscuring the exact boundaries [16]. Specifically, satellite imaging is aptly suited for large bodies of water - like oceans - to find a general water region around large bodies of land - like countries. However, the images lack resolution to be able to detect between a tidal region on the beach, where the exact water boundary is fluid and moves on a much smaller scale. Similarly, CNNs have a tendency to blur boundary pixels which further obscures the exact region. Despite this, satellite images are in abundance for this application, resulting in well-curated data to promote model performance. Currently, there are no well-established datasets using drone imagery, resulting in a need to collect and prepare the data for the model before creating a model.

III. Our Solution

In this section, we explicitly state how our solution described in Section I has addressed the data complexity and the classification infrastructure.

A. Description of Dataset

The raw training data set consists of two different components: orthomosaic (ortho) imagery of sandy beaches (as input data), and manually created dry beach outlines (as target/output). The ortho images are in GeoTIFF file format, containing georeferenced raster image data (RGB pixels). These files can be up to 3.8 GB in size, resulting in a need for significant memory capacity for training the model. To address this, the files are segmented into smaller, square sub-images with a 200 x 200 pixel size. The dry beach outlines are shape files (.shp), file type used in ArcGIS software. In this case, they are essentially georeferenced polygons.



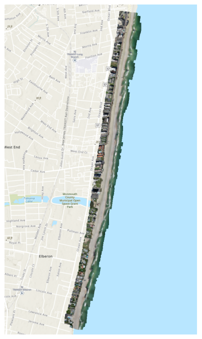
(a) Cross section of 100 pixel windows.

(b) Visualization of RGB values in R3 space.

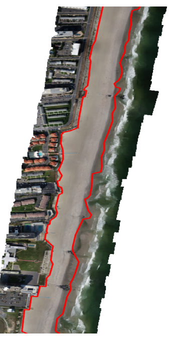
Fig. 1: Visualization of the dataset. A) is a representation of a 400 x 400 subsection of the image separated into individual 100 x 100 pixel windows. B) is a 3-D plot of the RGB values separated by class, with orange, green, and blue denoting 'Land', 'Unknown', and 'Water' labels respectively.

The data is being sourced from the Coastal Engineering Research Group at Stevens. Aerial photographs are collected using a DJI Phantom IV drone equipped with a 4K camera at 380 ft altitude. The individual photos are georectified and stitched together using the photogrammetry software AgiSoft Photoscan to create the orthos, in addition to Digital Elevation Models (DEMs). Each drone survey covers an area about 1,000 feet wide along 3 miles of North Jersey coastline. Photoscan outputs each survey as up to 4 individual image chunks, and each resulting ortho pixel covers approximately 1.2 inches square. An example full survey orthomosaic composing 4 individual GeoTIFF files is shown in Figure 2(a). Figure 2(c) demonstrates the scale of the original image pixels. The images are referenced to the NAD83(2011) New Jersey State Plane (ft US) geographic coordinate reference system, where the horizontal and vertical axes of the images correspond to easting and northing, respectively. The target classification data, in the form of dry beach outlines, was created by manually tracing the dry/wet beach line over the orthomosaics in ESRI ArcMap at non-specific map scales.

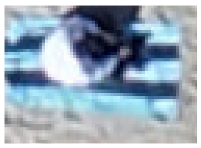
There were several problems while preparing the data for utilization. Primarily, the volume of information stored in the GeoTIFF files resulted in significantly slow loading times as well as conflicts with reading and writing using a comma separated file (csv) format. This was largely due to the original image dimensions, which were both very large (50,000 x 50,000 to 60,000 x 130,000 pixels) and inconsistent across the images. Furthermore, the raw images consist of large volumes of seemingly empty space, resulting in significant data overhead. Upon further inspection, these pixels were determined to be "black space" where the RGB values were all 255 and the Alpha was 0, thus creating an "empty" region when looking at the image that persists through the data.



(a) Full survey orthomosaic overlaid on a street map



(b) Dry beach outline over an orthomosaic image.



(c) A person sitting on a beach towel, demonstrating the scale of the original GeoTIFF pixels.

Fig. 2: Visual representation of original orthomosaic images. A) visualizes the region covered within one image. B) highlights a target region for boundary detection within the images. C) is an example of irrelevant objects that interfere with boundary detection.

