Insect Identification in LiDAR Images Using Changepoint Detection

Caroline Xu, Trevor Vannoy, Brad Whitaker, Nat Sweeney, and Ryan Ficken
Department of Electrical and Computer Engineering
Montana State University, Bozeman, Montana, United States
Email: carolinx@umich.edu

***Abstract*—Entomologists want to study insects through LiDAR images, but the large amount of data makes it inefficient to manually sort through. Although supervised machine learning methods have been used, it still takes time to train the models, and the dataset still needs to be labeled. Thus changepoint detection, an unsupervised learning method, is a proposed alternative that could greatly reduce computational power and time. Using pre-processing and post-processing techniques, changepoint detection is a viable alternative to supervised machine learning. Although the algorithm performance is not ideal to completely replace machine learning models, there are many methods to improve the algorithm, and the current algorithm still has the potential to shorten time consuming processes.**

I. Introduction

Insect populations have been decreasing over the past couple of decades. Because they are at the bottom of the food chain and important pollinators, their disappearance will affect the entire food chain, including humans. Insect behavior needs to be understood to prevent their population decline, and this has been attempted using LiDAR images of insects. However this requires a method of identifying when and where the insects are in respect to the laser pulses from the LiDAR machine. Manually processing the data takes too long, and supervised machine learning models not only require labeled data, but also take a significant time to train. An unsupervised learning model that does not need to be trained can reduce the time and computational power to process this data. Therefore, using changepoint detection as an alternative to supervised machine learning can increase the efficiency and accuracy of automated insect identification processes.

Target identification in LiDAR images using machine learning techniques have been implemented by Vannoy et al. [1] in detecting fish underwater using an airborne bathymetric LiDAR system. Vannoy et al. [2] also uses similar machine learning techniques to detect insects in LiDAR data, which include a neural network, ADABoost [3], and RUSBoost [4]. These specific models are used because of the small percentage of insects in the entire dataset. Class imbalances in the data make it more difficult to train the model, especially since the goal is to detect all insects in the data.

II. Methods

A. Datasets

There are two datasets used in this research. The first dataset (Hyalite dataset) was taken near Hyalite Creek in September 2020 [5], and the second dataset (bee dataset) was taken at the Montana State University Honey Bee Research Site and Pollinator Garden.

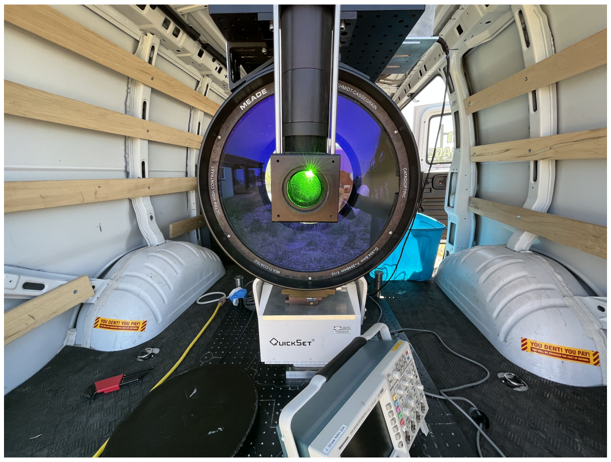


Fig. 1. Insect LiDAR used for data collection

The insect dataset consists of 10079 LiDAR images taken over the course of four days: 09/16/2022, 09/17/2022, 09/18/2022, and 09/20/2022. 173 of those images contained an insect.

The bee dataset consists of 4132 LiDAR images taken over the course of two days: 06/23/2022 and 06/24/2022. 1309 of those images contained an insect.

The LiDAR used for this data collection (**Fig. 2**) was developed by the Optical Remote Sensor Laboratory at Montana State University, and has a pulse frequency of approximately 8 kHz in order to detect wing beat frequencies of the insects [6]. All images are 178x1024 pixels. In MATLAB, they are stored as a 178x1024 matrix, with each value in the matrix being a pixel value. The rows represent the range bins the signal was detected in, with each range bin being approximately 0.75 meters. The columns represent each laser pulse. After collection, all of the data was normalized, where the dimmest pixel had a value of 0 and the brightest pixel had a value of 1. Only the normalized data was used for the changepoint detection algorithms.

A comparison between the original and normalized data is shown in **Fig. 2**, and **Fig. 3** shows sample images from the datasets.

