Summer 2024

# ALGORITHM DEVELOPMENT FOR ANALYSIS OF OPTICAL TIME-SERIES DATA GENERATED BY BIOMODAL OPTICAL-ELECTRICAL NANOPORE SENSOR

Yu-Shiuan Huang
*University of Texas at Arlington*

# ALGORITHM DEVELOPMENT FOR ANALYSIS OF OPTICAL TIME-SERIES DATA GENERATED BY BIOMODAL OPTICAL-ELECTRICAL NANOPORE SENSOR

by

YU SHIUAN HUANG

Submitted in partial fulfillment of the requirements

for the degree of Master of Science in Bioengineering at

The University of Texas at Arlington

August 2024

Arlington, Texas

Supervising Committee:

Dr. George Alexandrakis, Supervising Professor
Dr. Khosrow Behbehani
Dr. Salman Sohrabi

ACKNOWLEDGEMENTS

I am deeply grateful to everyone who has contributed to the completion of this thesis. First and foremost, I would like to thank Dr. Alexandrakis and Scott Renkes for their guidance, support, and immense knowledge, which have been instrumental in shaping this research. They have consistently offered their assistance, always trying to explain ideas multiple times to ensure my understanding. I also want to express my gratitude to my peers who were willing to discuss the fundamentals of signal processing and algorithm development with me. Furthermore, I owe a debt of gratitude to my family, whose unwavering support and encouragement throughout my master's degree have been invaluable. Without their financial and emotional support, I would not have been able to complete this degree.

Once again, I extend my most profound appreciation to all those who have contributed to the successful completion of this project and my journey at the University of Texas at Arlington.

August 9th, 2024

ABSTRACT

Algorithm Development for Analysis of Optical Time-Series Data Generated by Bimodal Optical-Electrical Nanopore Sensor

Yu-Shiuan Huang

The University of Texas at Arlington, 2024

Supervising Professor: George Alexandrakis

Nanopore biosensors have played an essential role in the scientific investigation of DNA, RNA, proteins, and other bio-analytes. Traditional nanopore biosensors detect variations in electrical current conductance caused by the analyte blocking the current passing transiently through a nano-sized aperture, i.e., the nanopore. This electrical data provides information about the amount of current blockage when the analyte is inside the nanopore and the time it takes for analytes to travel through the nanopore (translocation time). The translocation time of analytes in standard nanopore measurements is typically very short, usually tens of microseconds, limiting the accuracy of measuring the analytes' characteristics. The Self-induced Back Action Actuated Nanopore Electrophoresis (SANE) biosensor slows analyte translocation time by optically trapping analytes through the Self-Induced Back Action (SIBA) mechanism. The SANE sensor combines optical trapping and electrophoresis to trap and detect analytes. This allows us to slow down analytes and

observe them for several seconds, which is a much more extended period compared to traditional electrical nanopore techniques. The SANE sensor data acquisition hardware contained four channels: two for electrical and two for optical data. These data provide us with essential features in signal processing, such as optical step change, optical trap time, translocation time, and translocation current.

Previous developed MATLAB-based GUIs such as “EventPro: Nanopore Data Analysis App” and “AutoStepfinder” have already been developed by other researchers to process and analyze nanopore electrical data. This MSc thesis focuses on developing computer code functions for signal signature identification, processing, and quantification, specifically for the optical data obtained from the SANE sensor, as no satisfactory methods could be borrowed and used successfully from the existing scientific literature without modification. Specifically, this work focuses on the development and performance comparison between three different methods: (1) an extension of the previously published “AutoStepfinder” method developed by Loeff et al., (2) Bandpass filtering using one the Chebyshev, Elliptical, Butterworth, or Sinc filters, followed by a peak-finding algorithm, and (3) a convolutional filter method. The data processed using 'AutoStepfinder' suggests that although it could be an effective tool for processing multi-step data, it tends to detects small optical signal step changes, which are noise, as real steps. Therefore, the results from the 'AutoStepfinder' generate too many false positive data. In contrast, the results obtained from bandpass filtering, using a Sinc function, followed by a peak detection algorithm show high similarity to the manually identified optical signal step-changes, used as the

gold standard for comparisons in this work. However, limitations exist, such as the need to apply thresholds for peak detection, which prevents the identification of peaks near the baseline noise level. The convolutional method is still being developed and requires further optimization for optical signal processing.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

Appendix

# CHAPTER 1

# INTRODUCTION

## 1.1 SANE Sensor

Nanopore biosensors have played an essential role in the scientific investigation of DNA [1], RNA [2], proteins [3], and other bio-analytes. Traditional nanopore biosensors detect variations in electrical current conductance caused by the analyte blocking the current passing transiently through a nano-sized aperture, i.e., the nanopore [4]. This electrical data provides information about the amount of current blockage when the analyte is inside the nanopore and the time it takes for analytes to travel through the nanopore (translocation time). The translocation time of analytes in standard nanopore measurements is typically very short, typically tens of microseconds, which can limit the accuracy of measuring the analytes' characteristics.

The Self-induced Back Action Actuated Nanopore Electrophoresis (SANE) biosensor slows down analyte translocation time by optically trapping analytes through the Self-Induced Back Action (SIBA) mechanism [5]. The SANE sensor combines optical trapping and electrophoresis to trap and detect analytes. The double nanohole (DNH) structure focuses light on a gold layer for optical trapping, while the Solid-State nanopore (ssNP) allows analytes to pass through and be detected electrically. This allows us to slow down analytes and observe them for several seconds, which is a much more extended period compared to traditional electrical nanopore techniques.

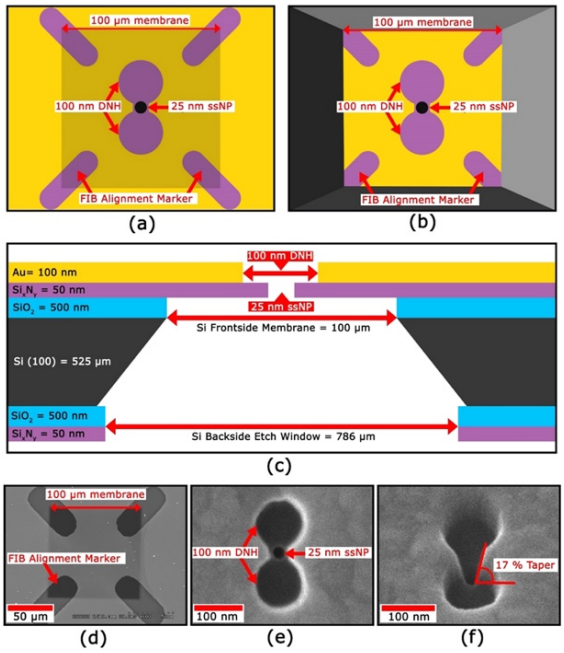


Figure 1.1: **a.** Front side view of SANE chip. **b.** Back side view of SANE chip. **c.** Cross-section of the SANE sensor chip. **d.** SEM micrograph of front side of the SANE chip before FIB drilling. **e.** and **f.** He ion microscope image of top and tilted view of milled DNH with 17% sidewall taper and a 25 nm ssNP drilled at its center [5].

The SANE sensor data acquisition hardware contained four channels: two for electrical and two for optical data. These data provide us with essential features in signal processing, such as optical step change, optical trap time, translocation time, and translocation current [5]. In Figure 1.2, regions A, B, and C show the changes as the analytes approach, get trapped, and leave the sensor. Region A shows the nanoparticle entering the sensor with increased optical transmission amplitude and a positive spike in the filtered ionic current. Region B shows the duration of time when the nanoparticles are trapped within the nanopore. Region C shows the analyte escaping from the trap towards the trans well, with decreased optical transmission amplitude and a negative spike in the filtered ionic current [5]. The difference in molecule size and geometry will result in variation in these extracted features, allowing us to characterize different analytes more accurately.

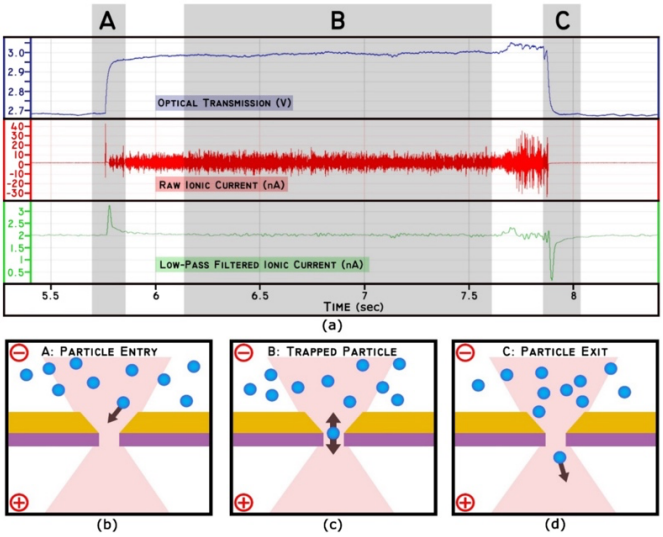


Figure 1.2: **a.** Plots of simultaneously recorded optical transmission (top, blue; V), raw ionic current (middle, red; nA) 20 Hz low-pass filtered ionic current (bottom, green; nA) versus time (sec) for the single 20 nm SiO2 nanoparticle trapped in the SANE sensor. Physical interpretation schematics for the signals recorded within gray-shaded regions A, B, and C are shown in panels (b), (c), and (d), respectively. **b.** Region A: negatively charged nanoparticle entering the DNH-ssNP under applied bias. **c.** Region B: nanoparticle trapped and bobbing inside the DNH near the ssNP mouth. **d.** Region C: nanoparticle exiting the optical trap after the electrophoretic force dominates translocation. Figure and caption reproduced with permissions by [5].

## 1.2 Goal of this Research

The SANE sensor data recording experimental setup, with which all the data analyzed in this work were collected, is located in Dr. Alexandrakis' lab (ERB 182). The experimental setup description and the experimental protocol for data acquisition has been described previously (Figure 1.3) [5]. The above-mentioned optical data include optical transmission, shown in Figure 1.2.a, and optical reflection data, which should exhibit the opposite amplitude change. The optical transmission data consist of the transmitted light and some scattered light that is collected by the condenser lens and photodiode (Figure 1.3). The optical reflection data usually contain cleaner

data compared to the optical transmission data because it is spatially filtered (Figure 1.3). However, signal processing using the optical transmission data is our goal, because the measurements will enable creating a compact device, as the optical transmission is collected closer to the sensor.

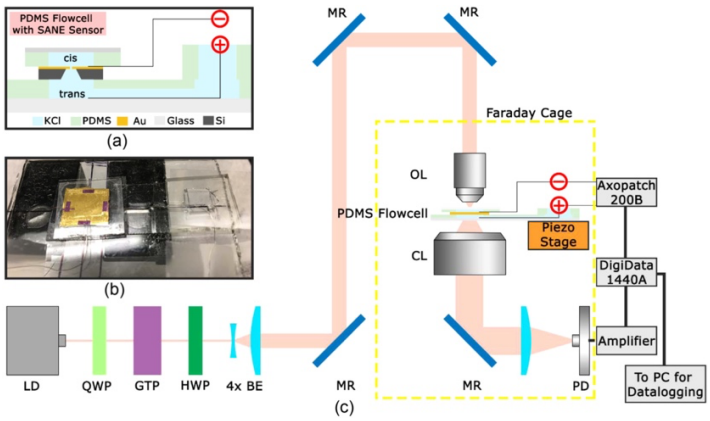


Figure 1.3: **a.** PDMS flow cell cross-sectional view with SANE sensor. **b.** Image of prepared PDMS flow cell with SANE chip ready for placement on piezo-controlled stage. **c.** Complete optical setup with PDMS flow cell placement and measurement instruments. LD: laser diode, QWP: quarter wave plate, GTP: Glan-Thompson polarizer, HWP: half wave plate, 4x BE: 4x beam expander, MR: mirror, OL: Carl-Zeiss 1.3 N.A. 63x objective lens, CL: condenser lens, PD: photodiode [5].

Previous developed MATLAB based GUIs such as “EventPro: Nanopore Data Analysis App” [6] and “AutoStepfinder” have already been developed by other researchers to process and analyze nanopore electrical data [7]. This MSc thesis focuses on developing computer code functions for signal signature identification, processing, and quantification, specifically for the optical data obtained from the SANE sensor, as no satisfactory methods could simply be borrowed and used successfully from the existing scientific literature without modification. Specifically, this work focuses on the development and performance comparison between three different methods:

(1) an extension of the previously published “AutoStepfinder” method developed by Loeff et al. [7], (2) Bandpass filtering using one the Chebyshev, Elliptical, Butterworth, or Sinc filters, followed by a peak-finding algorithm, and (3) a convolutional filter method, to be described in detail. Both methods yield the same output metrics of optical trap step size and duration from the recorded time-series signals. Each algorithm reads the data from an Azure database and saves the output metrics for each data set to that same database. As the database size of the SANE sensor optical time-series data increases, the optical data outputs of multiple runs can be compiled into histograms representing the distribution of trapping duration measurements for each analyte.

