Target Detection in Underwater Lidar using Machine Learning to Classify Peak Signals

Connor F. Hunt[1], Jacob T. Young[1], Jacob A. Brothers[1], Jonathan O. Hutchins[2], Luke K. Rumbaugh[1], David W. Illig[3]

[1] *Department of Electrical and Computer Engineering,* [2]*Department of Computer Science*
Grove City College, Grove City, PA, USA; huntcf17@gcc.edu
[3]*Electro-Optic and Special Mission Sensors Division, Naval Air Warfare Center – Aircraft Division, Patuxent River, MD, USA*

***Abstract*—A machine learning model was used to detect target signals for an underwater lidar system operating in turbid water. For this system, scatter returns from the underwater environment and shot noise can create false alarms with signatures resembling true target peaks. A binary peak classifier model was implemented that classifies each peak in the lidar signal as a target or a non-target. For a gray target positioned at 7 m downrange, ≥97% target detection accuracy was achieved at turbidities of c≤0.8 m^{-1}, ranging from clear waters up to simulated open ocean conditions, and ≥85% target accuracy was achieved at simulated harbor-like turbidities of 1.0 m^{-1} ≤c≤1.2 m^{-1}, where c is the attenuation coefficient of the turbid water. This approach can enhance the detection capabilities of a system already using frequency-domain discrimination techniques.**

Keywords—Underwater lidar, target detection, machine learning

I. Introduction

Lidar systems can be used underwater for range finding, proximity detection, and 3D imaging [1,2]. A common approach to 3D imaging is to emit a laser beam across an object of interest. Depth and brightness images are then formed, where each image pixel shows the object's distance downrange and its reflective brightness at a certain point. This approach depends on correctly identifying the signal being reflected from the object in the lidar time-of-flight (ToF) return waveform. Figure 1 shows an example of a depth image that was collected from a recent test tank lidar experiment using a variety of water conditions [3], as well as an example ToF return waveform from a single pixel.

A common challenge that arises when this approach to 3D underwater imaging is used in turbid water is that the receiver may see unwanted backscatter from particles in the water. This can create false peaks (both from scatter reflections and from scatter-induced shot noise) that make it hard to determine which peaks in the ToF-return correspond to the object [4]. Frequency-domain discrimination techniques can be used to improve performance in turbid water [5], but detection can still be problematic without employing outside aid.

This paper describes a machine learning (ML) approach to predicting which peaks are object returns, and which peaks are noise or scatter returns. The test tank experiment featured a gray 3D target consisting of a stack of cylinders [3]. The ToF returns from scanning that target were used to train a Fine Gaussian Support Vector Machine (SVM) binary classifier that determines if a given peak is likely to be a target peak ("true peak") or a scatter or noise peak ("false peak"). We tested this classifier on other ToF returns from the same experiment, and have found that at a target distance of 7 m, the ML model classifies peaks in clear and low-turbidity water (i.e. where the attenuation coefficient c was less than or equal to 0.8 m^{-1}) with ≥97% accuracy, and in turbid water (1.0 m^{-1} ≤c≤1.2 m^{-1}) with ≥85% accuracy.