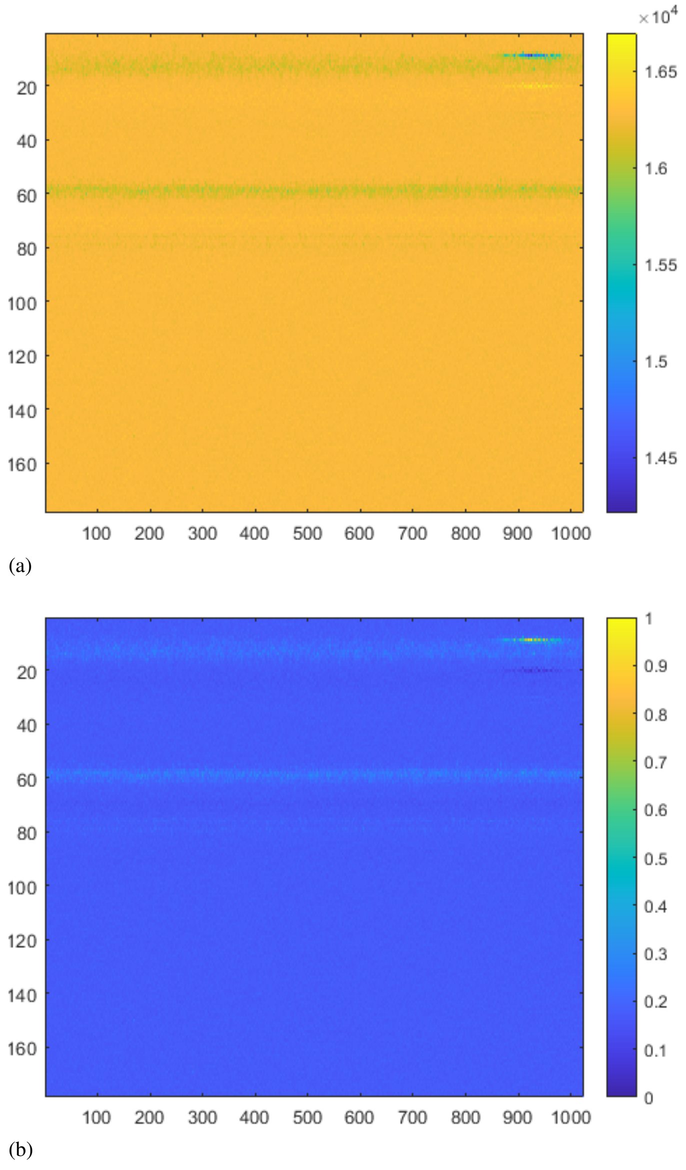


Fig. 2. a) Original LiDAR data versus b) normalized LiDAR data

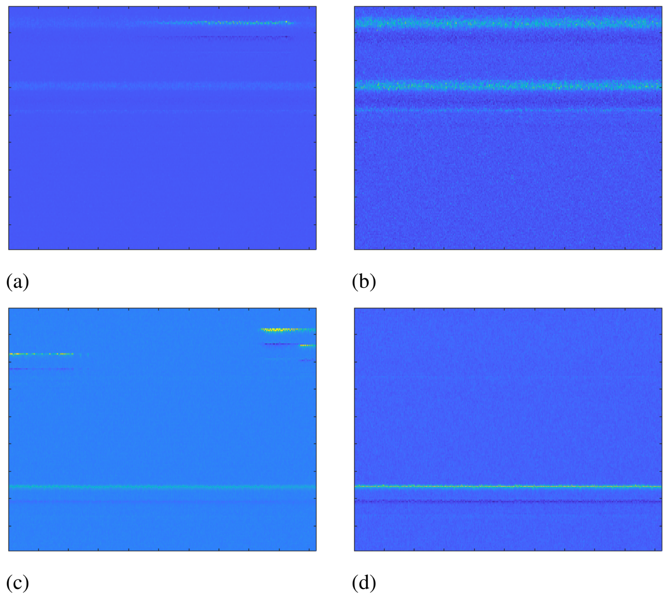


Fig. 3. Sample images from a) insect dataset with insect, b) insect dataset without insect, c) bee dataset with insect, and d) bee dataset without insect

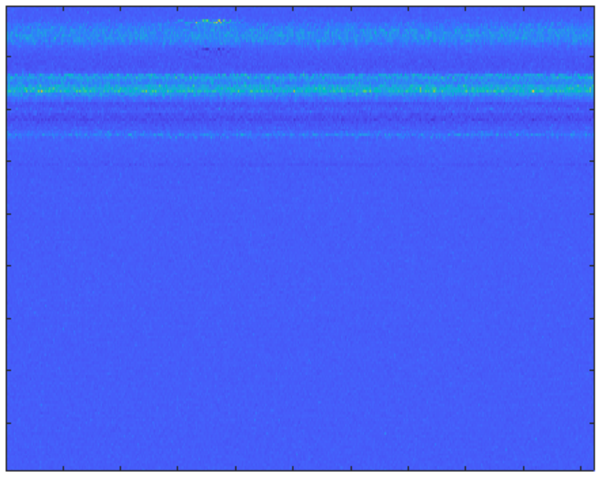


Fig. 4. Original image before any preprocessing has been done

B. Preprocessing

The changepoint detection function `findchangepts` in the MATLAB library can take either a 1-dimensional or 2-dimensional matrix as an input. Because of noise and discrepancies in the LiDAR images, using the entire matrix as the input parameter will not guarantee the detection of the insect in the image. Thus the input parameter was a row of the image, which is a 1-dimensional matrix. Because it would be inefficient to check all 178 rows of the image, preprocessing techniques needed to be done to remove unimportant rows.