# CHAPTER 2

# METHODOLOGY

## 2.1 Optical Data Processing

All data are pre-processed, including baseline correction, because each dataset has a different baseline. Baseline correction is needed to adjust the data amplitude between 0 and 1, ensuring consistency across all datasets. A median filter is used to reduce noise and unwanted spikes in the data. This filter does not blur sharp edges, allowing us to preserve step changes in the optical transmission data (Figure 2.3). Although the median filter can remove noise, the optical transmission data contains oscillations throughout the full data set that cannot be removed simply using the Fourier Transform method followed by bandpass filtering. Further signal processing methods need to be developed for the optical data obtained from the SANE sensor. The previously developed 'AutoStepfinder' GUI is initially used to identify step changes in optical transmission and reflection data. Two other methods, using a bandpass filter followed by peak finding and a convolution method, are also explored to find better signal processing techniques.

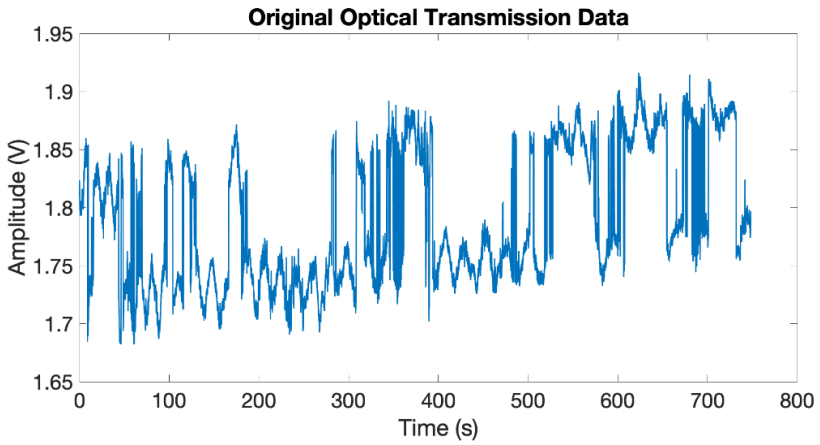


Figure 2.1: Original optical transmission data.

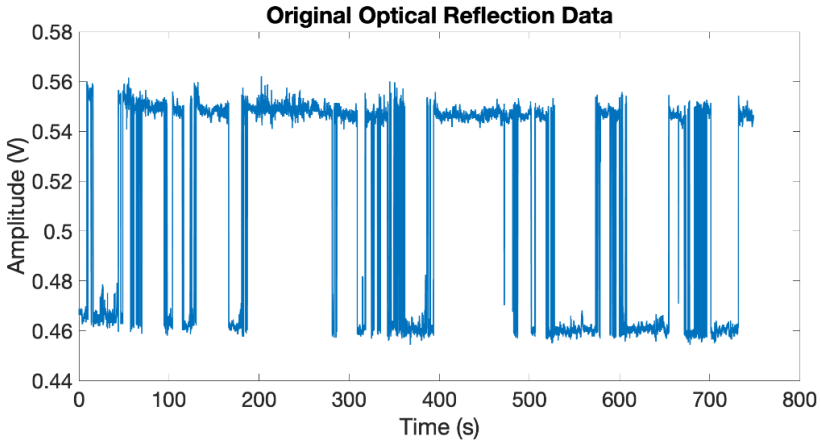


Figure 2.2: Original optical reflection data.

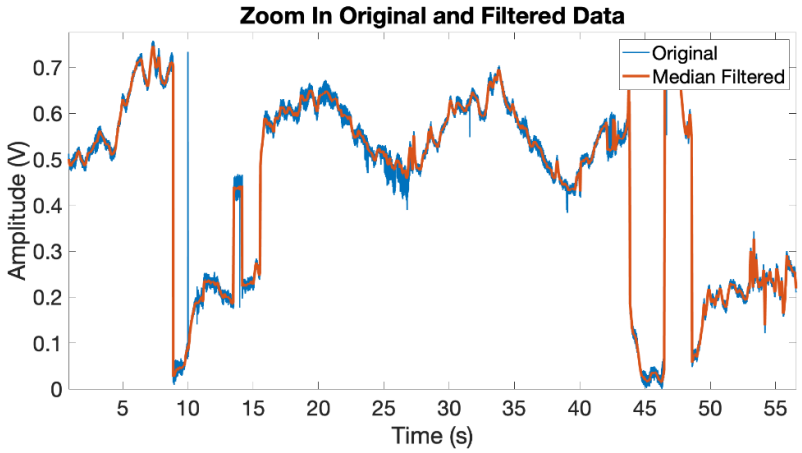


Figure 2.3: A zoom in section of the Normalized optical transmission data (amplitude between of 0 and 1) and median filtered optical data.

*2.1.1 Optical Data Processing with 'AutoStepfinder' GUI*

The 'AutoStepfinder' GUI, developed by Loeff *et al*. [7], is an app for automated step detection in data containing a variety of step sizes. Traditional methods, such as manually selecting steps or using thresholds and pairwise distribution analysis, are not effective in detecting steps close to the noise level. AutoStepfinder is designed as a first-order analysis tool that does not require prior knowledge about the location of steps or the types of signals in the data [7].

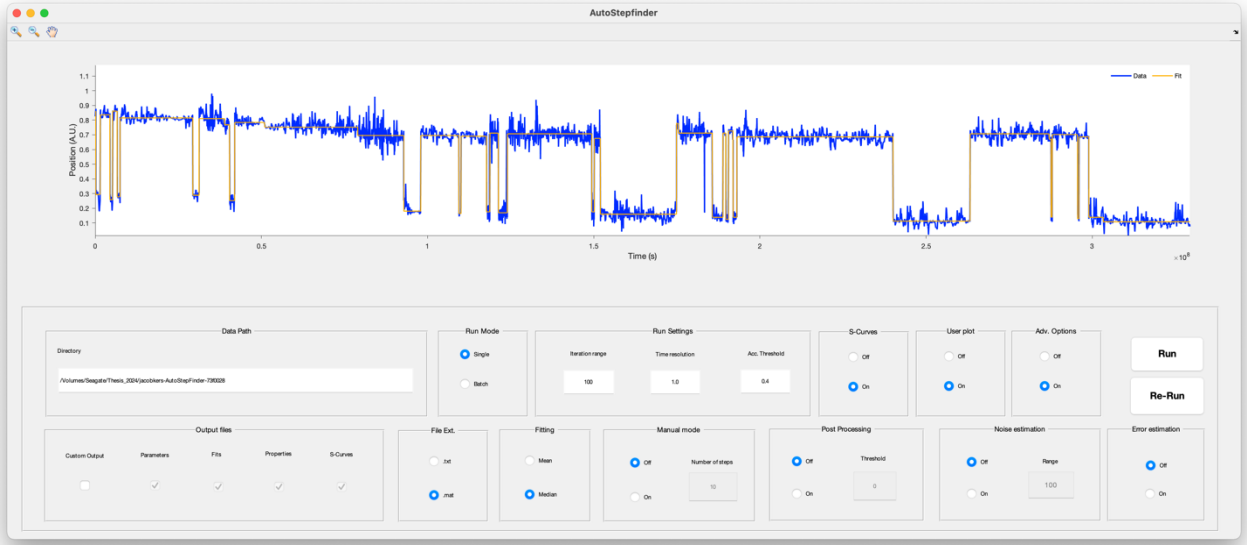


Figure 2.4: 'AutoStepfinder' user interface contains a plot for the step-fitted results along with the original input data. Filter and fitting parameters can be adjusted by users through a GUI.

The workflow of the 'AutoStepfinder' algorithm begins by importing one or multiple .txt files (Figure 2.4). Users should check and set all relevant parameters in the user interface. While running the algorithm, each iteration adds a single step to the data to minimize variance ($\sigma^2$). During each iteration, the quality of the fit is assessed by comparing the variance of the existing fit to the variance of a counter fit [7]. A counter fit involves placing steps randomly to represent a bad-fit scenario. A good fit will result in a significantly lower variance than the counter fit. A higher S-score indicates a significant difference between the counter fit and the existing fit, meaning that the existing fit captures the steps effectively. When the S-score is close to 1, it indicates either underfitting or overfitting [7]. Finally, the algorithm selects the best fit and generates a step spectrum (S-curve), allowing visualization of the quality of the fit.

$$\text{S score} = \frac{\sigma^2_{\text{counter fit}}}{\sigma^2_{\text{existing fit}}}$$

Sharp peaks in the S-curve indicate that the data fits well, while a flat curve indicates a poor fit [7]. A clear peak in the S-curve plot suggests the optimal number of iterations (Figure 2.5).

'AutoStepfinder' can handle data with various step sizes because it implements a dual-pass strategy [7]. The algorithm identifies the best fit based on the highest peak of the curve ($S_{P1}^{max}$) from the first fit, subtracts this fit from the data, and performs a second round of fitting to identify additional steps not captured in the first fit. The second fit is only accepted when it passes the acceptance threshold shown in the user interface, ensuring reliable step detection.

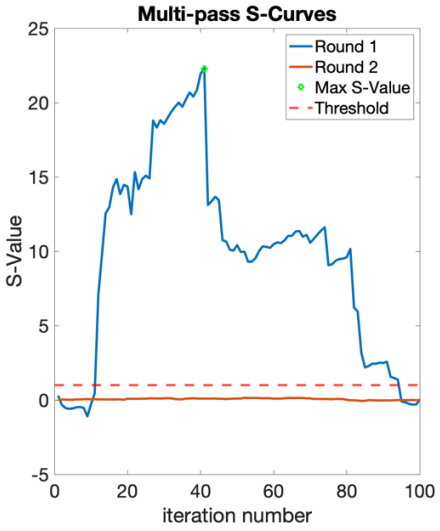


Figure 2.5: S-curve with a clear peak that indicates good fit of the data.

'AutoStepfinder' provides a good fit for larger steps compared to smaller ones. Detection accuracy decreases when the steps are small, and noise increases [7]. However, 'AutoStepfinder' can still detect steps when the signal-to-noise ratio (SNR) is as low as 0.75 [7]. The addition of a bootstrap analysis function helps improve accuracy. After finding the best fit, the algorithm resamples each segment many times to accurately determine the position of the steps, providing a 95% confidence interval for step sizes and step times.

We used 'AutoStepfinder' to process both optical transmission and reflection data. Due to the large datasets, we needed to separate them into four sections to allow 'AutoStepfinder' to save the fitted data. The fitted data from the same datasets will be post-processed to connect it back together. The optical reflection data outcome showed better results than the optical transmission

data due to the abovementioned oscillations noises. These oscillations can cause 'AutoStepfinder' to mistakenly identify them as steps, as shown in the next chapter.

*2.1.2 The Method of Bandpass Filtering Followed by Peak Finding*

The data output from the SANE sensor is loaded into MATLAB using the 'abfload' function. This function allows us to obtain the data, the total length of the data recording, the sampling interval (which allows us to calculate the sampling rate if it is unknown), and the memory requirement for uploading the data set. We processed the optical transmission and reflection data to compare the output in the result chapter. The optical data loaded will be defined as a variable for further processing. As mentioned, all data is pre-processed for baseline correction and median filtered.

The oscillation in the baseline still appears throughout the entire dataset after pre-processing steps. We manually selected data points to compute the possible frequency, and the manually selected wavelength of the oscillation averaged about 20 seconds (Figure 2.6), which is significantly above typical trapping durations of few seconds, or less. The Fast Fourier Transform (FFT) was computed to identify prominent frequencies that could be used as cutoff frequencies for designing the bandpass filter. However, there was no clear peak when we combined all the FFT data from 13 datasets (Figure 2.7).

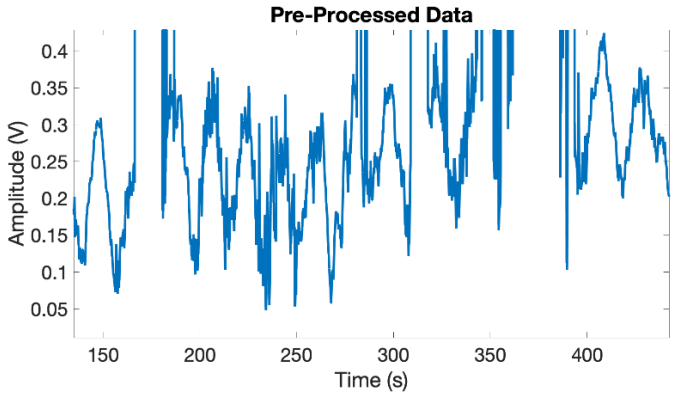


Figure 2.6: The oscillation shown in the pre-processed.