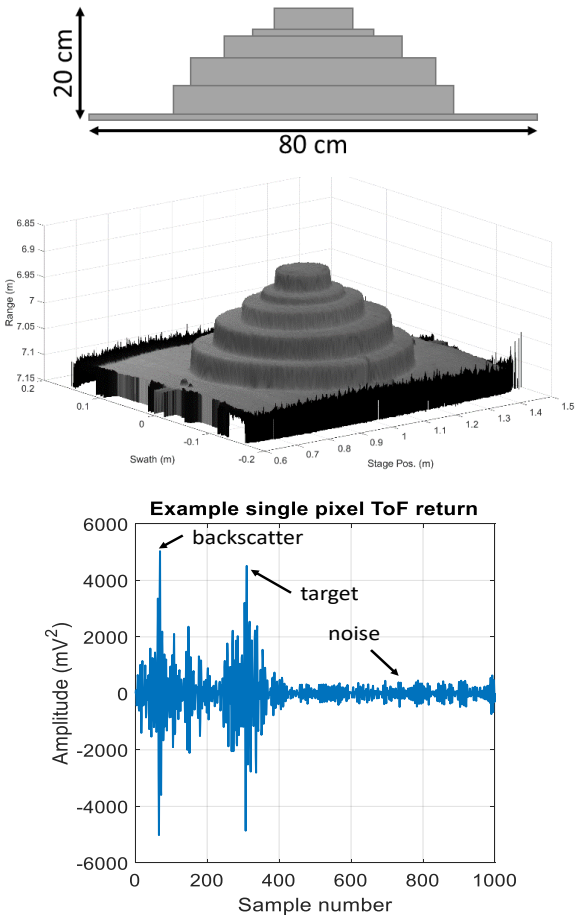

Figure 1. Underwater lidar depth image of a 3D target. Top: *The gray target used for the experiment [3].* Middle: *Clear-water depth image of target.* Bottom: *Raw lidar time-of-flight return, with target peak and backscatter/noise peaks highlighted.*

II. Methodology

A. Underwater Lidar Data Set

The underwater lidar data set is comprised of the ToF lidar returns collected while scanning a gray 3D target underwater.

The object was positioned 7 m downrange of the scanner and varied in height from 0 to 20 cm [3]. Six scans were included in the data set, at six different water conditions. The water conditions ranged in turbidity from an attenuation coefficient of 0.24 m^{-1} to 1.20 m^{-1}. Each scan was 350 x 350 pixels, so that the entire data set included 735,000 pixels. The time resolution of each ToF-return was 2.5 ns/sample, and each pixel included 1000 samples.

Importantly, the lidar system used a modulation of 500-1000 MHz, so that frequency-domain discrimination was already being implemented. The use of this highpass waveform was intended to reduce the backscatter signal observed in the ToF return waveform. Consequently, the ML classifier was trained on a data set that was already effectively highpass filtered.

B. Peak Data used for Training the Classifier

To build the training set, we first extracted all major peaks from all 735,000 ToF-returns. This was done by smoothing the magnitude of the ToF return and using Matlab's built-in "findpeaks" function. This typically resulted in 2-20 peaks per ToF return. Each peak was labeled as either a true peak or a false peak, based on the known location of the object. Each extracted peak record included the 100 time samples before the peak, the sample at the peak, and 300 time samples after the peak, for a total of 401 samples per peak record. In total, 3.1 x 10^6 peaks were extracted from the data set. From these, a training set of 12,000 peaks and a testing set of 12,000 peaks were selected at random, each with 1,000 true peaks and 1,000 false peaks at each of the 6 turbidity levels.

C. Features used by the Classifier

Eleven feature extraction functions were created, each of which processed a peak record and outputted a scalar value quantifying that feature. When training the model, we used all feature values together as inputs for the model. The features are listed in Table 1 and included: endBeginningDifference, maxPeak, endMinusBeginningSums, peakHeight, smoothedSymmetry, symmetry, averageAfterPeak, averageAbsAfterPeak, peakWidth, avgAbsBeforePeak, and avgBeforePeak, all to be described below. Unless otherwise stated, each feature processed the magnitude of the smoothed peak record.

The endBeginningDifference feature is meant to find the consistency of the noise after the peak. For a true peak, the noise will tend to be fairly flat after the peak. The value is calculated by taking the average of points 200 – 400 (100 after the peak up to the end of the sample) and takes the sum of the difference of each point to that mean.

The maxPeak feature aims to find if the target peak is the largest in the sample, and if not then by how much. It finds the largest peak within the sample and subtracts the target peak.

The endMinusBeginningSums feature acts the same way as endBeginningDifference, but instead of using the mean of the end, uses the mean of the first 50 points (ending 50 points before the sample), and subtracts the value at each point after 150.

The peakHeight feature pulls out only the suspected peak's value (at point 101) and returns its value. This function gauges whether a peak is a target peak or a backscatter peak solely based of off the magnitude of the peak.

The smoothedSymmetry function is trying to see to what extent the points are symmetrical around the peak itself. It smooths the data more than normal and returns the sum of the difference of each pair of points surrounding the target peak, up to 50 away. For example, it would take $abs(f(100) - f(102)) + abs(f(99) - f(103)) + \cdots$). This dataset is more smoothed than others.