Because the algorithm is detecting the location of the insect and not whether or not there is an insect present in the image, the remaining rows need to be recorded. A vector consisting of integers from 1 to 178 was created, and the corresponding numbers in the vector were removed when the row was removed from the actual image.

Fig. 4 shows the original image. The first rows removed were empty rows. Each row was split into eight equal parts, and the average of each part was found. If all eight averages were less than 90% of the image average, the row was considered an empty row and it was removed from the image. The corresponding row number was also removed from the vector that stored all the row numbers.

Rows that contained a hard target, or a DC signal, were removed next. Because the image was normalized to values between 0 and 1, the hard target threshold was simply set to 0.8, where anything above that value was removed from the image. Once again, each row was sectioned into eight parts, the average of each section was found, and the values were compared to the threshold. To maximize the number of rows removed, an additional threshold of $2.25 * imageAverage$ was established, and the same method of determining whether or not the row was above that threshold was applied.

There are also rows that are too noisy to be considered empty but not bright enough to be considered a hard target. Thus rows that stay relatively the same also need to be removed. Each row was divided into 16 sections and the average of each section was found. The percent difference between each section was found, and if all of the average had less than a 5% difference between them, the row was considered insignificant and thus removed.

Because insect wing beats have a frequency, there needs to be a way to tell the difference between an insect or a grass or leaf passing through the laser. To detect fake signals, a continuous 1-dimensional wavelet transform was performed on each remaining row using MATLAB's `cwt` function. The function produces a magnitude scalogram, where brighter spots in the image indicate a more prominent frequency. If there are no insects, the entire image is relatively consistent, otherwise a peak can be spotted at column locations the insect is at. Also, when there is less frequency, the maximum value in the scalogram is smaller, and that can also be used to set a threshold that determines whether or not there is an insect in the row.

To get rid of these rows with no frequency, each row is used as the input parameter of the `cwt` function. The absolute value of the resulting matrix/scalogram is taken, and the maximum value of the entire matrix is found. If the maximum value is less than 0.4, then the row is considered unimportant. If the maximum value is greater than 0.4, the scalogram is divided into eight side-by-side sections, and the average of each rectangle is found. If the percent difference between each of the averages is less than 10%, then it is considered an insignificant row and is also removed.

After each row removal, a conditional statement checked whether or not there were any rows left in the image. If there were not any rows, the algorithm would label it as no insects and it would finish the iteration. This row removal process was tested on the entire insect dataset to ensure that no insects would be removed during this process. Because it did not remove any insects in the insect dataset, it is assumed that no insects are removed from the bee dataset. **Fig. 5** shows how the image changes after each row reduction iteration. Note that the image is originally 178 rows.

C. MATLAB changepoint algorithm

The insect and bee datasets each had an algorithm. This is because the insect algorithm was a lot more sparse in insects.

In the insect algorithm, the remaining rows, if any, were each put into the `cwt` function once again. The absolute value of the resulting 71x1024 matrix was calculated and the maximum value of the matrix was also found. Because we are looking for peaks and want to maximize the difference between the peaks and the rest of the signals, a threshold of $0.225 * \textit{maximum value of the matrix}$ was established. Any pixel less than that threshold was set to 0. The matrix was also cropped down to 40 rows, and the remaining matrix was normalized to values between 0 and 1.

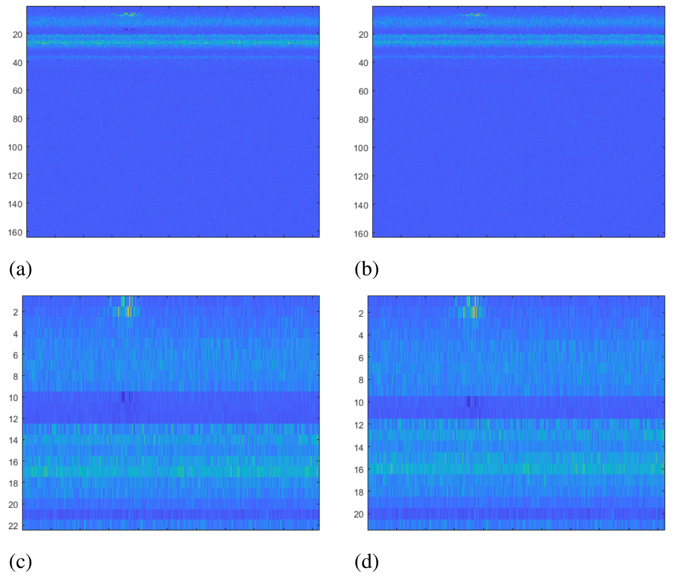


Fig. 5. Image after a) removing empty rows, b) removing the hard target, c) removing rows with no change, and d) removing rows with no frequency