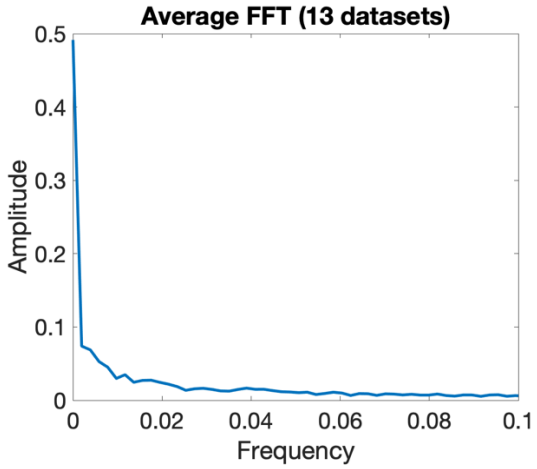


Figure 2.7: Average of 13 FFT datasets.

We selected multiple bandpass filters (Figure 2.8), including Butterworth (Figure 2.9), Sinc (Figure 2.10), Elliptic (Figure 2.11), and Chebyshev filters (Figure 2.12), to compare if the baseline oscillation was reduced or if there were significant features that could be extracted from the processed data. The step-like characteristics of the data are a signature of optical data and represent event translocation times. These chosen filters allow us to preserve the sharp step-changing edges in the optical step-change signals.

Comparing all the bandpass data, we observed peak features in the output of the Sinc bandpass filter where the filter did not uniformly attenuate frequencies across the entire spectrum (Figure 2.10). We set the low cutoff frequency at 0.039 and the high cutoff frequency at 0.04 for all data sets. These limits were determined empirically. Adjustments to the cutoff frequencies enabled us to remove noise without over-processing the data. This narrow range of cutoff frequencies restricts the bandwidth, causing ringing effects that allow only a narrow band of frequencies to pass, thereby preserving the location of peaks at sharp transitions. As shown in Figure 2.13, the start and end points of the event resulted in significant peaks. These peaks were evident and exceeded a certain amplitude threshold, making them a substantial feature for

detecting the time locations. The 'findpeaks' MATLAB function was used in our code to locate the start and end times of the events, allowing us to calculate the duration of the trapping events.

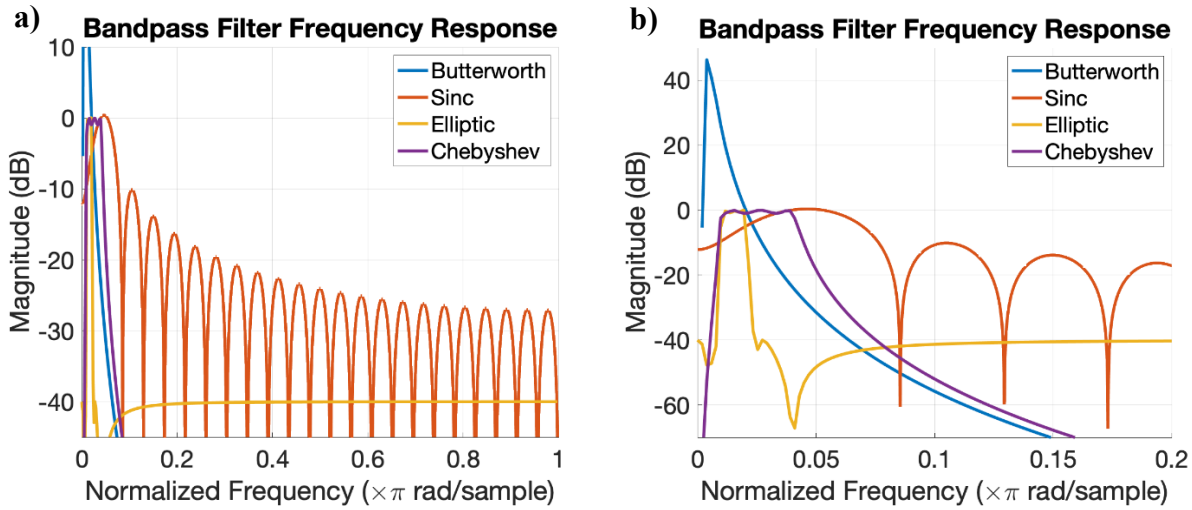


Figure 2.8: a) Frequency response curves of Butterworth, Sinc, Elliptic, and Chebyshev bandpass filters. b) Zoomed-in region providing detailed information at lower frequencies.

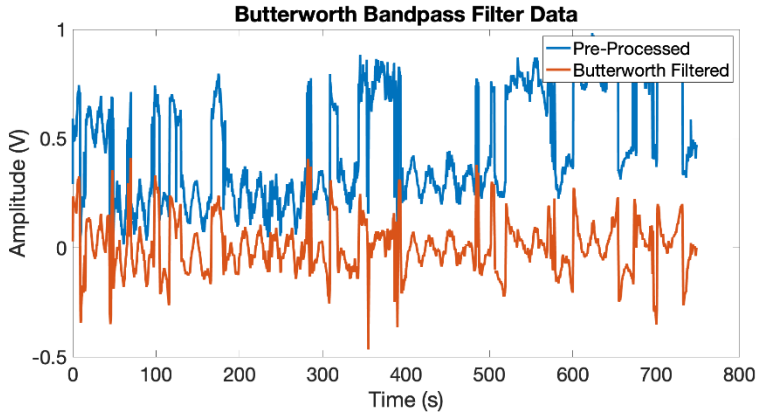


Figure 2.9: Pre-processed optical transmission data compared with Butterworth bandpass filtered data.

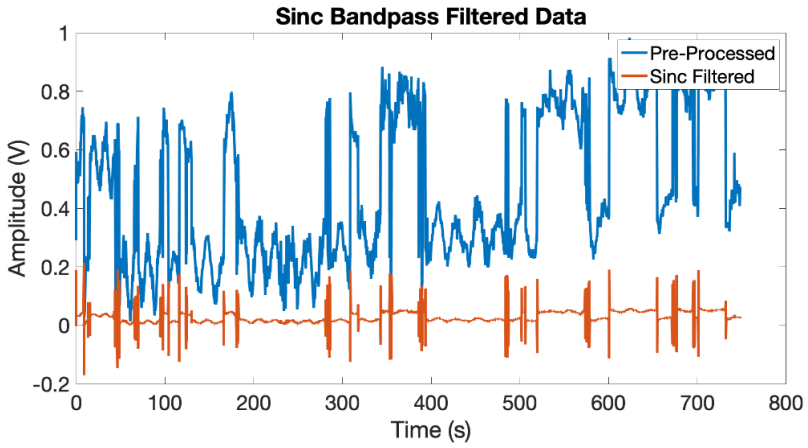


Figure 2.10: Pre-processed optical transmission data compared with Sinc bandpass filtered data.

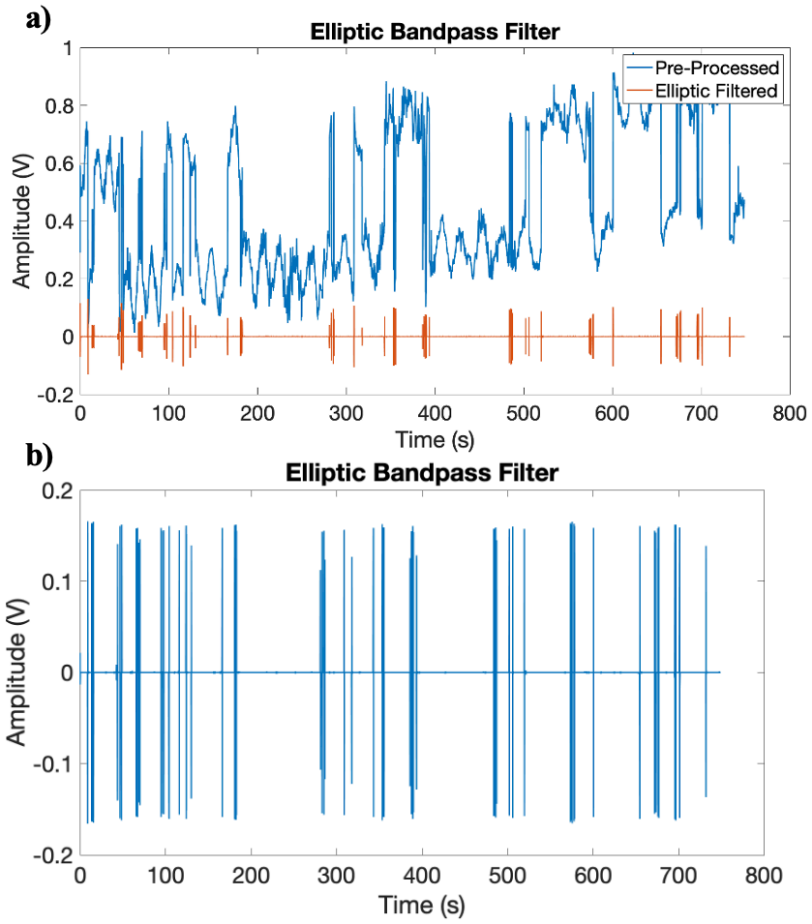


Figure 2.11: a) Pre-processed optical transmission data compared with Elliptic bandpass filtered data. b) Elliptic bandpass filtered data.

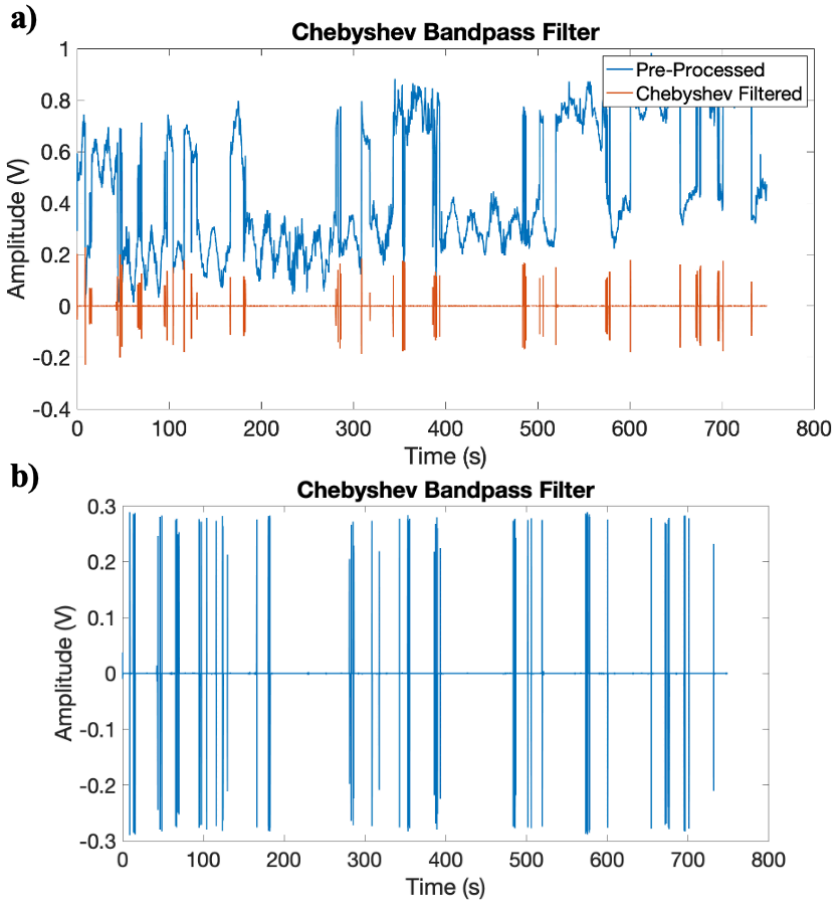


Figure 2.12: a) Pre-processed optical transmission data compared with Chebyshev bandpass filtered data. b) Chebyshev bandpass filtered data.

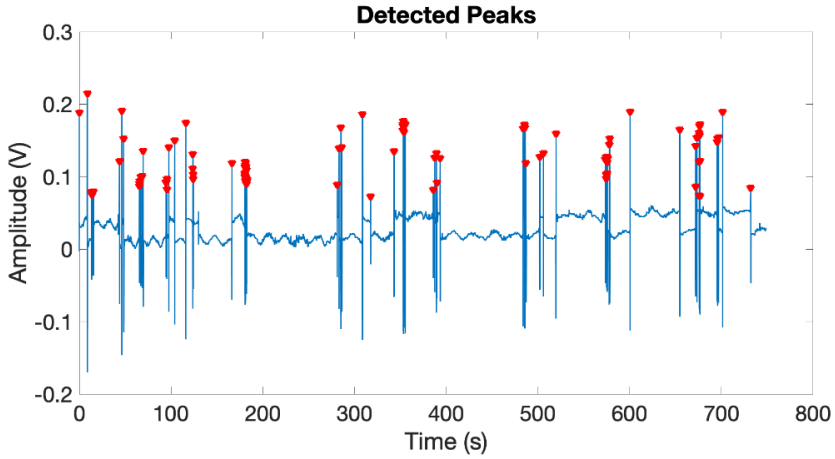


Figure 2.13: Event start, and end time points (peaks) detected from the Sinc bandpass filtered data.