The symmetry function acts the same as smoothedSymmetry but is not smoothed and reads up to 100 points away instead of 50.

The averageAfterPeak function takes the average of all the samples after the suspected target sample and averages them.

The averageAbsAfterPeak feature returns the mean of the magnitude of all the points after the featured peak.

The peakWidth function Finds the nearest minimum to the left and right of the suspected target and adds the distances from each to the suspected target.

The avgAbsBeforePeak function takes the average of the absolute value of the first fifty samples and returns that value.

The avgBeforePeak function takes the average of the first fifty samples and returns that value.

We used Matlab's Classification Learner App [6] to train a model that used all features simultaneously to predict whether the 101st time sample in a peak record represented an object. The 12,000 peak training data set was used for training, and the 12,000 peak testing data set was used for testing. Multiple types of classifiers were tested, and a Fine Gaussian SVM model [7] was selected because it had the highest accuracy.

The function value distributions for each feature extraction function are shown in Figures 2-12. These figures plot a random sampling of the function values for peaks from each turbidity. Many functions have good separation between the true peak and false peak distributions for low turbidities (especially turbidity 1), but the separation is not as apparent at high turbidities (especially turbidity 6). Figures 13-14 plot this same function value distribution data at turbidity 1 and turbidity 6. It can be seen that some functions do offer some separation at turbidity 6.

Table 1. Function results of every predictive feature, and its corresponding accuracy.

Feature	**Highest Accuracy**
avgAbsBeforePeak	89.9%
symmetry	88.6%
smoothedSymmetry	88.2%
averageAbsAfterPeak	86.0%
averageAfterPeak	84.9%
endMinusBeginningSums	84.1%
avgBeforePeak	82.3%
peakHeight	82.3%
maxPeak	81.3%
peakWidth	70.0%
endBeginningDifference	60.4%

D. Comparison with a Baseline Detector

A simple baseline system was used to detect true and false peaks for comparison sake against the ML system that we created. This system used the magnitude of the peak and the magnitude of the 200 samples after the peak to classify the peak. The system classified the peak as a true peak if the magnitude of the peak was at least four standard deviations above the mean of this record. As shown in Table 2, this classifier performed well in low turbidity, but did not do well in high turbidity.

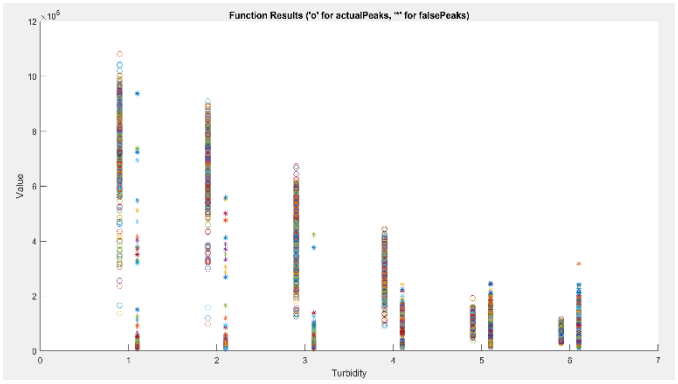

Figure 2. Function results of avgAbsBeforePeak. True peaks are on the left of the turbidity number and false peaks on the right.

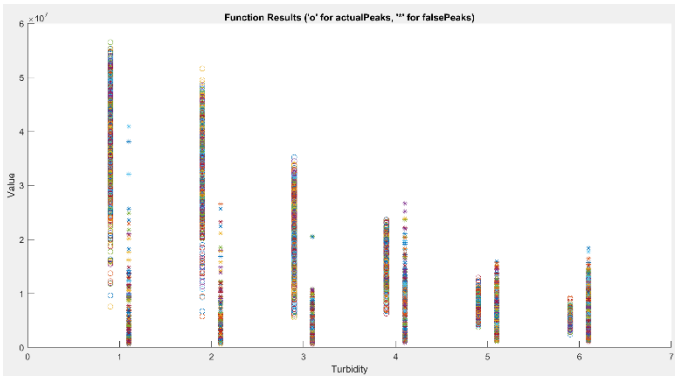

Figure 3. Function results of Symmetry. True peaks are on the left of the turbidity number and false peaks on the right.