To address this, the data was first loaded using 200 x 200 pixel windows and converted to their RGBA values. These values were then passed through an averaging algorithm to determine if the square window entirely consisted of black pixels. Based on the outcome of the algorithm, all empty windows were ignored and every other window was saved to it's own 200 x 200 PNG file. The same process was conducted using 100 x 100 pixel windows. By consequence, a new dataset was created for each original GeoTIFF image consisting of the average RGB values within the boundaries of the windows depicted in Figure 1. At the end of the process, there were between 9,999 and 56,491 stored windows, reducing the volume of each image by 10% - 84% for the 200 pixel windows, and 92% - 98% for the 100 pixel windows.

Separately, the orthomosaic dataset contained no labels on its own, resulting in the development of a baseline labelling system. Specifically, if the average RGB values in the window were in the blue region of the color spectrum they were labelled as 'Water', and if they were within the yellow region they were labelled as 'Land'. Values that were intermediary or outside of either region were labelled as unknown, indicating either a mixture of both land and water. These labels are not meant to be final and are adjusted throughout the model.

Alternatively, the previously mentioned dry beach outline shape files were used to label compressed orthomosaic images. This process was handled using MATLAB to take advantage of its Mapping Toolbox which can easily handle geospatial raster images and shape files. First, a fixed georeferenced grid was created with 10 ft cell size to span the entire area of a drone survey. Then, an original orthomosaic GeoTIFF image was read as a matrix of RGBA pixel values, and the image was cropped to the rectangular extent of non-empty data. Non-empty pixels were then grouped by their location within the fixed grid, and their RGBA values were averaged to create a new compressed raster matrix of 10 ft square pixels. Next, the dry beach outline was used to label all pixels within the outline polygon with a value of 1 (the positive class) and the pixels outside the polygon were labeled with a value of 0 (the negative class). Finally, the compressed raster was converted to a CSV file containing pixel locations (both relative to the image and spatially) along with RGBA values and labels. Figure 2(b) shows an example of a dry beach outline.

B. *Machine Learning Algorithms*

For the model design, there are two main components requiring different machine learning approaches. Specifically, the auto-labeller requires an unsupervised algorithm while the classifier requires a supervised algorithm. To address these, the k-Means clustering and k-nearest neighbors classification algorithms were selected for the respective components.

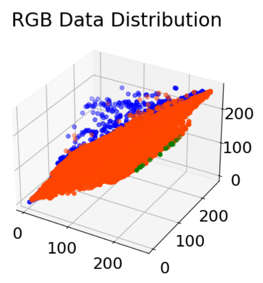
1) k-Means Clustering Algorithm: As an algorithm, k-means clustering presents an intriguing balance between intra-cluster and inter-cluster correlation [17]. Specifically, minimization of intra-cluster similarity for selecting centroids and maximization of inter-cluster similarity for decision boundaries present an opportunity to assimilate unknown data into classes. In other words, training the model with 3 labels and restricting the number of possible classifications to 2 allows for determining the region-specific water boundary. Additionally, the intra-cluster similarity enables visibility through local class validation within clusters by passing in data with known labels.

From a design perspective, k-means clustering allows for a reduction in unique labels under the assumption that a cluster equates to a label. To identify the relationship, a comparison of the original labels against each cluster allows for direct relabelling of unknown data. For example if 80% of the known 'Land' data falls is grouped under cluster 1, then all data within cluster 1 are labelled 'Land'. Inherently, this does not address the algorithm's susceptibility to noise when the data distributions are correlated. However, this is done purposefully to identify any 'Unknown' data as well as incorrectly labelled data from the initial labelling method in Section III A.

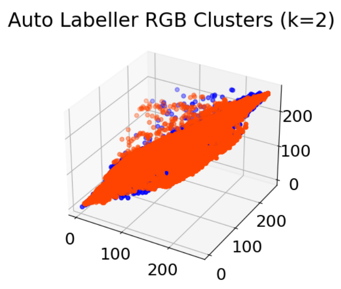
With this understanding, the number of neighbors was fixed at 2 with a cluster corresponding to either 'Land' or 'Water' labels. Additionally, the initialization was set to random to produce different clustering iteratively. The baseline accuracy of the model was 34%, measured by determining the maximum of the percentage of 'Land' data labelled within each cluster. This will be explicated in further detail in Section III C1.

2) k-Nearest Neighbors Classification Algorithm: Algorithmically, k-nearest neighbors (kNN) classification has the same foundation as k-means clustering. Notably, the difference exists on boundary point performance, where k-means relies on the inter-cluster similarity while kNN relies on weight functions on each neighborhood. Furthermore, this classification method allows for variable neighborhood sizes that can differ from the k-means deterministic size, resulting in an additional level of label validation.