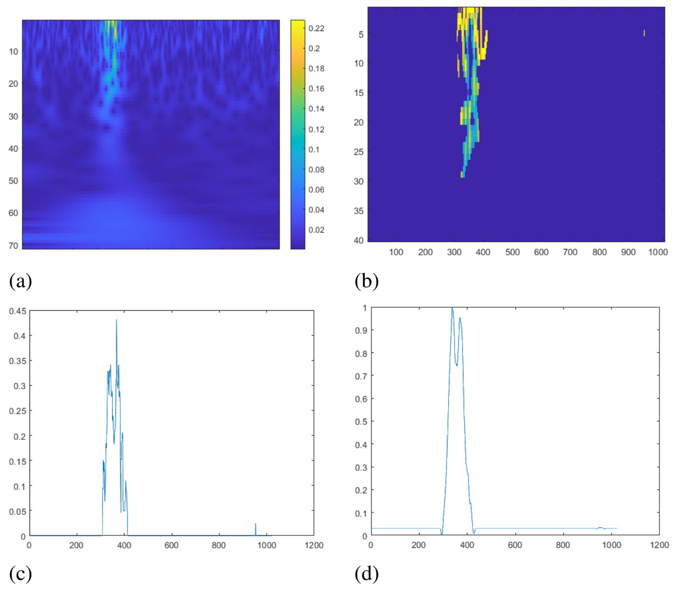


Fig. 6. a) Scalogram of row 2 of the processed image, row 7 of the original image, b) processed scalogram, c) signal of the average of each column of the processed scalogram, d) smoothed and normalized signal

The average of each column of that matrix is stored into a vector, and then smoothed and normalized once again. This final vector is then put into the `findchangepts` function. **Fig. 6** shows the magnitude scalogram and the average siganals for a row with an insect in it. Comparing this to the scalogram in **Fig. 7**, the insect peak can be identified and the maximum value of the scalogram is much larger.

```
ipt = findchangepts(tempSignal,'Statistic','mean',
            'MinThreshold',3.9);
```

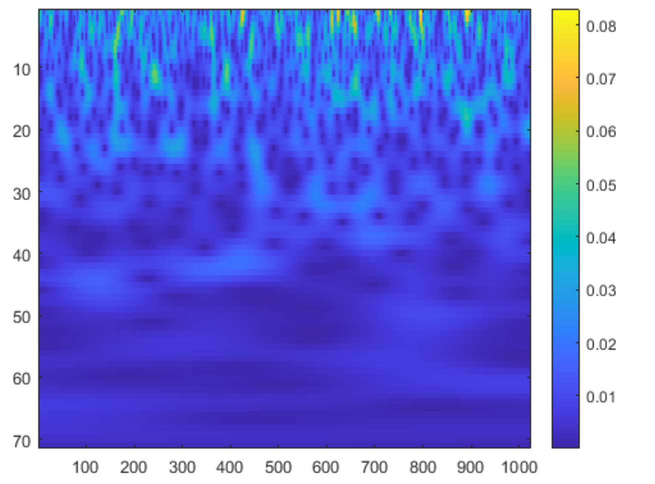


Fig. 7. Scalogram of a row that does not contain an insect

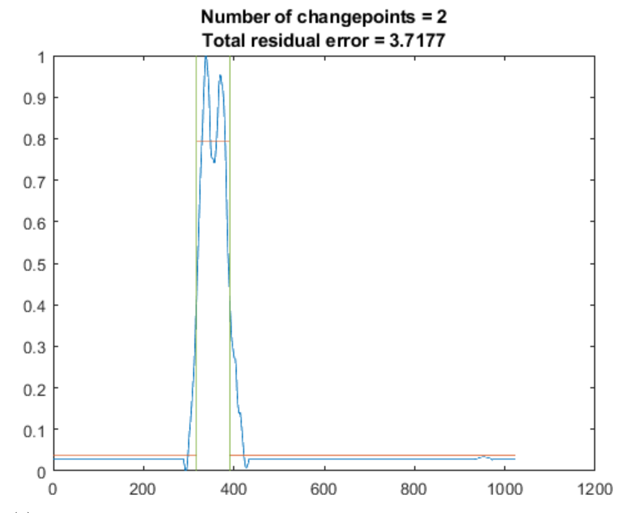
takes in the vector `tempSignal`, which is the smoothed average of the scalogram, looks at the change in mean (`'Statistic','mean'`), and has a minimum threshold of 3.9 (`'MinThreshold',3.9`). If the function finds changepoints in the signal those changepoints are stored into `ipt`. The original signal is then smoothed, normalized, and put into

```
    ipt2 = findchangepts(normalize(smoothdata
(img(row,:),'sgolay',20),'range'),'Statistic',
           'mean','MinThreshold',3);
```