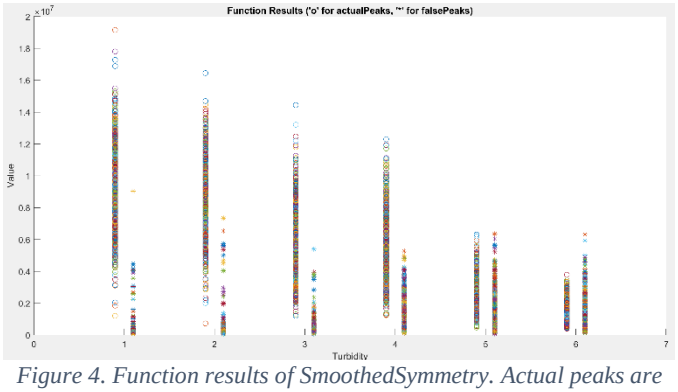

Figure 4. Function results of SmoothedSymmetry. Actual peaks are on the left of the turbidity number and false peaks on the right.

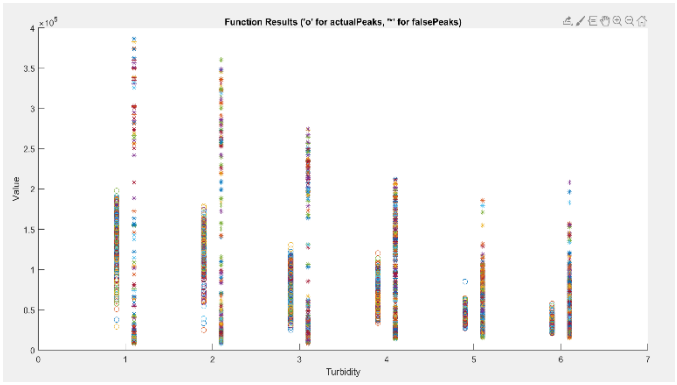

Figure 5. Function results of averageAbsAfterPeak. Actual peaks are on the left of the turbidity number and false peaks on the right.

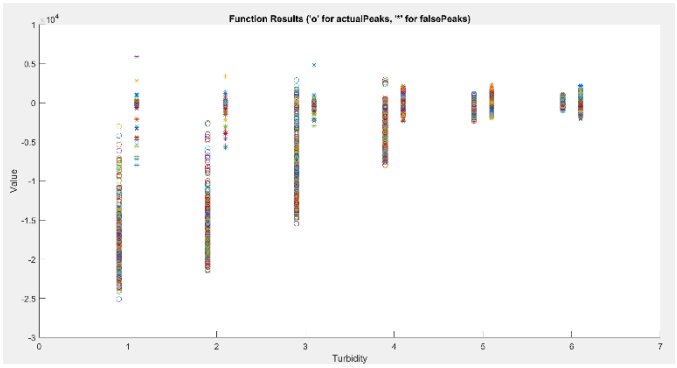

Figure 6. Function results of AverageAfterPeak. Actual peaks are on the left of the turbidity number and false peaks on the right.

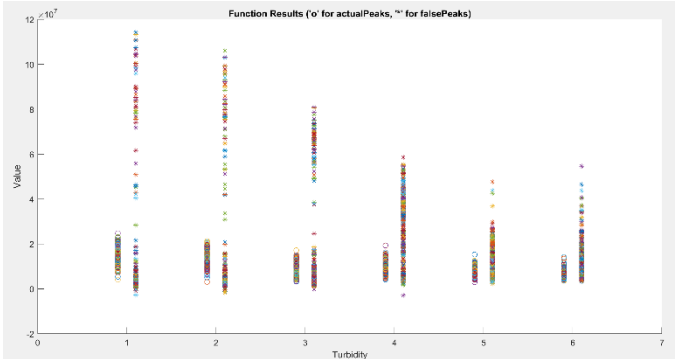

Figure 7. Function results of endMinusBeginningSums. Actual peaks are on the left of the turbidity number and false peaks on the right.