In principal, combining these two techniques in series - where k-means propagates the labels to kNN - increases the robustness for classifying data near the decision region. This aligns with the purpose of the classifier as a refined decision region results in pixel-specific water boundary detection. Furthermore, this enables the computational cost to be reduced



(a) Original data labels.



(b) Autolabeller results with k=2.

Fig. 3: Visualization of the dataset. A) is the representation of the data prior to the autolabeller. B) is a 3-D plot of the RGB values separated by cluster, with orange denoting 'Land' and blue denoting 'Water'. The green data denotes the 'Unknown' data, which are assimilated in to the 2 clusters in B).

significantly after training, as it can be used to classify pixels only within 'Unknown' windows.

For these reasons, the kNN algorithm was implemented to refine the water boundary detection. At initialization, the weight distribution was uniform and the number of neighbors was 4. The base performance of this model on the original data (including 'Unknown' labels) was 82%. The base performance using the k-means labels was 84%. The optimization of this model is outlined in Section III C2.

3) SVM and SVM-C Classification Algorithm: Support Vector Machine was selected for classification due to how SVMs are resilient in nature and deal very well with outliers over other classification methods. SVMs can also deal with high dimensional data utilizing kernel trick. RBF works by mapping the input data into a higher-dimensional feature space, where the classes can be separated by a hyperplane. This classifier utilizes this advantage of Radial Basis Function (RBF) kernel, which enables non-linearity and flexibility in the decision boundary. This is required to accommodate the characteristics of water boundaries in images.

As an alternative form, SVM can be adopted using c-support vectors (SVM-C). These vectors are inherently linear by nature, requiring the hyperplane separation to also be linear. This additional condition is separate to the SVMs, as the ability to represent higher dimensionalities becomes difficult. However, this condition can be satisfied so long as the margins are can be linearly solved. As such, SVM-C can outperform SVM models in linearly separable cases.

C. Implementation Details

In this section, we explicate the optimization process, experimental results, and analysis of these results.

1) Autolabeller: In order to achieve high labelling performance, the autolabeller requires a clearly separable dataset. As shown in Figure 3(b), the clusters are not noticeably separable. To address this, linear discriminant analysis (LDA) was performed on the dataset to increase the separability of the data in 3-dimensions or create a lower-dimension feature space with higher separability.

	Cluster	1D	2D	Base
TP	1	49.9%	51.9%	52.9%
TP	0	50.0%	51.0%	47.9%
FP	1	54.3%	50.2%	49.4%
FP	0	52.1%	50.5%	53.4%

TABLE I: Average true and false positive rates for autolabeller with LDA reduction to 1D, 2D and no reduction.

The performance was quantified through calculating the sensitivity (true positive rate) and false positive rate of 'Land' and 'Water' labels in the autolabeller. The true labels were given by the initial labelling scheme and were used to compared the maximum percent of true positives in each cluster, thus giving the corresponding label of the cluster. These calculations were performed through 10 iterations with random initial cluster starting points, for 1D, 2D, and 3D (original) cases of the dataset. These values are expressed in Table I.

Based on these results, it is clear that reducing the data to 2-dimensions provides the best labelling performance. Furthermore, the closeness between the two rates indicates that the clusters may be separating the data perpendicular to the actual separation, indicating further separation is required. The significance of this discovery has yet to be determined due to the results of the classifier in Section III C2.

Furthermore, dataset for water border detection includes images with extremely high dimensions (about 20,000 x 50,000) and high memory size(1-2GB). Therefore, the first phase in the label generation process involved downsampling these images to bring ease in computation. The height of every image was reduced to 250 pixels maintaining the aspect ratio, and the RGB values were normalized to the range of 0 to 1. The images were then transformed into dataframes, where each image was represented by columns labeled 'R', 'G', and 'B' denoting the RGB values, and 'xPos' and 'yPos' indicating the pixel location.

The k-means clustering algorithm was applied to partition the RGB values into two clusters, separating the blue and green components of the sea water from the rest of the area. However, a drawback of segmenting photos using this method is that it clusters pixels based on color similarity and would also group together grass patches and swimming pools as Seawater.

To overcome this limitation in labeling the pixels of the image, around 40% of the pixels on the left side that were initially labeled as water were reclassified as land. Furthermore, the images had inappropriate dimensions, with transparent pixels present on both sides. To address this issue, 15% 1of the pixels on the right side were transformed into water. These adjustments were done considering that, in a typical image inside the collection, the left side of the image consists of sand and cityscape, while the right side represents the sea. As displayed in Figure 4 labelling the data by manipulating clustering, refines the output eliminating the irrelevant patches on the left end. The K-means clustering was evaluated with inertia of 2518.787 and Cluster Score of -26811.489.