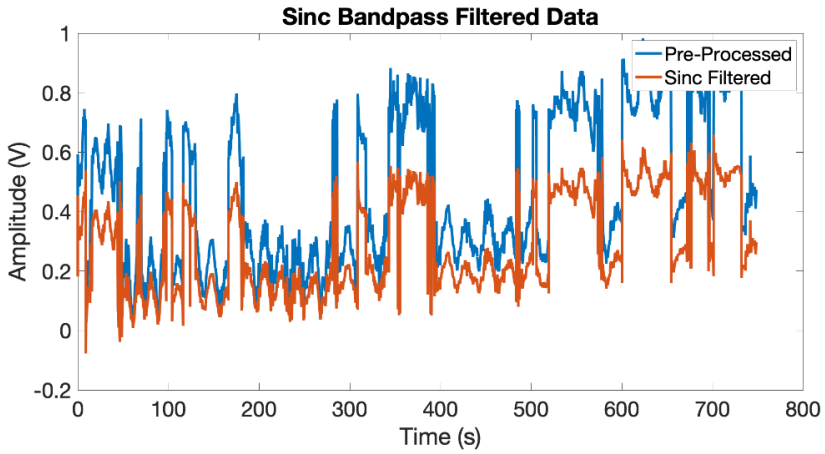


Figure 2.14: The Sinc bandpass filtered data with the low cutoff frequency set at 0.03, and the high cutoff frequency set at 0.035.

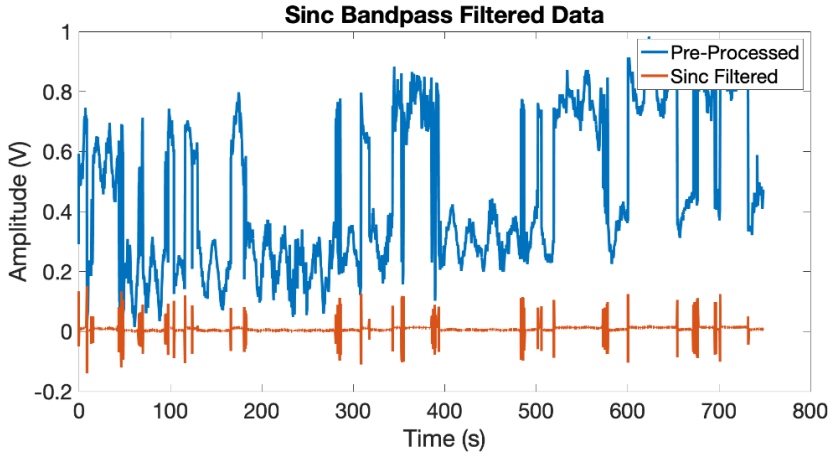


Figure 2.15: The Sinc bandpass filtered data with the low cutoff frequency set at 0.04, and the high cutoff frequency set at 0.045.

However, there were instances of under-detection and over-detection due to the threshold set within the 'findpeaks' function (Figure 2.16). Extra data points were detected if the threshold was too low. The threshold was set to 0.07 V, so the amplitude of peaks higher than 0.07 V were recorded. Users can adjust this threshold when the amplitude within the translocation duration for an event is higher than 0.07 V to ensure accurate results. The average duration between data points

detected using the peak-finding function is less than 3 milliseconds. We implemented a while loop to remove data points closer than three milliseconds apart to address the over-detection issue.

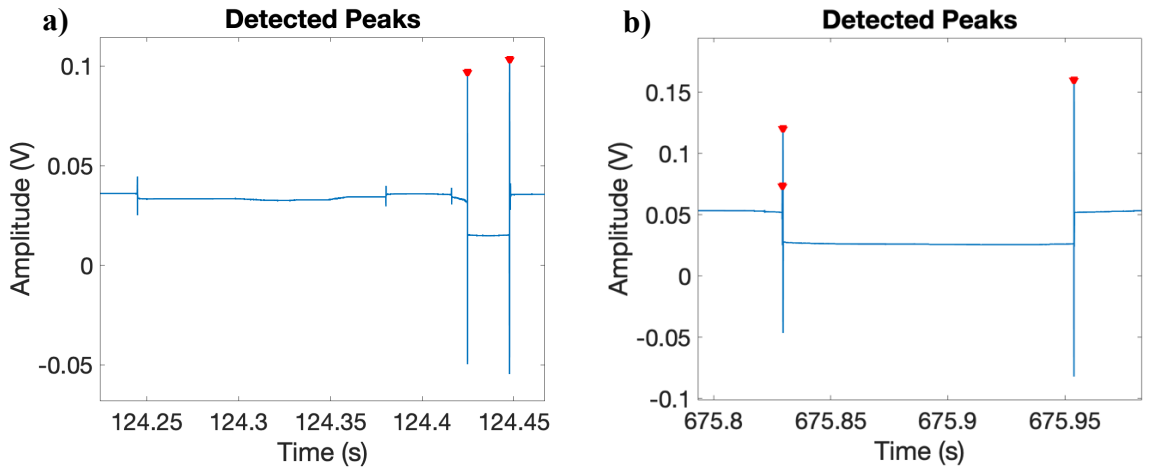


Figure 2.16: a) Under-detected and b) over-detected peaks in the data due to the threshold set for within the 'findpeaks' function.

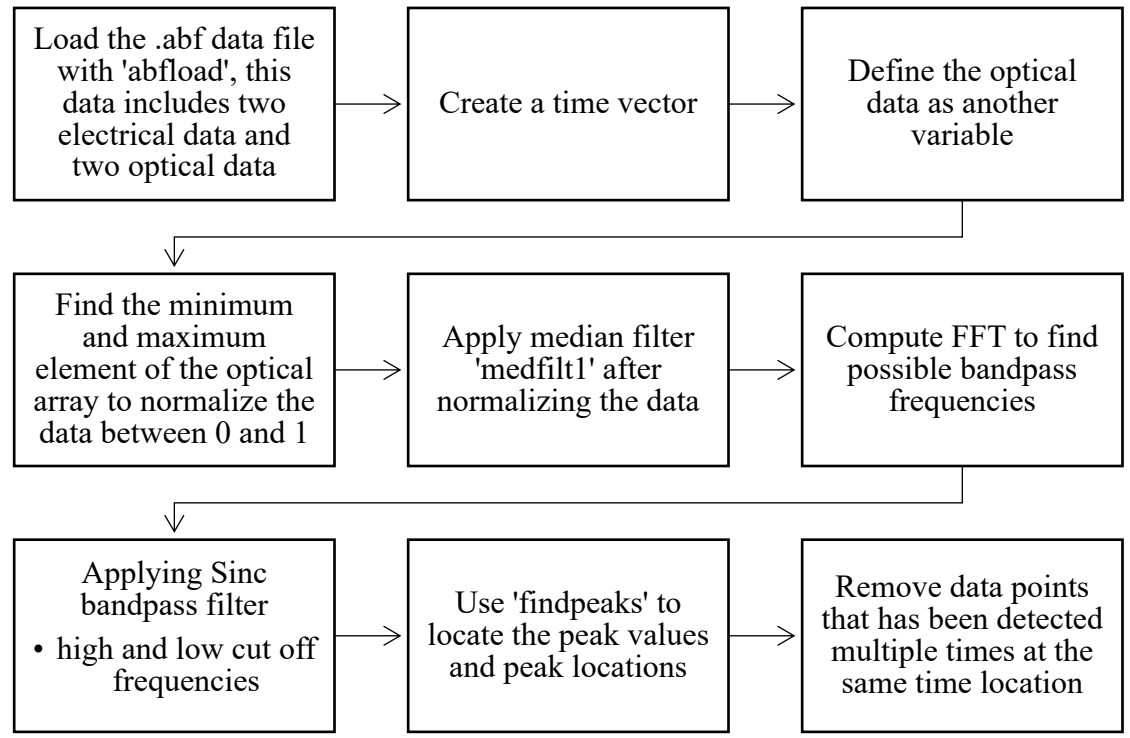


Figure 2.17: Flowchart of pre-processing the optical data with bandpass filters to remove unwanted noises.

*2.1.3 The Convolutional Filter Method*

One key feature of the optical data compared to the electrical data is the pedestal-like wave recorded from the SANE sensor (Figure 2.1). We decided to experiment with a convolutional filter method to better recognize these pedestal-like waves without visualizing them through a plot and reduce the assumptions made while processing the data. Cross-correlation is commonly used to measure the similarity between two signals. In simpler terms, the convolutional method is like a Barker code, which is employed to detect known sequences between the sender and receiver in telecommunications [8]. In our case, the duration of the event is variable. A cross-correlation was performed between a reference wave and the optical signal to identify the time when the two signals were best fitted, enabling us to determine the starting time and duration of the event. The optical signal ($\mathrm{x}[n]$) and the generated reference signal ($\mathrm{y}[n]$) are not equal lengths, so $\tau$ represents the time lags needed to align the two signals during cross-correlation.

$$\mathrm{R_{xy}}[\tau] = \sum_{\mathrm{n}} \mathrm{x}[\mathrm{n}]\mathrm{y}^{*}[\mathrm{n} - \tau]$$

Time lag ($T$) is the maximum value of the cross correlation of the two signals, where the two signal is best fitted.

$$\mathrm{T} = \arg\max\left|\mathrm{R_{xy}}[\tau]\right|$$

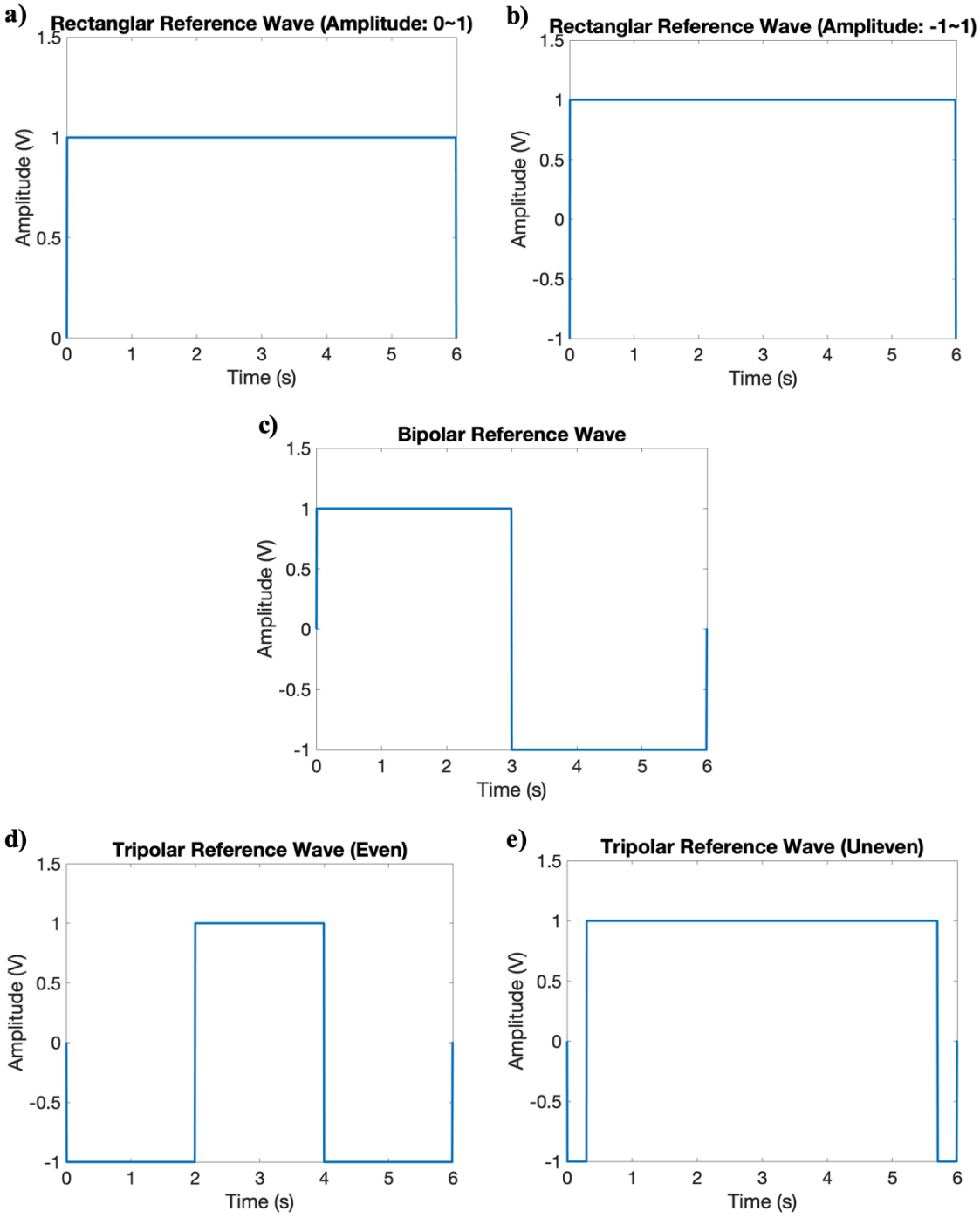


Figure 2.18: Various types of reference waves: a) a rectangular reference wave with an amplitude of 1; b) a rectangular reference wave with an amplitude of -1 to 1; c) a bipolar reference wave; d) a tripolar reference wave with three event sections; e) a tripolar reference wave with smaller first and third sections compare to the middle section.