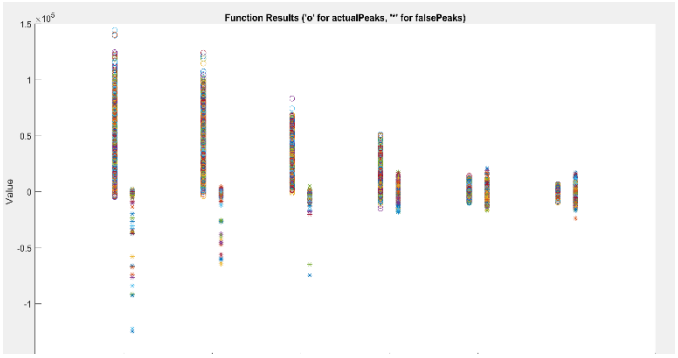

Figure 8. Function results of avgBeforePeak. True peaks are on the left of the turbidity number and false peaks on the right.

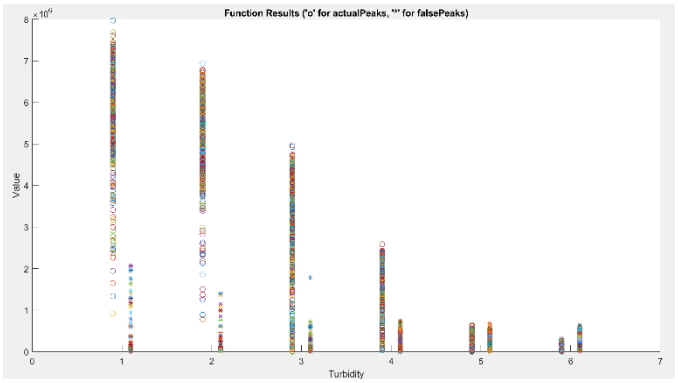

Figure 9. Function results of PeakHeight. Actual peaks are on the left of the turbidity number and false peaks on the right.

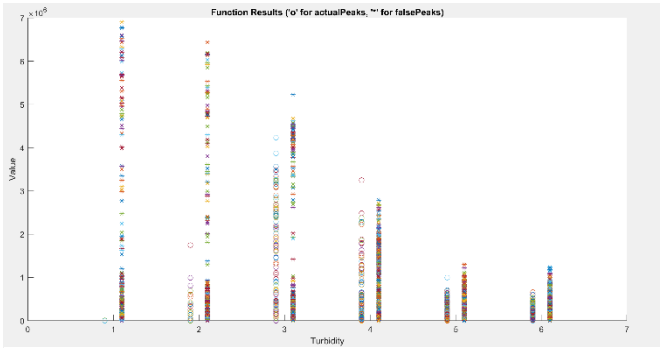

Figure 10. Function results of maxPeak. True peaks are on the left of the turbidity number and false peaks on the right.

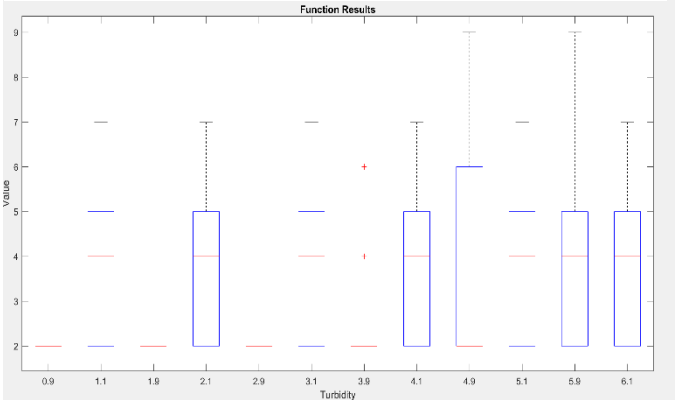

Figure 11. Function results of peakWidth. True peaks are on the left of the turbidity number and false peaks on the right.