2) kNN Classification: To improve on the final classification, the kNN algorithm was optimized iteratively. Fundamentally, this would provide a graphical representation for

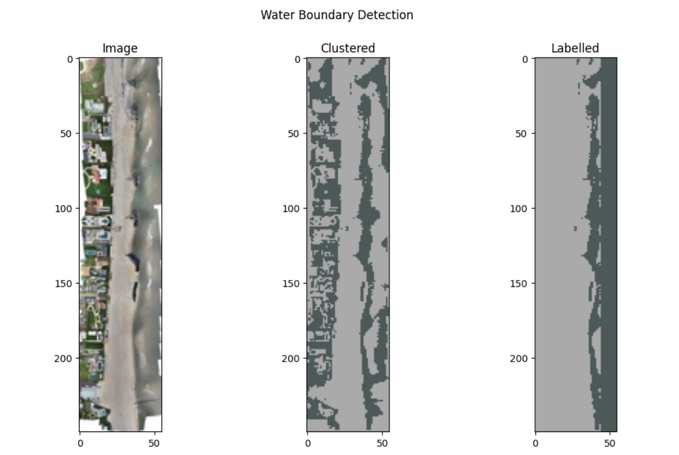


Fig. 4: Side-by-side representation of the original image, clustering image, and the labelled image.

the optimal tuning parameters as well as define a relationship between the autolabeller and the classifier. In practice, there were two measurements of performance: one using the original labels and another using the labels from the autolabeller.

Before comparing performance, optimization of the model parameters was conducted. As shown in Figure 3, the optimal number of neighbors for the model is 5 and the optimal weight distribution is uniform. These values were calculated through iteratively training and testing the model after setting the weight distribution to uniform, then incrementally increase the number of neighbors from 1 to 25. This procedure was conducted again using the inverse distance between points as the distribution function. Both datasets were used separately during the optimization procedure, yielding the same results.

The performance of the optimized model against both sets of labels was done using the accuracy score of the classifier. As previously mentioned in Section III B2, the baseline accuracy on the initial dataset was 82% and the baseline accuracy with the autolabller was 84%. After optimization, the accuracy using the initial dataset was 89% and the accuracy using the autolabeller was 99%. These correspond to a 8.5% and 17% overall performance increase in the model.

Notably, the performance increase using the autolabeller is sharp. This can be attributed to the similarity in mathematical foundation between k-means and kNN, where both algorithms generate similar decision regions. Primarily, this indicates a codependency between the generated labels and the final classification, which may negatively impact the water boundary detection.

3) Logistic Regression Classification: In order to reach new conclusions using the labelled dataset, a logistic regression model was created. Based on the kNN results, the data may be linearly separable with dispursed overlap between classes. Consequently, the logistic regression algorithm should provide Ultimately, the model The best parameters were determined to be: L2 penalty, Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) solver, maximum of 1500 iterations, and balanced class weights.

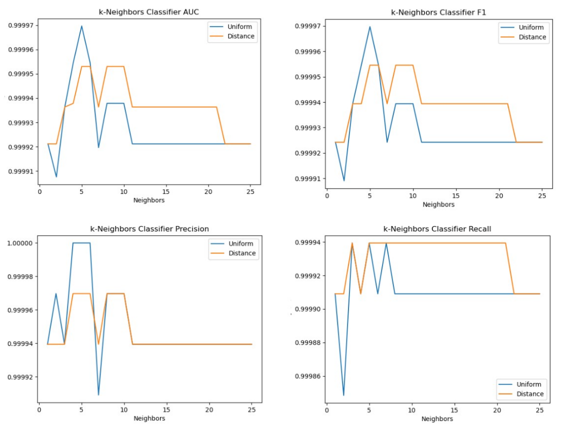


Fig. 5: Performance of kNN classifier against the labels generated from autolabeller. These graphs were used to determine optimal number of neighbors and weight distribution to be 5 and uniform respectively. The values being nearly 100% indicate a high correlation across neighborhood sizes.

Data Class	Precision	Recall	Accuracy
Training Class 0	92%	84%	85%
Training Class 1	72%	84%	84%
Testing Class 0	93%	95%	91%
Testing Class 1	89%	85%	90%

TABLE II: Logistic regression performance using both generated labels and preliminary labels.