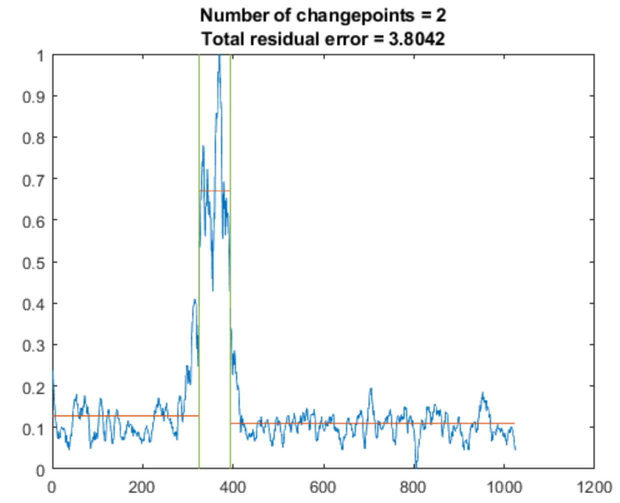
This looks at the changepoint in the smoothed and normalized row signal. The minimum threshold is decreased to 3 because there is more noise in the row signal. If no changepoints are found, there are no insects in the image. If there are changepoints, the scalogram changepoints and the original row changepoints are compared. If they are within 21 pixels of each other, which is less than a 2% difference, then the scalogram changepoint is kept. Otherwise, it is discarded. The scalogram is chosen over the original row changepoint because there are some signals that have a strong frequency in them, but are covered by noise and thus more detectable by the wavelet transform. **Fig. 8** shows the changepoint detection on the scalogram average signal and the original row signal, respectively.

After the changepoints have been established, the location of the insect is found by finding the brighter spots of the scalogram signal. However this process only considers a maximimum of two insects. This process would need to change to accommodate more than two insects per row. The insect locations are stored in the insect position matrix that contains the row number, starting column of the insect, and ending column of the insect. Note that although the insect locations in each image have been stored, the accuracy of these locations were not tested or verified.

After the locations have been established, rows that have insect signals longer than 605 pixels or shorter than 15 pixels are removed from the insect position matrix. If any rows



(a)



(b)

Fig. 8. The a) first and b) second changepoint function iterations

remain, then there is an insect in the image.

The bee algorithm is relatively the same. All of the preprocessing steps remained the same. The changepoint functions had reduced thresholds because there were significantly more insect signals that were barely visible. The first changepoint threshold was reduced from 3.9 to 2, and the second changepoint threshold was reduced from 3 to 0.8. After the changepoints were verified, the maximum length of an insect signal was increased from 605 to 700.

The insect rows were then grouped together by insect, and then the middle column of the signals was found. That column was then normalized and put into the `islocalmax` function with a minimum prominence of 0.6. If one of the peaks was located at one of the rows of the insect signal, it was considered an insect. Otherwise, it is not an insect. **Fig. 9** shows the peak that `islocalmax` identified in the column signal.

Fig. 10 and **Fig. 11** show a general workflow of the changepoint detection algorithm development and the post-

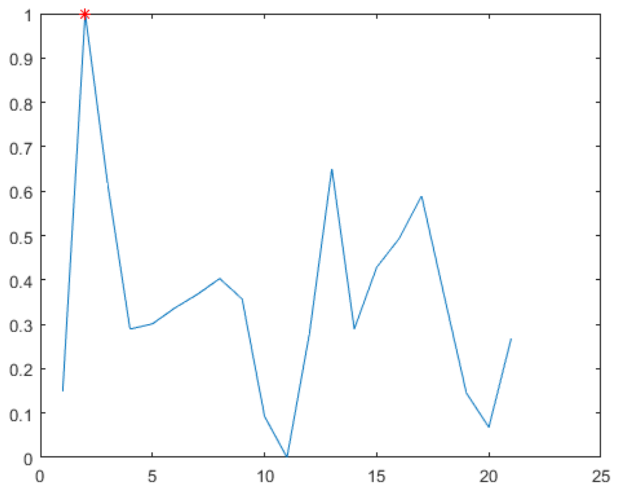
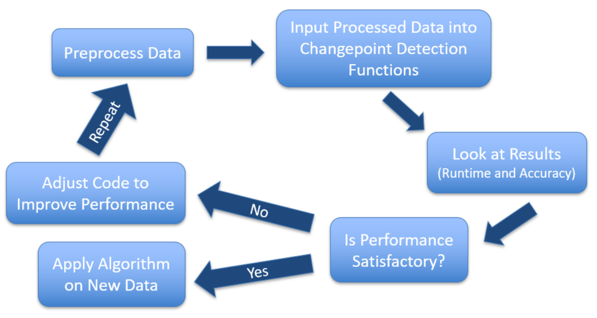


Fig. 9. Identified local maximum of the middle column of the insect



Preprocess Data
Input Processed Data into Changepoint Detection Functions
Look at Results (Runtime and Accuracy)
Is Performance Satisfactory?
No
Adjust Code to Improve Performance
Repeat
Yes
Apply Algorithm on New Data

Fig. 10. Workflow for developing the changepoint detection algorithms

detection filtering of the changepoints, respectively.