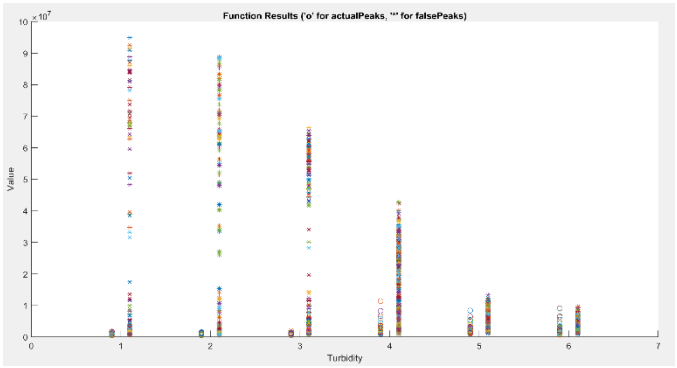

Figure 12. Function results of endBeginningDifference. True peaks are on the left of the turbidity number and false peaks on the right.

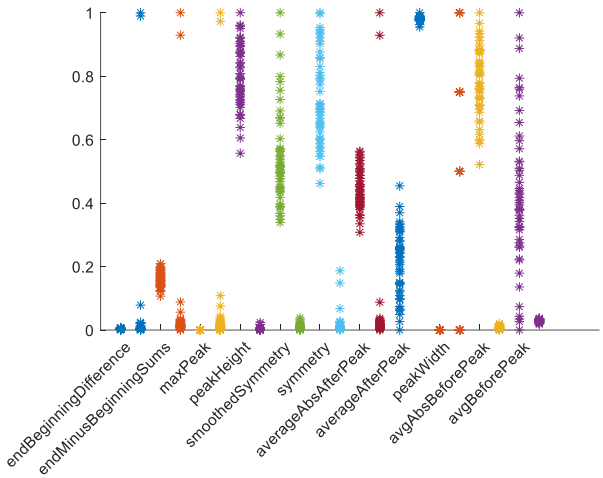

Figure 13. Function results from each classifier function running on Turbidity level 1. For each distinct color, points plotted left of center are false peaks, points plotted right of center are true peaks.

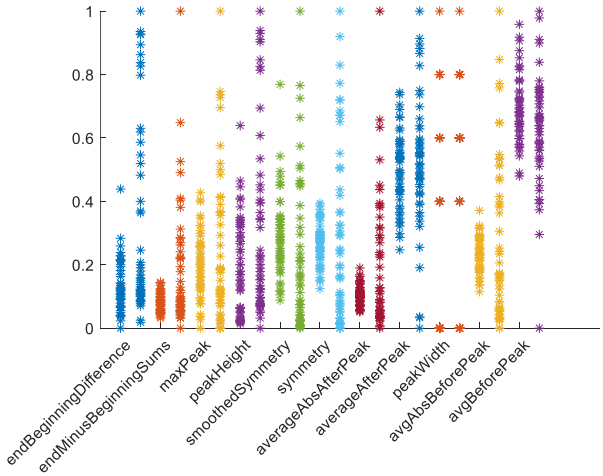

Figure 14. Function results from each classifier function running on Turbidity level 6. For each distinct color, points plotted left of center are false peaks, points plotted right of center are true peaks.

III. Results

Figure 16 visualizes the output of the model on both a clear-water (c=0.24 m^{-1}) and a turbid-water (c=1.2 m^{-1}) ToF return (corresponding to attenuation lengths of 1.7 to 8.4). The outer dashed purple vertical lines represent the maximum and minimum possible locations available for prediction (due to needing 100 samples before and 300 samples after). The remaining vertical lines indicate the sample at which a peak was detected. Red lines show peaks classified by the model as false peaks; green lines show peaks classified as true peaks. In this example, the actual target location was at sample #320. These examples were typical: peaks were well classified in clear water, while some false positives appeared in turbid water. As shown however, many false positives were near the correct location. Table 2 summarizes the accuracy of the model at each turbidity. The accuracy of the ML classifier was significantly better than the accuracy of the baseline detector, especially at high turbidities.