The performance of the classifier is shown in Table II. Notably, the classifier performed poorly when implemented into the complete model with a maximum accuracy of 64.44%, indicating instability due to noise. The consistent performance for both precision and recall indicates a heavy reliance on initial labelling, which explains the high performance when using the original labels compared to the estimated labels.

4) SVM Classification: The model was implemented in Python using scikit-learn. One image's labeled CSV file was read and any remaining empty pixels were removed from the data. The data was downsampled to 1000 sample pixels per label for a total of 2000 samples in the training and testing dataset, which were split by an 80/20 ratio. Figure 6 displays the samples in 3- and 2-dimensional color spaces. The RGB data was then scaled using StandardScaler before being used to fit the SVC algorithm with a Gaussian RBF kernel. After an initial training and testing with default hyperparameters, a grid search was performed using the GridSearchCV function to find optimal parameters "C" and "gamma", with results of 1000 and 1, respectively. The scoring metric used was the F1-score, as the goal was to optimize precision and recall. Within the downsampled data, training data precision and recall were 0.92 and 0.98, respectively, and for testing data the scores were 0.90 and 0.97.

The similarities between the training and testing scores indicate that the model was not overfit to the training data. After the optimized model was developed, it was applied to the full image data before downsampling, which included both

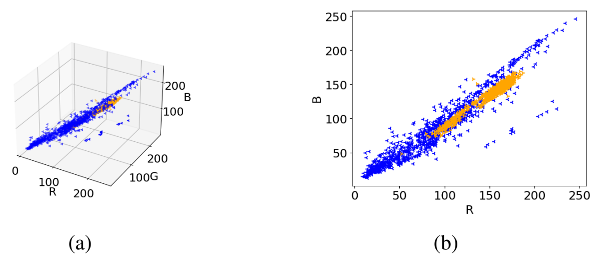


Fig. 6: Visualization of the downsampled manually labeled data in (left) Red-Green-Blue color space and (right) Red-Blue color space. Orange markers indicate “dry beach” labels (positive class) and blue markers indicate “other” labels (negative class).

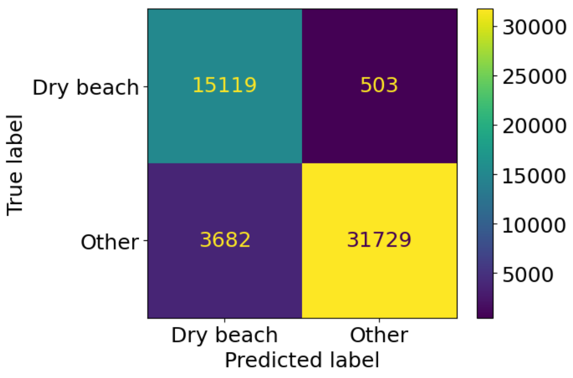


Fig. 7: Confusion matrix for the optimized support vector classifier applied to the full image.

the training and testing data as well as many more datapoints which were not used to build the model. The precision score across the full image was reduced to 0.80, but the recall remained at 0.97. The confusion matrix is shown in Figure 7. In other words, the model erred towards classifying a pixel as positive, so it tended to capture the vast majority of the dry beach pixels while also classifying some “other” pixels as dry beach. An issue with evaluating the model based on ground truth related scores is that the manual labels are noisy, with some dry beach classified as “other”, an ambiguous border between dry beach and water, as well as random objects on the beach being classified as dry beach while existing anywhere on the RGB color spectrum.

To better understand the model’s performance conceptually, the predicted labels were combined with the original pixel locations and plotted to produce a map of predicted labels, alongside the “true” labels in Figure 8. The model appears to do especially well differentiating the water from the dry beach (water is on the right), but foam on the waves probably gets misidentified often, which would explain the streaks of misclassified water near the shoreline. Understandably, many pixels behind the beach are classified as part of the beach. This can include any random features like buildings, roads, lawns, etc. Also, we can see places on the beach that were classified as other, which likely includes other objects present on the beach such as groins (shore perpendicular rock structures).