Multiple types of reference waves were tested, including rectangular, bipolar, and tripolar waves (Figure 2.18). The tripolar reference wave was generated in a for-loop with a changing duration to convolve with a window of the optical data, which is pre-processed for baseline correction and median filtering as described before. The window length was set to 6 seconds to reduce the possibility of multiple events appearing within the same window. The more events that appeared in the same window, the more peaks were generated in the plot of duration and the gradient of the max correlation plot, reducing the clarity of the duration peak features. This was because every start and end edge of an event could be seen as a start and end edge. For example, if there are two events, the code might detect the start of the first event and the end of the second event as a single square wave, leading to an incorrect duration output (Figure 2.19). The problem with a short window length is that it can cut an event into different windows.

Figure 2.19 shows a window of data that contains two events and another plot of various durations plotted against the gradient of the max correlation, which will be discussed later in this section. If we manually selected the data points and calculated the duration of the events, the durations were found to be 1.667 and 0.285 seconds. The plot on the right displays the duration versus the gradient of the maximum correlation, showing two detected peaks with durations of 1.584 and 1.926 seconds. There is another smaller peak showing 0.252 seconds. Comparing these three durations with the manually calculated durations, we observe that 1.584 and 0.252 seconds are more accurate. However, due to the small peak, the duration at 0.252 seconds is not detected. By manually subtracting the first event's starting point from the second event's ending point, we get a result of 2.035 seconds, which is close to one of the detected durations, 1.926 seconds. This result might be due to the two events being too close together, so when the window of the optical signal is convolved with the reference wave, the maximum correlation value is higher.

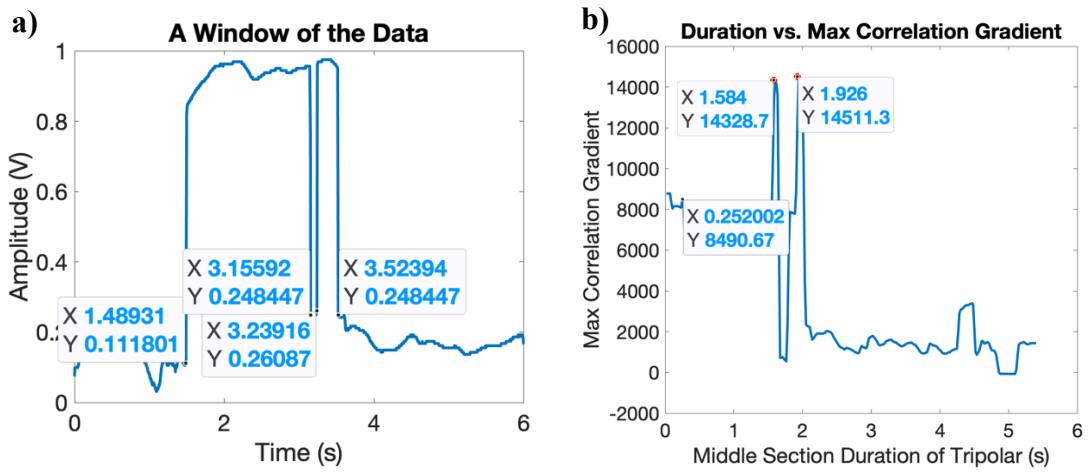


Figure 2.19: a) A window with two events. b) The plot shows various duration of the middle section of the tripolar reference wave plotted against the gradient of the maximum correlation

The final duration of the reference wave is set to match the window length for convolution with the data. The entire length of the tripolar wave was divided into three sections. We set the first and third sections of the tripolar wave to the same size, which is shorter compared to the middle section (Figure 2.18e). The shorter the length of the first and third sections, the sharper the valley will appear in the convolutional result (Figure 2.20, right). The sampling rate of all data is 500,000 Hz, which means each time step is $\frac{1}{500000} = 0.002$ milliseconds. In our code, we set the length of the reference wave to be '1:10000:length(Window)', where 'Window' represents the data of the selected window. The '1' represents the starting index, with a step size increase 10,000 for each iteration. This means that the for loop set to generate tripolar waves with different sizes will start with a duration of 0 milliseconds, and each iteration will increase by 20 milliseconds until it reaches the total duration, which is 6 seconds per window. Each generated reference wave with a different duration for each iteration will convolve with the optical signal to produce convolution results. This convolution step is the most time-consuming part of the code.

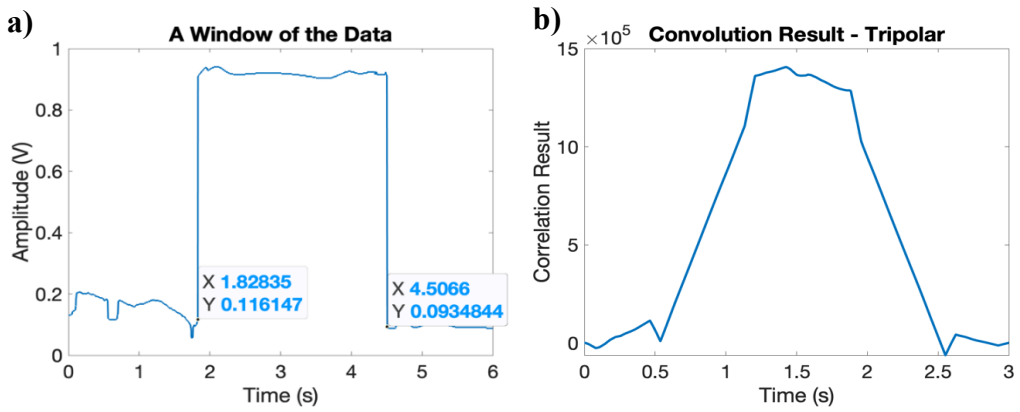


Figure 2.20: a) A single event in a window. b) The plot show the convolutional result after convolved with the tripolar reference wave.

Each iteration outputs a convolution result, which will have a maximum correlation data point stored in a separate array. As mentioned above, the duration of the reference wave changes for each iteration. The middle section of the tripolar wave should have the closest duration compared to the optical translocation duration. We plotted various durations of the middle section recorded from each iteration against the maximum correlation recorded from each convolution result (Figure 2.21a). This plot resembles a slope with changing points at different durations, where a pedestal-like shape was detected. However, the changing point of the slope needs to be more transparent before being extracted and used as the duration output.

We generated another plot with the gradient of the maximum correlation slope on the y-axis and the duration of the middle section recorded from each iteration on the x-axis (Figure 2.21b). This plot provides a clear peak when only a single event is presented in a window. This peak corresponds to the maximum correlation and provides us with a duration. As mentioned earlier, each iteration increases the size of the reference wave by 20 milliseconds until it reaches 6 seconds, which is the full size of a window. The duration output will have an error compared to the actual duration of the optical translocation because it does not increase by every time step (0.002 milliseconds).

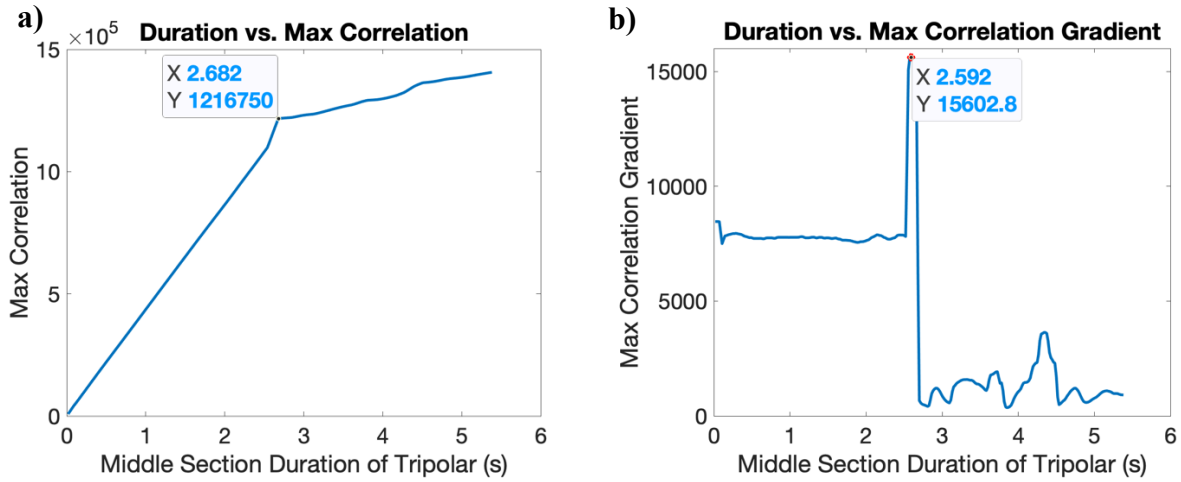


Figure 2.21: a) Different durations of the middle section of the tripolar wave were plotted against the maximum correlation stored from each iteration. b) After taking the gradient of the maximum correlation, a peak appears, indicating the best fit of the tripolar wave to the actual wave.

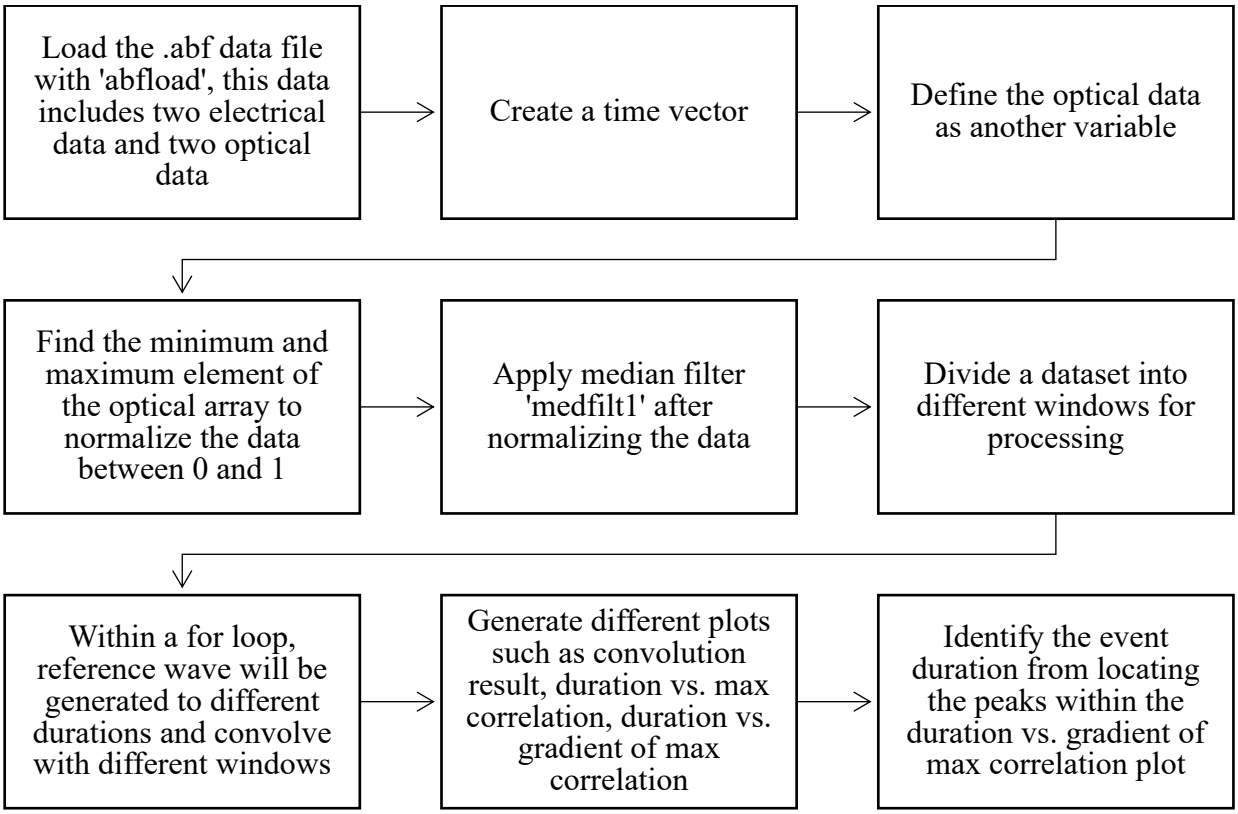


Figure 2.22: Flowchart of convolutional filter method to estimate the duration of the optical events.

# CHAPTER 3

# RESULTS

## 3.1 Data Output

Comparisons were performed for the histograms of step-change duration determined by applying the following remaining methods: 1) AutoStepfinder, 2) Sinc bandpass filtered data followed by findpeak, and 3) convolutional filtering. We processed a total of 13 datasets using these three methods and compiled the identified step-changes by each method to histograms of step duration. Both optical transmission and optical reflection data were analyzed this way. The gold standard for the output results was defined here as the optical step-changes identified manually, for which 296 steps of various durations were counted in total. The pairwise distance test (Matlab function *pdist2*) between the standard (manually selected data) and the data obtained from the above three data analysis methods were used to compare the step-counting accuracy of different methods.