III. Results

A. Insect Data

The insect algorithm took around 14474 seconds, or about 4 hours to run through all of the images. The algorithm had a 97.6% accuracy. **Fig. 12** shows the resulting confusion matrix. Out of the 173 images, the algorithm detected 123 of them, or 71.1% of the insect images. There were also 50 false negatives (insects the algorithm did not detect) and 195 false positives

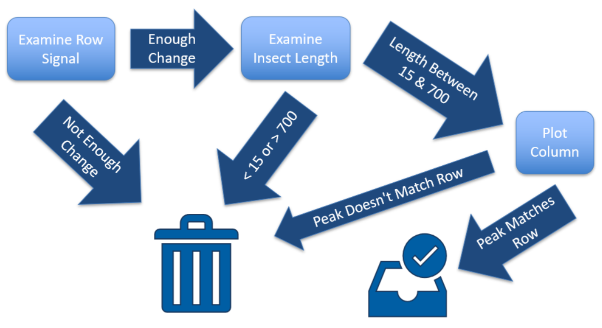


Fig. 11. Post processing to determine which changepoints are valid and which ones should be discarded

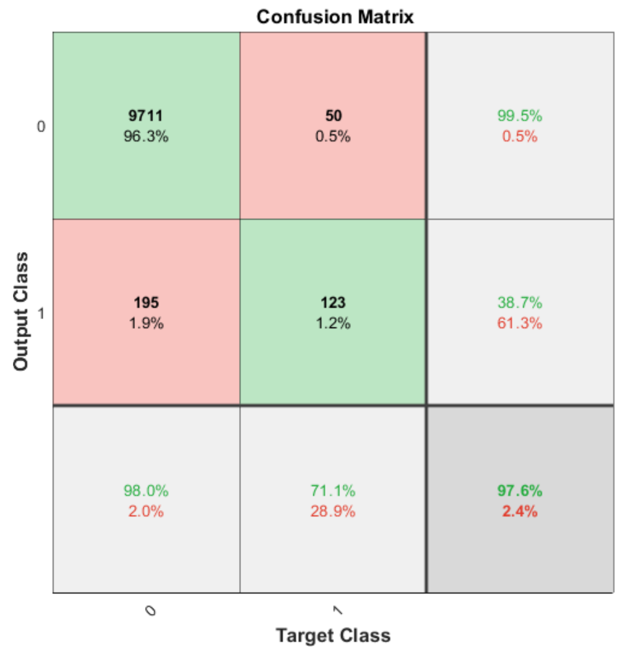


Fig. 12. Confusion matrix of the insect algorithm on the insect dataset

(images that the algorithm thought had insects in it, but did not actually have any).

B. Bee Data

The bee dataset was tested on both the insect algorithm and the bee algorithm. **Fig. 13** shows the confusion matrix from the bee algorithm. The bee algorithm took around 8066 seconds, or about 2 hours and 15 minutes to run through all of the images. The algorithm had a 73.3% accuracy. Out of the 1309 images, the algorithm detected 857 of them, or 65.5% of the insects. There were 452 false negatives and 651 false positives.

The insect algorithm on the bee dataset took around 10419 seconds, or 2 hours 50 minutes to run. **Fig. 14** shows the confusion matrix from that algorithm. It had an accuracy of 81.4%. Out of the 1309 images, the algorithm detected 671 of them, or 51.3% of the insects. There were 638 false negatives and 132 false positives.

IV. Discussion

Although the insect algorithm on the insect dataset appears to do the best, the amount of insect images along with the significant class imbalance means that false positives and false negatives barely affect the overall performance. There are more false positives than actual identified insects. Thus when the insect algorithm was used on the bee dataset, the performance went down significantly. Another explanation for the decreased performance could be that the insect algorithm was tested and refined using the insect dataset instead of the bee dataset. The bee algorithm was then modified from the insect algorithm, but because of how late the dataset was labeled, there was no time to really refine the algorithm to work well with the bee dataset.