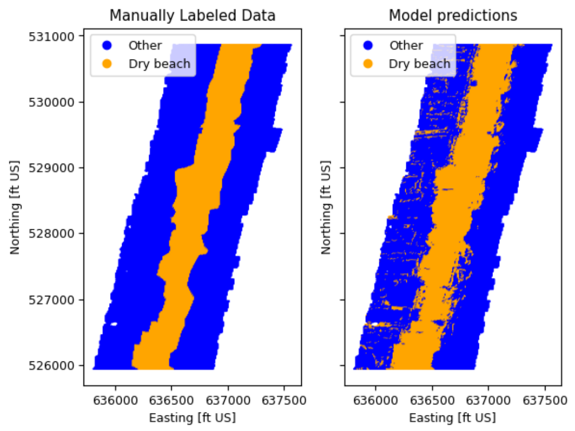
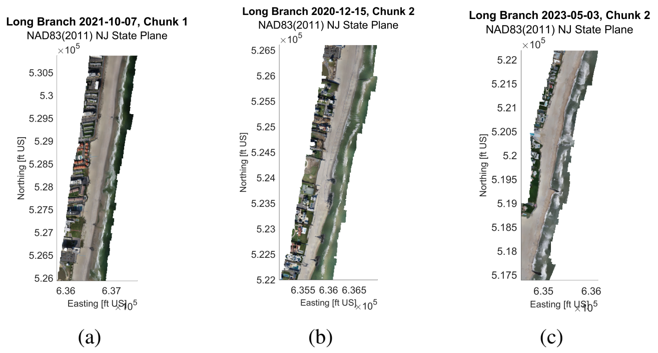


Fig. 8: Map of labeled and predicted classifications for the image used to train the SVC.



Long Branch 2021-10-07, Chunk 1
NAD83(2011) NJ State Plane
Long Branch 2020-12-15, Chunk 2
NAD83(2011) NJ State Plane
Long Branch 2023-05-03, Chunk 2
NAD83(2011) NJ State Plane
Northing [ft US]
Easting [ft US]
(a)
(b)
(c)

Fig. 9: Orthomosaics of the three images used to train and evaluate the model. Images numbered 1, 2, 3 from left to right.

This model built from a singular image was then tested on two more images to gain further insight into its performance. The three images used here are shown in Figure 9. The model had very poor performance with image 2, shown in Figure I. The precision score was 0.73 and the recall was 0.66. Referring back to its orthomosaic, it appears that for some reason, the color is shifted a bit on the south half of the image, perhaps from Photoscan’s image processing. The north-south difference in the accuracy displayed in Figure I looks like it is probably due to this color difference. The model must have been trained on sand of a color similar to the north half, and the color difference is enough to push the southern pixels outside the decision boundary. The model performed well on the third image with precision of 0.84 and recall of 0.87, with an unremarkable map comparison.

This model could be greatly improved by combining labeled pixels from multiple images across multiple surveys rather than using a single image for model training. This would allow the model to be trained with data that includes more varied shading and coloring of the sand and water, whether from real differences, light levels, camera settings, and image processing differences. This would theoretically fix the issue seen with the second image. To test this concept, the 3 images previously evaluated were combined into a single CSV file and

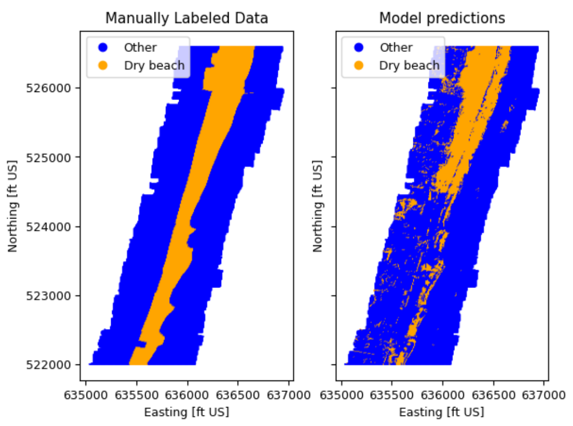
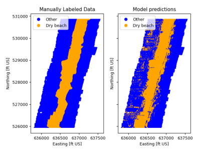
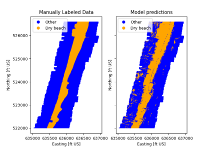


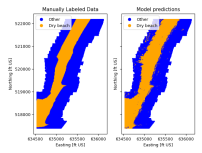
Fig. 10: Map of labeled and predicted classifications for image 2 run through the model trained on image 1.



(a) Map of labeled and predicted classifications for image 1.



(b) Map of labeled and predicted classifications for the image 2.



(c) Map of labeled and predicted classifications for the image 3.

Fig. 11: Visual representation of boundary detection using the model trained on all three images. Results for image 1, 2, and 3 are denoted by A), B), and C) respectively.

used to train a new model using the same process as before. In order of images 1 to 3, the precision scores were 0.75, 0.68, and 0.82, while the recall scores were 0.95, 0.95, and 1.00. Again, the model had higher recall than precision scores. The recall performance was very good across all 3 images. The final classification maps are shown in figures 11(a), 11(b), and 11(c). These maps show that again, the model does better differentiating between the beach and water than between the beach and land behind the beach, so most of the false positives can be attributed to other land-based pixels.