### *3.1.1 Data analysis on 'AutoStepfinder' Data*

In the figures below, the blue line indicates the optical reflection and optical transmission data, while the red line represents the fitted data output from 'AutoStepfinder'. As mentioned in the previous chapters, there was more noise in the optical transmission data compared to the optical reflection data. As a result, the fitted data of the optical transmission data contains overfitted steps (Figure 3.1). It identified the baseline oscillation as steps and some of the noise within a trapping event as multiple levels. The fitted data for the optical reflection data shows a clearer fit with less overfitting (Figure 3.2). A post-processing step is needed to select the data points that are over a certain amplitude threshold, so we excluded the fitted steps of user defined plateau value. The threshold can be determined by plotting a histogram of amplitudes output from 'AutoStepfinder'

to observe the amplitude cutoffs for different step levels. For these 13 datasets, the threshold varies between 0.5 and 0.6 (Figure 3.3).

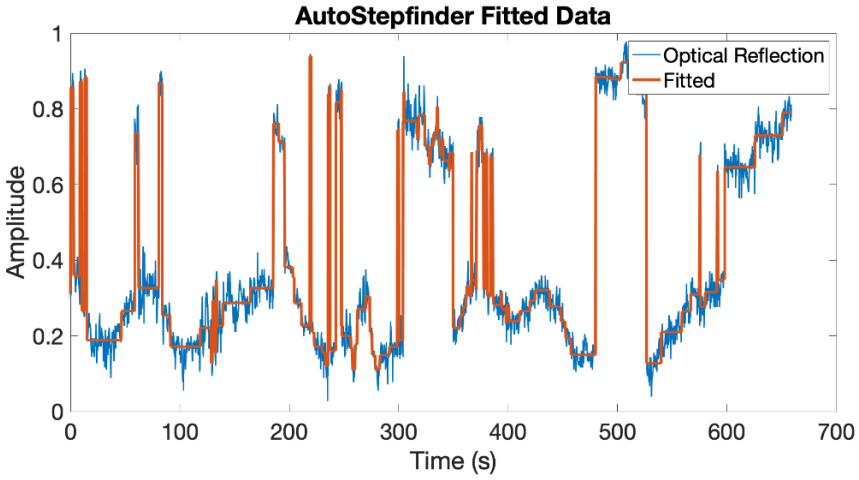


Figure 3.1: Optical transmission data plotted with the 'AutoStepfinder' fitted data.

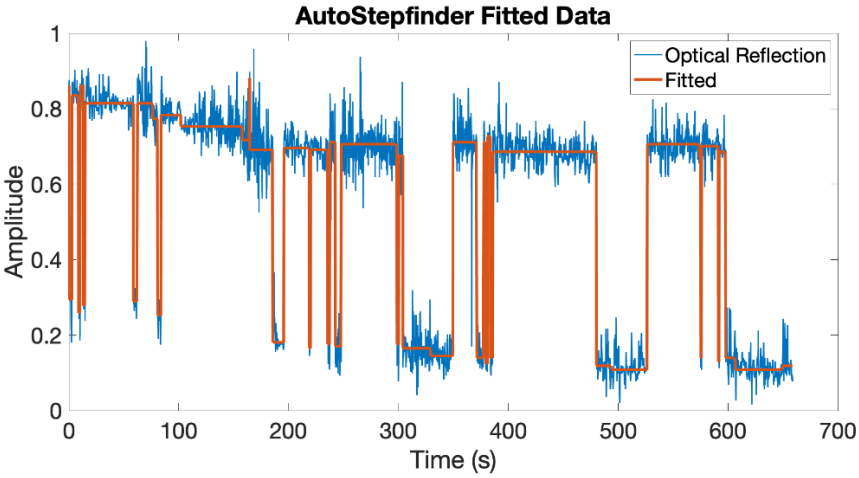


Figure 3.2: Optical reflection data plotted with the 'AutoStepfinder' fitted data.

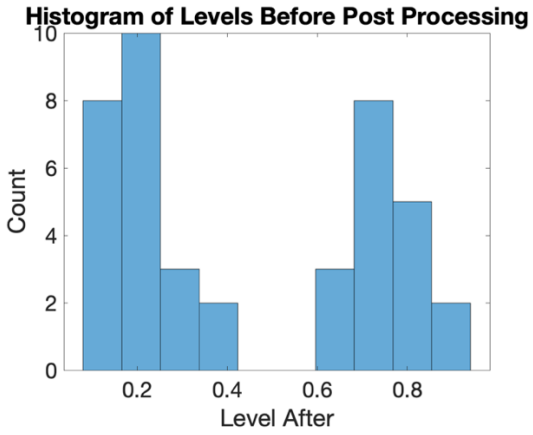


Figure 3.3: Histograms of levels before post processing for threshold selection to remove time points in baseline. 'Level After' is a variable name meaning level of the plateau after the step occurred.

*3.1.2 Data analysis on Sinc Bandpass Filtered Data*

The blue signals in Figures 3.3 and 3.5 show the optical transmission and optical reflection data, while the red signals represent the data processed with a Sinc bandpass filter. We applied the 'findpeak' function in MATLAB to the Sinc bandpass filtered data, which was able to detect peaks indicating the start and end of the trapping event (Figures 3.4 and 3.6). Importantly, in this case the results from the (concurrently acquired) optical transmission and optical reflection data were nearly identical, which demonstrates the robustness of this approach to noise.

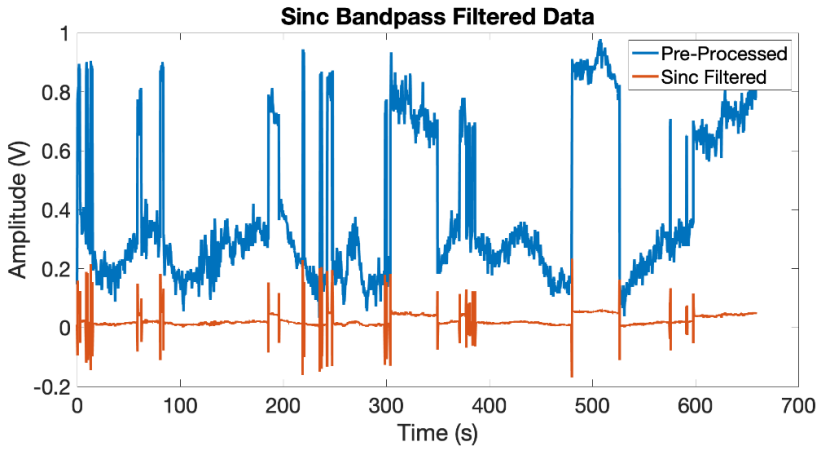


Figure 3.4: Optical transmission data plotted with the Sinc Bandpass filtered data.

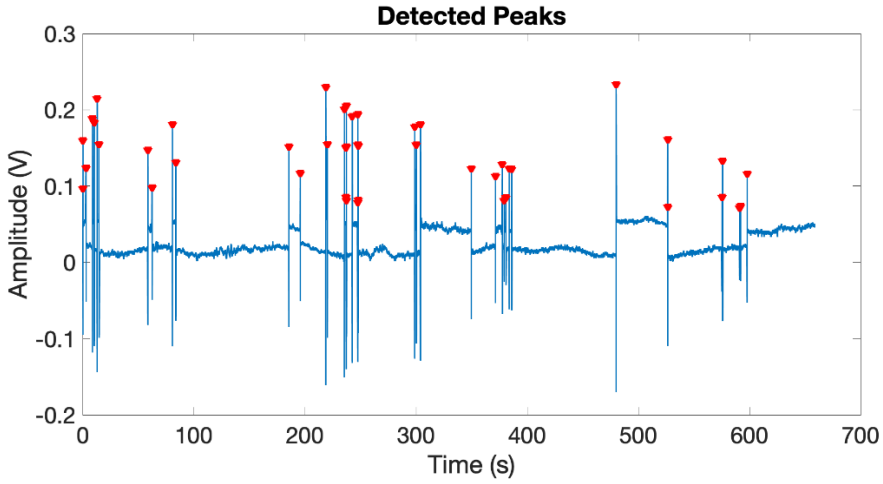


Figure 3.5: Optical transmission data event start, and end time points (peaks) detected from the Sinc bandpass filtered data.

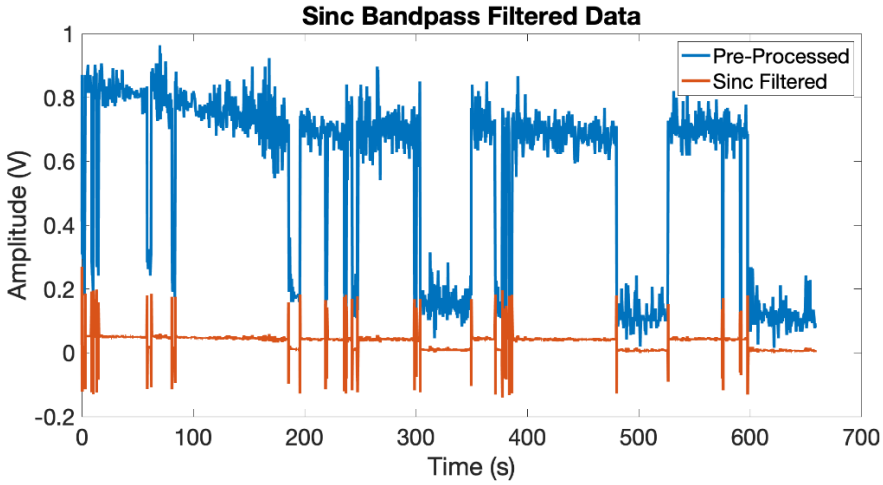


Figure 3.6: Optical reflection data plotted with the Sinc Bandpass filtered data.

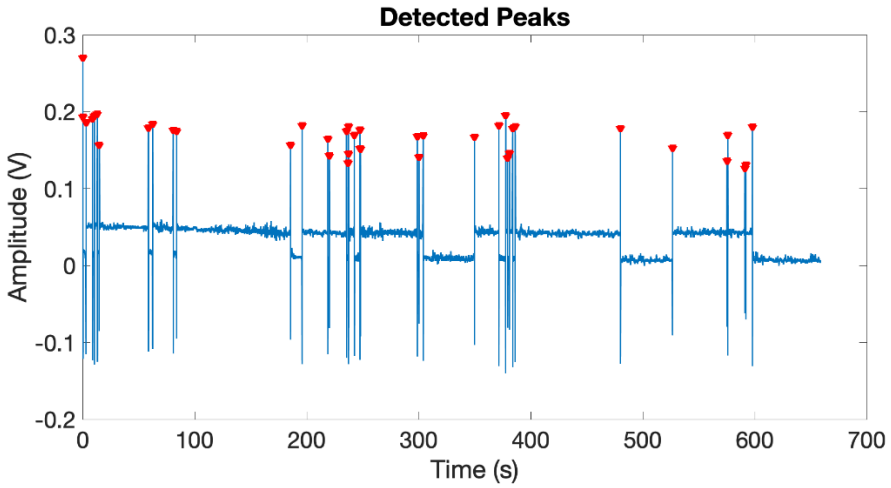


Figure 3.7: Optical reflection data event start, and end time points (peaks) detected from the Sinc bandpass filtered data.

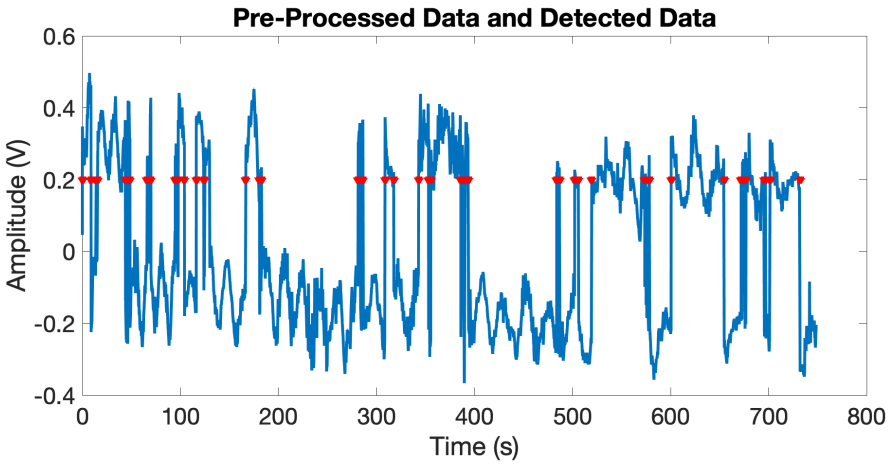


Figure 3.8: Manual identification of detected data points on pre-processed data to see if it matches the location of events.