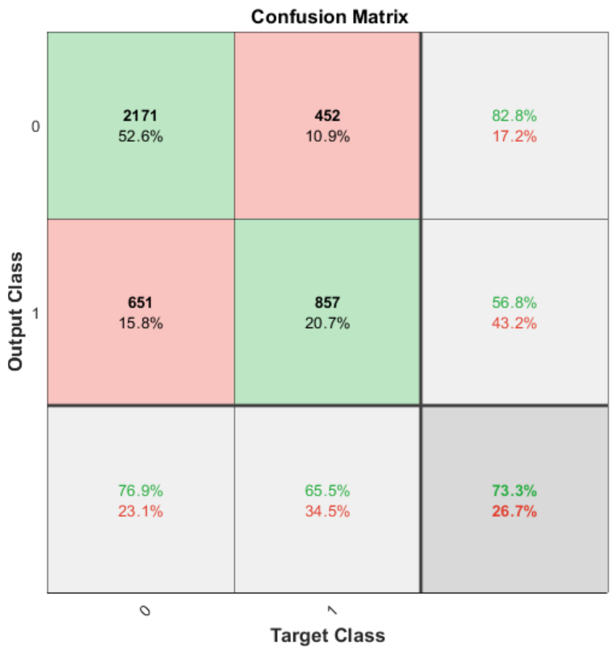


Fig. 13. Confusion matrix of the bee algorithm on the bee dataset

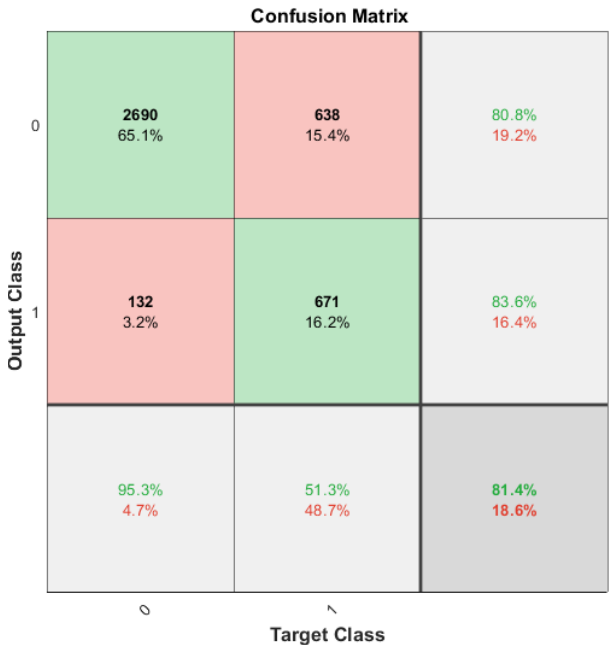


Fig. 14. Confusion matrix of the insect algorithm on the bee dataset

The amount of time to label the insects was greatly reduced. the bee dataset took 35 hours to label. This was split between 5 people and was done over the course of five days. In contrast, the slower insect algorithm on the bee dataset took 2 hours 50 minutes. Although this is already significantly faster, the time can be further reduced if code was run in parallel instead of linearly.

The algorithms tended to only be able to detect stronger insect signals. Once a signal got too weak or was in too much noise, it was overlooked. The algorithms also focused on the frequencies in each row. Although this may help with the insect detection, some noise was falsely identified to have frequency, and thus would be considered an insect.

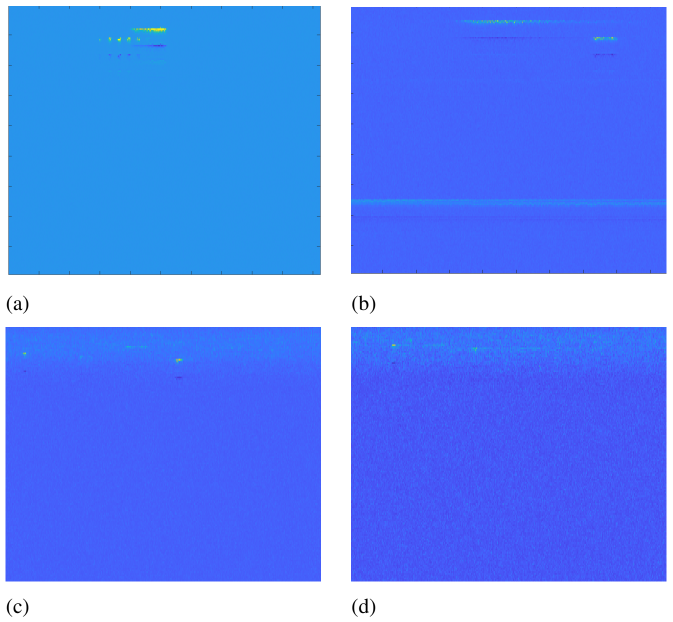


Fig. 15. Examples of all the different insect signals

In addition, the algorithms have the capability to detect the exact rows and columns the insect signal is in. Although this still needs to be majorly refined, it could be able to determine precise insect location instead of a general area.