5) SVM-C Classification: As seen in Figure 12 the SVM-C classifier creates an overall smoother decision boundary and creates an optimal classification between two regions eliminating the discrepancies in the Labelled dataset. In this case, the discrepancies are caused by sea-waves. The classifier is evaluated on accuracy based on the generated labels as shown in Table III. Although these accuracies aren't completely true, but they give an overall better understanding of classification. The SVM-C model had an 85% training accuracy and a 91% testing accuracy.

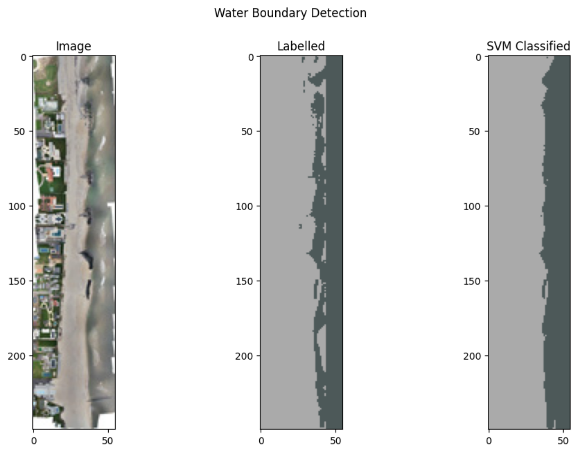


Fig. 12: Comparison of original image (left), clustering results (middle), and SVM-C classified data (right).

Data Class	Precision	Recall	Accuracy
Training Class 0	92%	84%	85%
Training Class 1	72%	84%	84%
Testing Class 0	93%	95%	91%
Testing Class 1	89%	85%	90%

TABLE III: SVM-C classification results using generated labelling.

6) Future Implementation Plans: Potential future improvements to this model could be to treat water as the positive class instead of dry beach, as was done with the unsupervised learning model. This could potentially result in a clearer decision boundary with better results while still accomplishing the task of delineating between the beach and the water. This would require an adjustment to the labeling script and rerunning it to create new CSVs for each image. Ideally, more images would be used to train the model, but the process of labeling the data was time intensive, both in terms of drawing polygons and running the MATLAB script. A second classification step could be implemented which takes advantage of the clustering of the predicted labels in x-y space to define sharp boundaries between regions, whether the positive class is dry beach or water.

IV. Comparison

For comparison, the classification algorithms are evaluated. In order to determine the best performing model, a combination of quantitative and qualitative metrics are required. Specifically, the precision, recall, and accuracy are the determining metrics, with model simplicity and basic approach are key contributors. For basic approaches, the models are separated into 'Sand vs All' (SVA) and 'Water vs All' (WVA) classification approaches denoting which class is used as the determining factor. In essence, SVA will have irrelevant features classified as 'Water' while WVA will classify them as 'Sand' in our binary decision model.

For quantitative metrics, the logistic regression classification model performs 20% worse than the other models.

This is likely due to a large amount of data manipulation required to fit the feature space to the models. However, this is the simplest model implemented with the WVA approach. Comparatively, the kNN classifier outperforms all other classifiers. This model has relatively low complexity and and utlizes the WVA approach. Notably, this model will not create a new decision region from the generate labels, as the two algorithms used reach nearly the same conclusion. Therefore, kNN classification model is the least appropriate model while the logistic regression model is the worst performing model.

For the SVM comparison, the two models performed relatively similarly. Specifically, the SVM model had approximately 6% better recall and the SVM-C model had approximately 7% better precision. In terms of complexity, both models are nearly identical, with the c-support vectors being slightly better. Most notably, the SVM model uses a SVA approach while the SVM-C model uses a WVA approach. Because of this, the water boundary is less clear on the SVM when recreating the image. Therefore, the best classifier implemented is the SVM-C classifier.

V. Future Directions

Potential future improvements to this model could be to treat water as the positive class instead of dry beach, as was done with the unsupervised learning model. This could potentially result in a clearer decision boundary with better results while still accomplishing the task of delineating between the beach and the water. This would require an adjustment to the labeling script and rerunning it to create new CSVs for each image. Ideally, more images would be used to train the model, but the process of labeling the data was time intensive, both in terms of drawing polygons and running the MATLAB script. A second classification step could be implemented which takes advantage of the clustering of the predicted labels in x-y space to define sharp boundaries between regions, whether the positive class is dry beach or water.