Table 2. Peak Classifier Accuracy by Turbidity

Performance of Peak Classifier Model			
Turbidity	***Attenuation coefficient***	***Classifier Accuracy***	***Baseline Detector Accuracy***
1	c=0.24 m^{-1}	99.88%	93.80%
2	c=0.4 m^{-1}	98.68%	93.75%
3	c=0.6 m^{-1}	96.85%	92.20%
4	c=0.8 m^{-1}	97.78%	91.55%
5	c=1.0 m^{-1}	93.83%	67.85%
6	c=1.2 m^{-1}	84.80%	47.20%

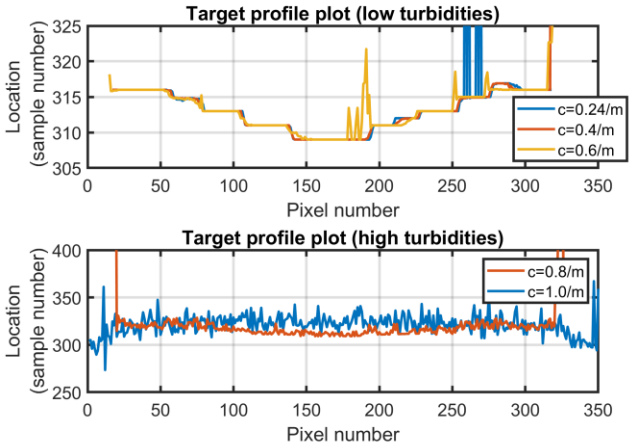

Figure 15. Plot of target profile using ToF returns processed by ML classifier. The profiles shown represent the average of 10 slices across the target for each turbidity except the c=1.2/m case. The c=1.2/m profile was not plottable on this scale since the peaks classified as "true peaks" varied widely in location.

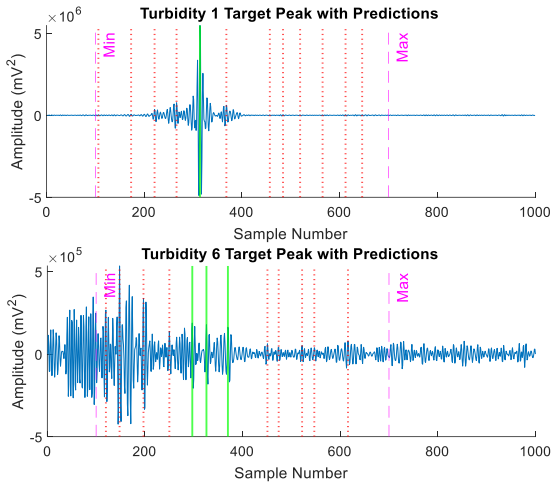

Figure 16. Example output of peak classifier. Top: *Output for clear water (c=0.24 m^{-1}): the true peak is clearly identified, and other peaks are correctly classified as false peaks.* Bottom: *Output for turbid water (c=1.2 m^{-1}): several peaks near the actual target are classified as true peaks. False peaks are correctly classified.*

IV. Conclusion

This work will improve performance for underwater lidar systems, especially those that already implement frequency-domain and other scatter-suppression techniques. The classifier works well for low turbidities and shows promise for use at high turbidities.

Future work will extend the feature set with a focus on improving results at higher turbidities, specifically by including adjacent pixel information in the feature set.

References

[1] Jaffe, J. S. (2014). Underwater optical imaging: the past, the present, and the prospects. *IEEE Journal of Oceanic Engineering*, *40*(3), 683-700.

[2] Caimi, F. M., & Dalgleish, F. R. (2013). Subsea laser scanning and imaging systems. In *Subsea Optics and Imaging* (pp. 327-352). Woodhead Publishing.

[3] Illig, D. W., Lee, R., & Mullen, L. (2018). Image processing inspired technique for enhancing performance of the underwater modulated pulse laser system. *Optical Engineering*, *57*(10), 107104.

[4] Illig, D. W. (2016). *Towards Enhanced Underwater Lidar Detection via Source Separation* (Doctoral dissertation, Clarkson University).

[5] Mullen, L. J., & Contarino, V. M. (2000). Hybrid lidar-radar: seeing through the scatter. *IEEE Microwave magazine*, *1*(3), 42-48.

[6] Mathworks Matlab. Classification Learner App. https://www.mathworks.com/help/stats/train-classification-models-in-classification-learner-app.html

[7] Jakkula, V. (2006). Tutorial on support vector machine (SVM). *School of EECS, Washington State University*, *37*.