We also used the MATLAB function 'detrend' and removed the mean of all data points to remove the slope shown in the baseline. Chapter Two mentions that the cutoff frequency range set in the Sinc bandpass filter is 0.039 to 0.04 normalized frequency, which can be identified in the FFT graph after removing data with amplitudes lower than 0.06 V (Figure 3.10). Two peaks appeared around a normalized frequency of 0.04. We also tested a broader cutoff frequency range of 0.033 to 0.043 normalized frequency, comparing it with data filtered using the original cutoff

frequency range of 0.039 to 0.04. As shown in Figure 3.11, the data filtered using the 0.039 to 0.04 normalized cutoff frequency range better removed unwanted noise and oscillations throughout the data.

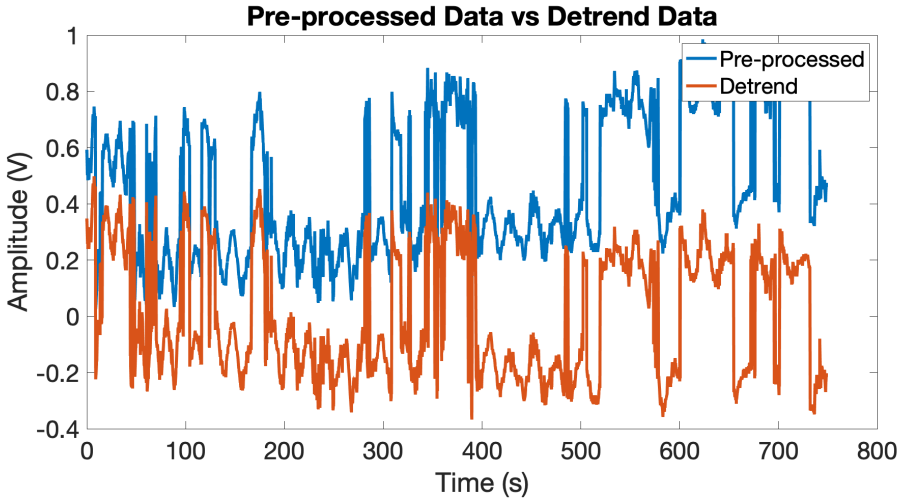


Figure 3.9: Pre-processed data compare to detrended data.

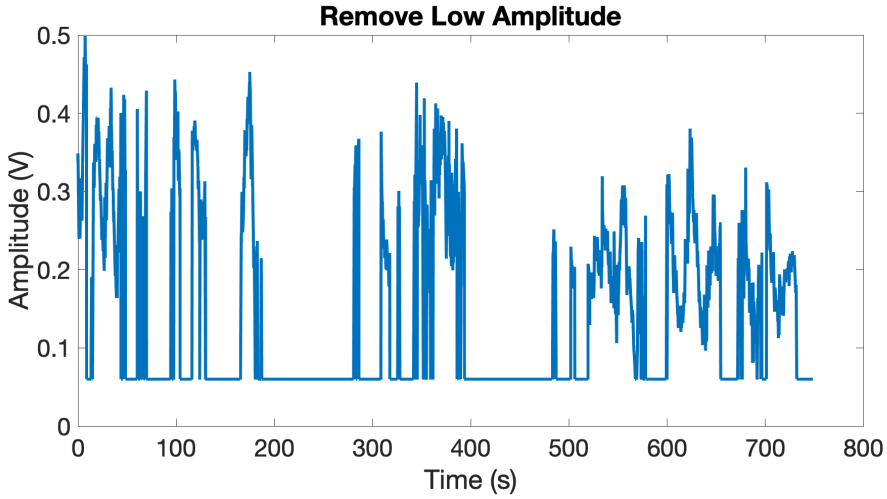


Figure 3.10: Data after pre-processing, detrend, zero out the mean, and remove the lower amplitude.

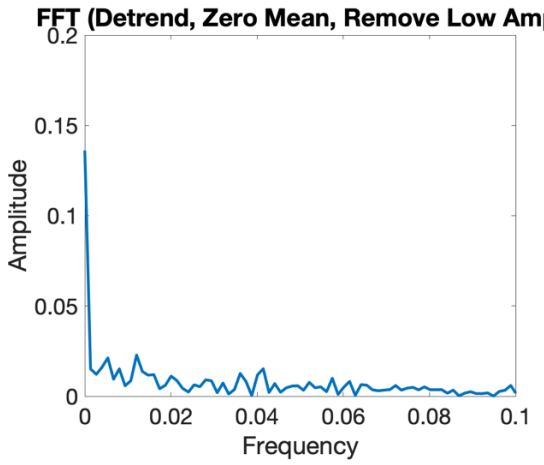


Figure 3.11: FFT of data after removing the low amplitudes. Two peaks appeared around a normalized frequency of 0.04.

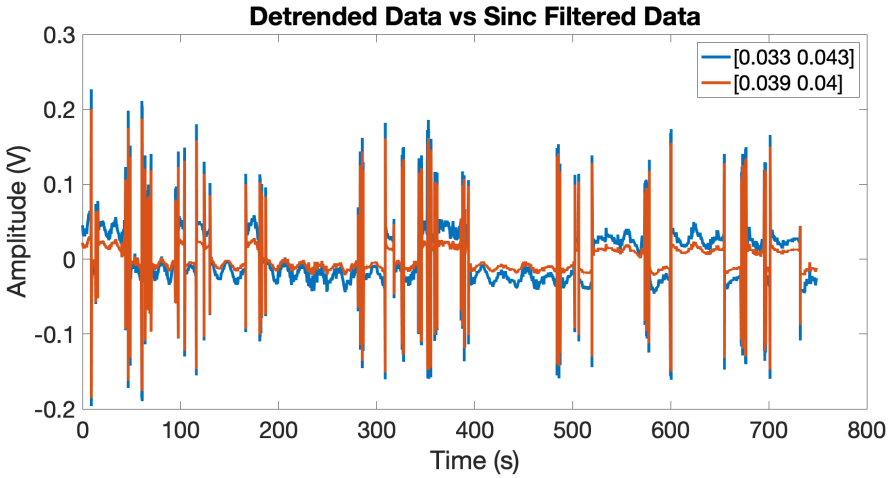


Figure 3.12: Sinc bandpass filtered data with cutoff frequency range set to be from 0.033 to 0.043 and 0.039 to 0.04 normalized frequency.

*3.1.3 Data analysis on Convolutional Filtered Data*

As mentioned in previous chapters, the duration of the windows used to perform a convolution on the data is set to 6 seconds. There are multiple examples of different scenarios: 1) a single event in a window; 2) an event lasting longer than 6 seconds, separated into three windows; 3) one single event with half of another event; 4) multiple close events located in the same window; and 5) a single event followed by multiple smaller events.

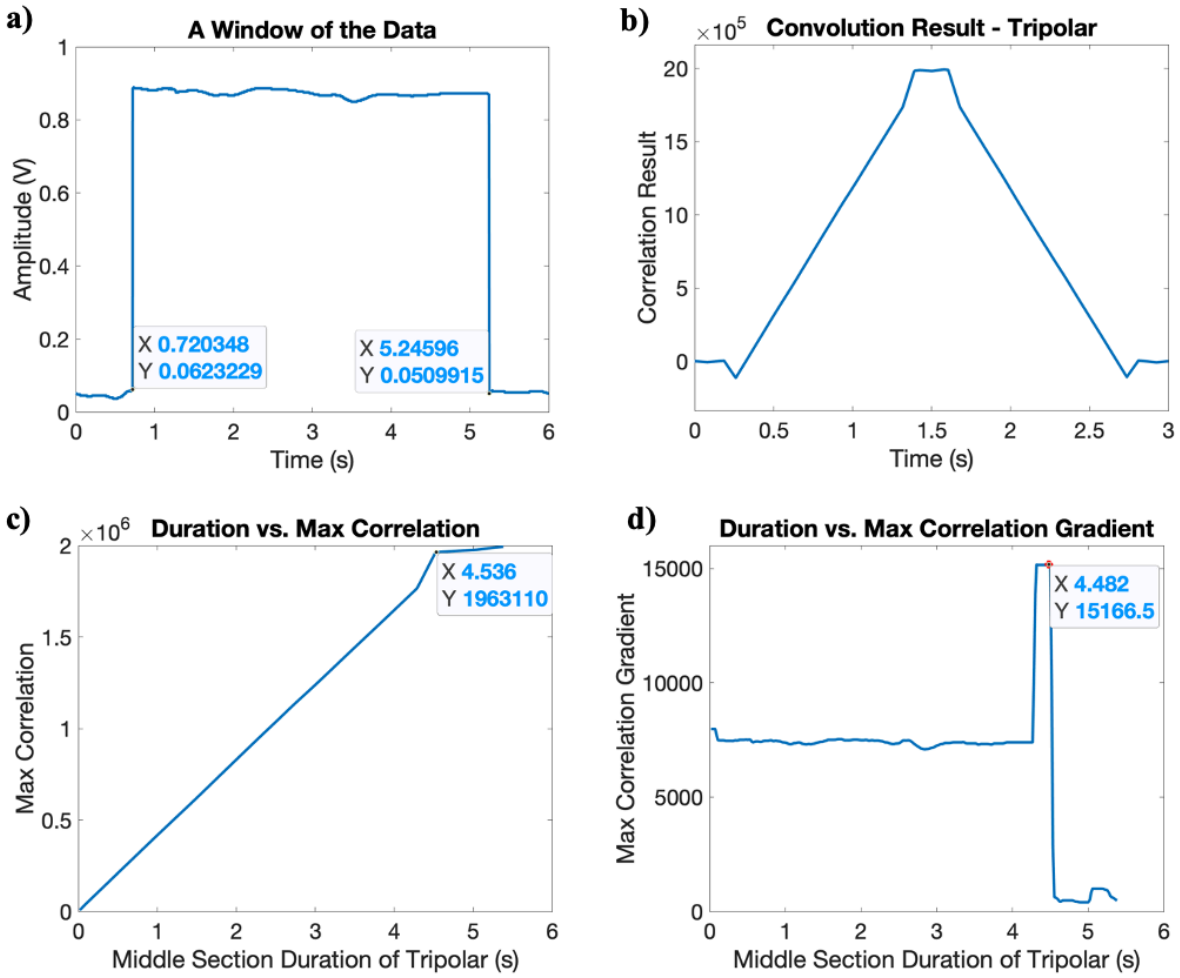


Figure 3.13: Single event in a window. The duration of the single event is 4.525 seconds. The peak data point selected from the duration and maximum correlation gradient plot is 4.482 seconds. There is a difference of 0.043 seconds.

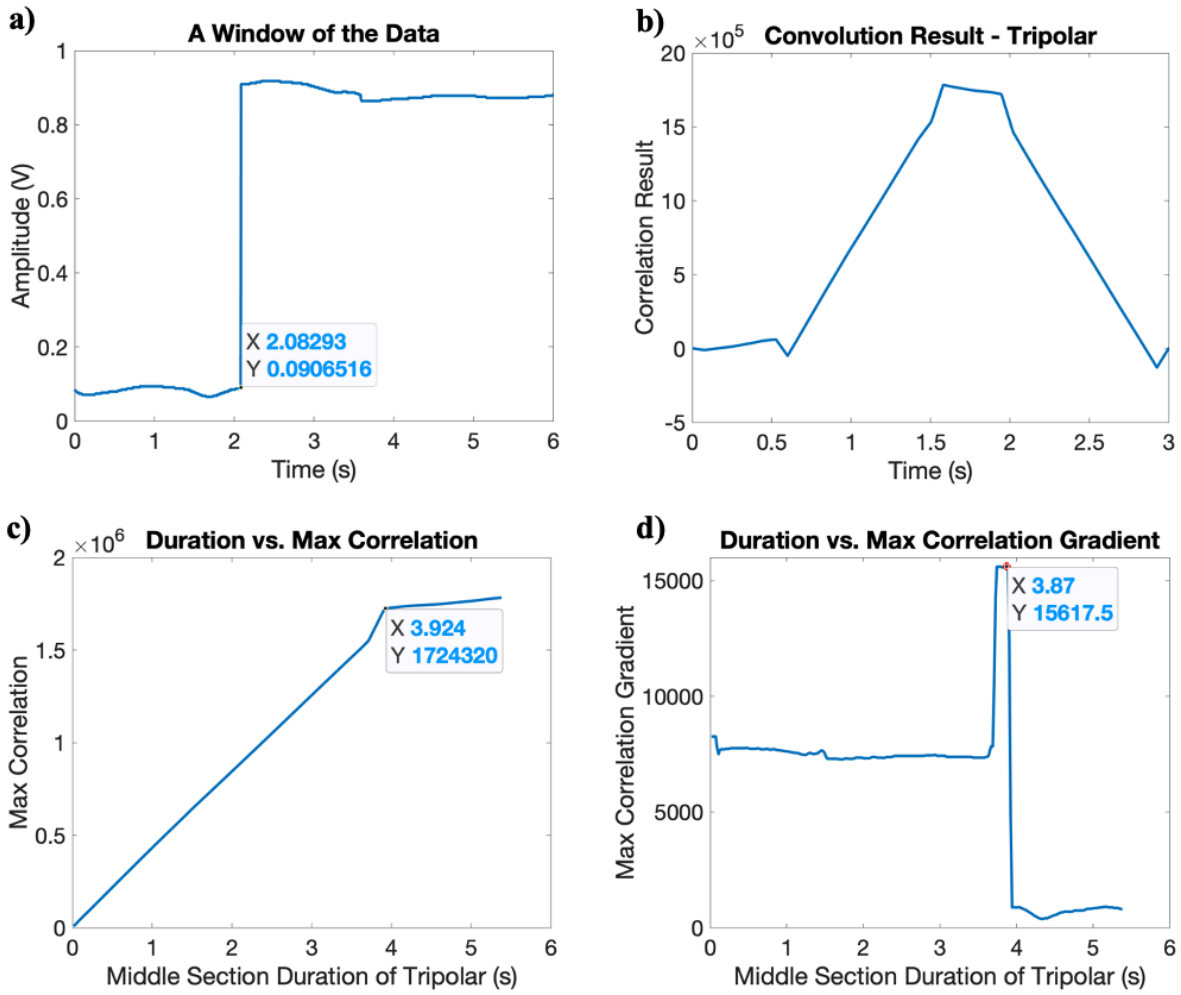


Figure 3.14: A start of event with duration longer than the window size of 6 seconds that is being separated into different windows. The first section of the event is 3.917 seconds, and the peak of the duration and max correlation gradient plot shows 3.87 seconds. There is a 0.047 second difference.