VI. Conclusion

Detecting water boundaries through drone imagery is a critical component towards understanding ocean topology. As shown through this paper, we have successfully detected water-land boundaries to near human levels using SVM-C classification. Furthermore, we have created an accurate semi-supervised labelling scheme that can label both pixels and small windows of images using k-means clustering. By combining both of these models together, the cumulative model has shown to be resilient with larger data volumes.

Videos with continuous data streams have temporal information and increase the complexity of the data analysis. Furthermore, continuous data introduces new sources of noise and error, resulting in an increase to complexity. Further research using drone video may be significant due to the rising use of drones. Additionally, research into addressing temporal complications can be applied to all uses of drones, resulting in a more broad application.

References

[1] V. Paravolidakis, L. Ragia, K. Moirogiorgou, and M. Zervakis, “Automatic coastline extraction using edge detection and optimization procedures,” *Geosciences*, vol. 8, no. 11, p. 407, Nov. 2018. [Online]. Available: http://dx.doi.org/10.3390/geosciences8110407

[2] Y. Reddy, P. Viswanath, and B. E. Reddy, “Semi-supervised learning: A brief review,” *Int. J. Eng. Technol*, vol. 7, no. 1.8, p. 81, 2018.

[3] J. Ran, Y. Ji, and B. Tang, “A semi-supervised learning approach to ieee 802.11 network anomaly detection,” in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019, pp. 1–5.

[4] F. Li, D. A. Clausi, L. Wang, and L. Xu, “A semi-supervised approach for ice-water classification using dual-polarization sar satellite imagery,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2015.

[5] S. Zhang, O. Jafari, and P. Nagarkar, “A survey on machine learning techniques for auto labeling of video, audio, and text data,” *CoRR*, vol. abs/2109.03784, 2021. [Online]. Available: https://arxiv.org/abs/2109.03784

[6] I. Hassan, A. Mursalin, R. B. Salam, N. Sakib, and H. M. Z. Haque, “Autoact: An auto labeling approach based on activities of daily living in the wild domain,” in *2021 Joint 10th International Conference on Informatics, Electronics Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision Pattern Recognition (icIVPR)*, 2021, pp. 1–8.

[7] I. Levner and H. Zhang, “Classification-driven watershed segmentation,” *IEEE Transactions on Image Processing*, vol. 16, no. 5, pp. 1437–1445, 2007.

[8] X. Dai, M. Xia, L. Weng, K. Hu, H. Lin, and M. Qian, “Multiscale location attention network for building and water segmentation of remote sensing image,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–19, 2023.

[9] S. Leigh, Z. Wang, and D. A. Clausi, “Automated ice–water classification using dual polarization sar satellite imagery,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 9, pp. 5529–5539, 2014.

[10] Y. Liu, L. Ma, W. Xie, X. Zhang, and Y. Zhang, “Water boundary line detection for unmanned surface vehicles,” *Recent Advances in Electrical Electronic Engineering*, vol. 13, pp. 1145–1152, 2020.

[11] D. Geman, S. Geman, C. Graffigne, and P. Dong, “Boundary detection by constrained optimization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 609–628, 1990.

[12] W. Wijesoma, K. Kodagoda, and A. Balasuriya, “Road-boundary detection and tracking using ladar sensing,” *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 456–464, 2004.

[13] K. Heidler, L. Mou, C. Baumhoer, A. Dietz, and X. X. Zhu, “Hed-unet: Combined segmentation and edge detection for monitoring the antarctic coastline,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2022.

[14] Z. Miao, K. Fu, H. Sun, X. Sun, and M. Yan, “Automatic water-body segmentation from high-resolution satellite images via deep networks,” *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 4, pp. 602–606, 2018.

[15] R. G. Tambe, S. N. Talbar, and S. S. Chavan, “Deep multi-feature learning architecture for water body segmentation from satellite images,” *Journal of Visual Communication and Image Representation*, vol. 77, p. 103141, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047320321000870

[16] R. Holyer and S. Peckinpaugh, “Edge detection applied to satellite imagery of the oceans,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 27, no. 1, pp. 46–56, 1989.

[17] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” *Proc. 5th Berkeley Symp. Math. Statistics Probability*, vol. 4, pp. 281–297, 1967. [Online]. Available: https://projecteuclid.org/ebooks/berkeley-symposium-on-mathematical-statistics-and-probability/Some-methods-for-classification-and-analysis-of-multivariate-observations/chapter/Some-methods-for-classification-and-analysis-of-multivariate-observations/bsmsp/1200512992