The process of removing the rows could also be refined. The `cwt` function takes a significant amount of time, so using that in the row-removing process is slightly inefficient. An alternative process that takes less time can be used. More processing can be done to maximize the number of rows removed. The `cwt` shows how prominent a frequency is, but if there is a way to extract the location of the prominent frequency, more rows can be removed which would reduce the runtime.

The changepoint detection function could also be modified. The thresholds could be adjusted to be more or less sensitive, which can help with decreasing the number of false positive and negatives. There could also be a better way to identify which changepoint sections are insects and which ones are not.

The identified signals in the LiDAR images have the possibility of not being an insect. Since there is currently no way to guarantee whether or not a signal is actually an insect, there might be false positives in the labels created by humans. That possibility, along with the variety of different insect signals (**Fig. 15**), makes it hard to create an algorithm that can detect everything. Further development of this algorithm can look at ways to be able to identify the different varieties of insect signals.

This research used the normalized data instead of the raw data. This meant all values in the image were between 0 and 1. However normalizing the data ignores the strength of the return signal. The data that contains the amount of volts in the return signal can be a more accurate image to preprocess and analyze. This is because the data is on a static scale instead of normalizing to the minimum and maximum value of the matrix. Thus signals returned from air will always be under a certain threshold, and insect signals will almost always be above a certain threshold. Using the volts matrix can also reduce the amount of preprocessing time because of these known thresholds.

V. Conclusion

Although the performance of these algorithms were not ideal, with some more refinement they can be a viable replacement to supervised machine learning. The significant reduction in the time it takes to identify all of the images makes changepoint detection worth investigating further.

Further improvements include running changepoint detection in parallel to further decrease the time to iterate all of the data. The voltage matrix should also be used instead of the normalized matrix. This ensures that all values are on a static scale, which can help identify noise better and also use the voltages as a factor to identify possible insects in the images. There should also be more investigation to look at ways to optimize the row reduction method, and to identify all deviations of the insect signals.

The row removing process can be implemented for both supervised and unsupervised machine learning purposes. This is because the process has the capability to eliminate images with no insects. Being able to remove images without insects from the dataset has the capability to fight class imbalance when training supervised machine learning models. It can also reduce the amount of images that go on to the next step of identification in unsupervised machine learning models. Both of these instances show that row reduction has the potential to improve model accuracy and/or decrease model runtimes.

Despite the model not being accurate, there are still implications of it being useful to reduce time consuming processes. For the bee dataset, it takes about 30 seconds to determine whether or not there is an insect in the image. In contrast, it takes the slower insect algorithm about 2.5 seconds. So even if it takes 2 hours 50 minutes to run through all of the data, and then an additional 6 hours 40 minutes to identify 51.3% of the insect images, it is still less than half of the total time it took for humans to label the entire dataset. Essentially it is 9.5 hours in contrast to 17.5 hours, which almost double the time. Thus the algorithm is still able to save some time, despite some inaccuracies. Using this and its performance on the datasets, the algorithm can still predict whether a scanned area has insects or not. Thus when conducting field tests in different areas, the model can predict whether or not there is a significant number of insects and which areas have relatively more insects. Knowing the relative accuracy of these models can also help estimate the actual number of images in the dataset that contain an insect.

References

[1] Trevor Vannoy, et al., "Machine learning-based region of interest detection in airborne lidar fisheries surveys," J. Appl. Rem. Sens. vol. 15, no. 3, July 2021, doi: 10.1117/1.JRS.15.038503.

[2] T. C. Vannoy, T. P. Scofield, R. D. Logan, E. M. Rehbein, J. A. Shaw, and B. M. Whitaker, "Detection of insects in class-imbalanced LiDAR field measurements," 2021 IEEE International Workshop on Machine Learning for Signal Processing, Oct. 25–28, 2021, Gold Coast, Austrailia.

[3] Yoav Freund, et al., "Experiments with a new boosting algorithm," in *icml*. Citeseer, 1996, vol. 96, pp. 148–156.

[4] Cris Seiffert, et al., "RUSBoost: A hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010.

[5] T. C. Vannoy, T. P. Scofield, R. D. Logan, E. M. Rehbein, J. A. Shaw, and B. M. Whitaker, "Dataset for insect lidar supervised classification," Zenodo, 13-Sep-2021. [Online]. Available: https://zenodo.org/record/5504411.YtHXHzfMKUk

[6] Tauc, M. J., Fristrup, K. M., Repasky, K. S., and Shaw, J. A., "Field demonstration of a wing-beat modulation lidar for the 3d mapping of flying insects," OSA Continuum 2, 332 (Jan. 2019).

[7] Hocking, T. D., Rigaill, G., Fearnhead, P., Bourque, G., "Generalized functional pruning optimal partitioning (gfpop) for constrained changepoint detection in genomic data," arXiv, 2018.