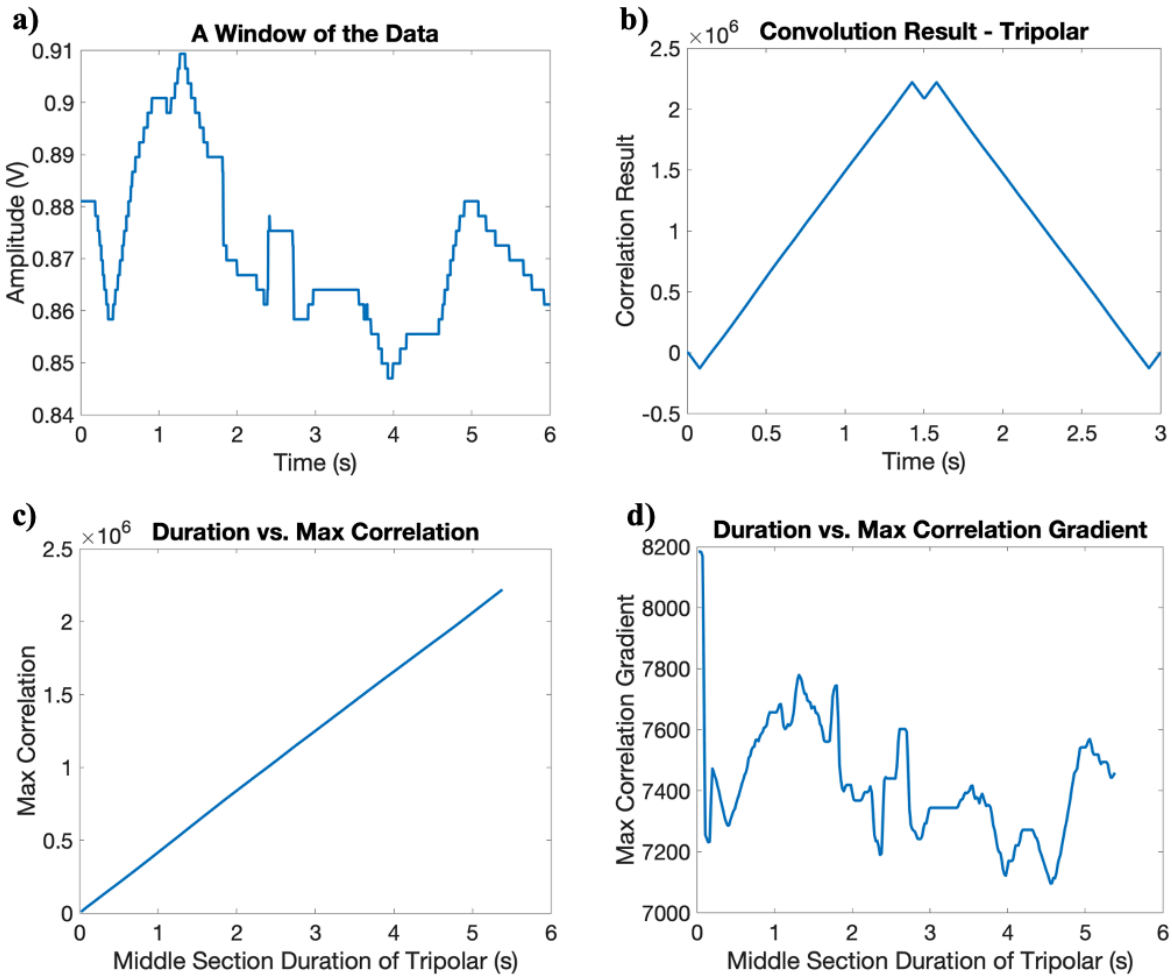


Figure 3.15: A middle section of an event with longer duration.

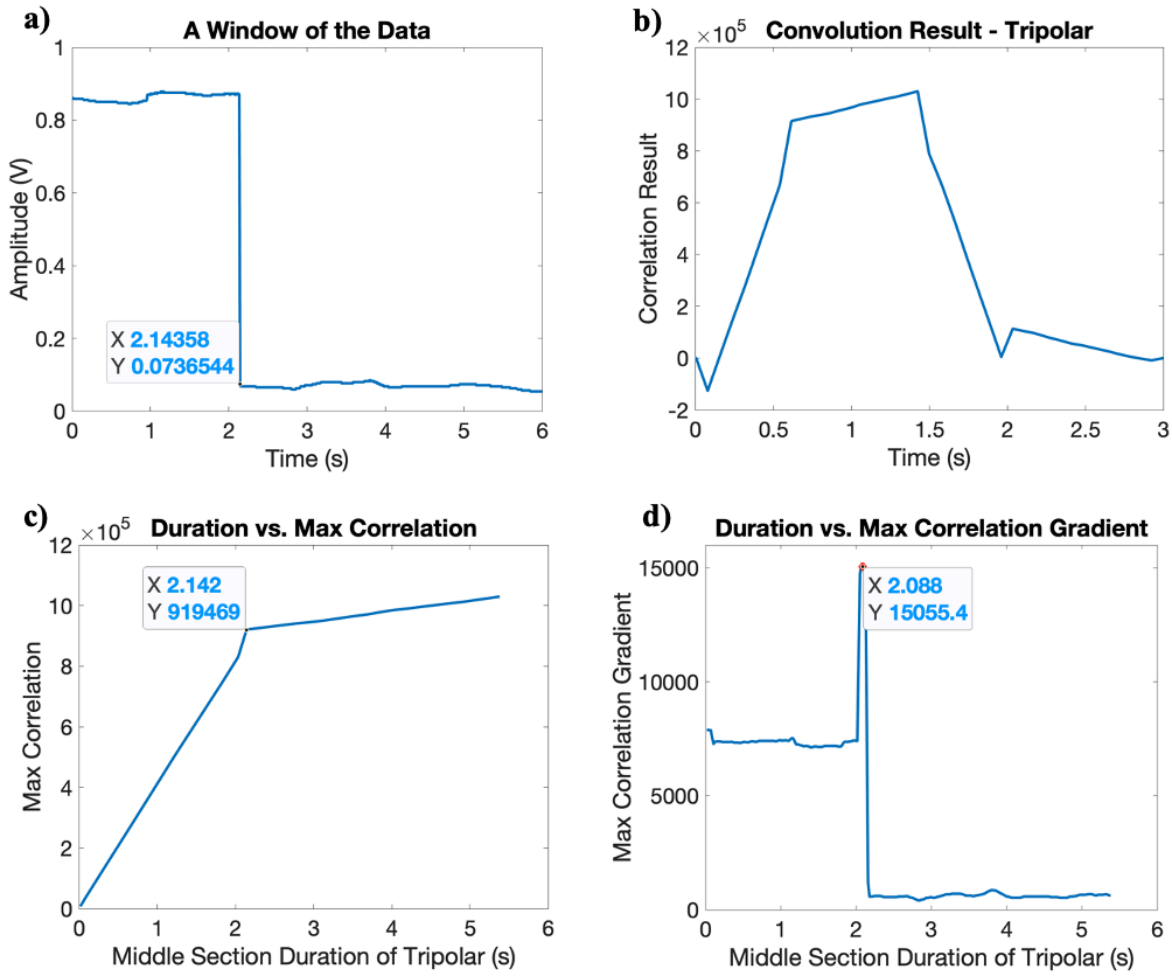


Figure 3.16: An ending section of an event with duration longer than the window length of 6 seconds. The end section of the event is 2.144 seconds, and the peak of the duration and max correlation gradient plot shows 2.088 seconds. There is a 0.056 second difference.

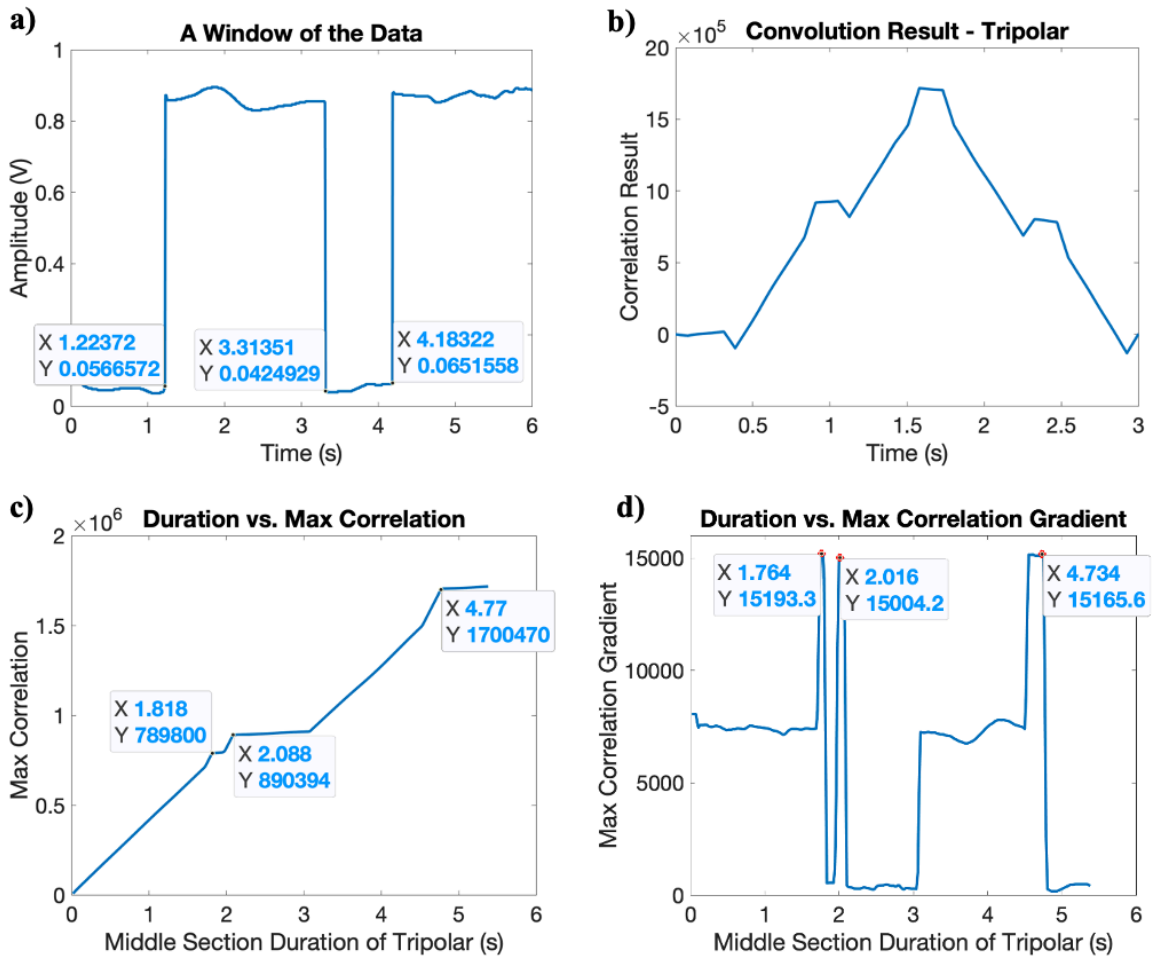


Figure 3.17: An event with a full duration and half of another event in a window. The first peak of the duration and maximum correlation gradient plot at 1.764 seconds indicates the duration of the half trapping event, which had a duration of 1.817 seconds. The second peak of the duration and maximum correlation gradient plot at 2.016 seconds indicates the event with full duration, which had a duration of 2.09 seconds. There is also a third peak in the duration and maximum correlation gradient plot at 4.734 seconds. This occurs when the reference wave and the signal convolve, identifying the start of the single event at 1.224 seconds to the end of the window at 6 seconds as another event, with a duration of 4.77 seconds.

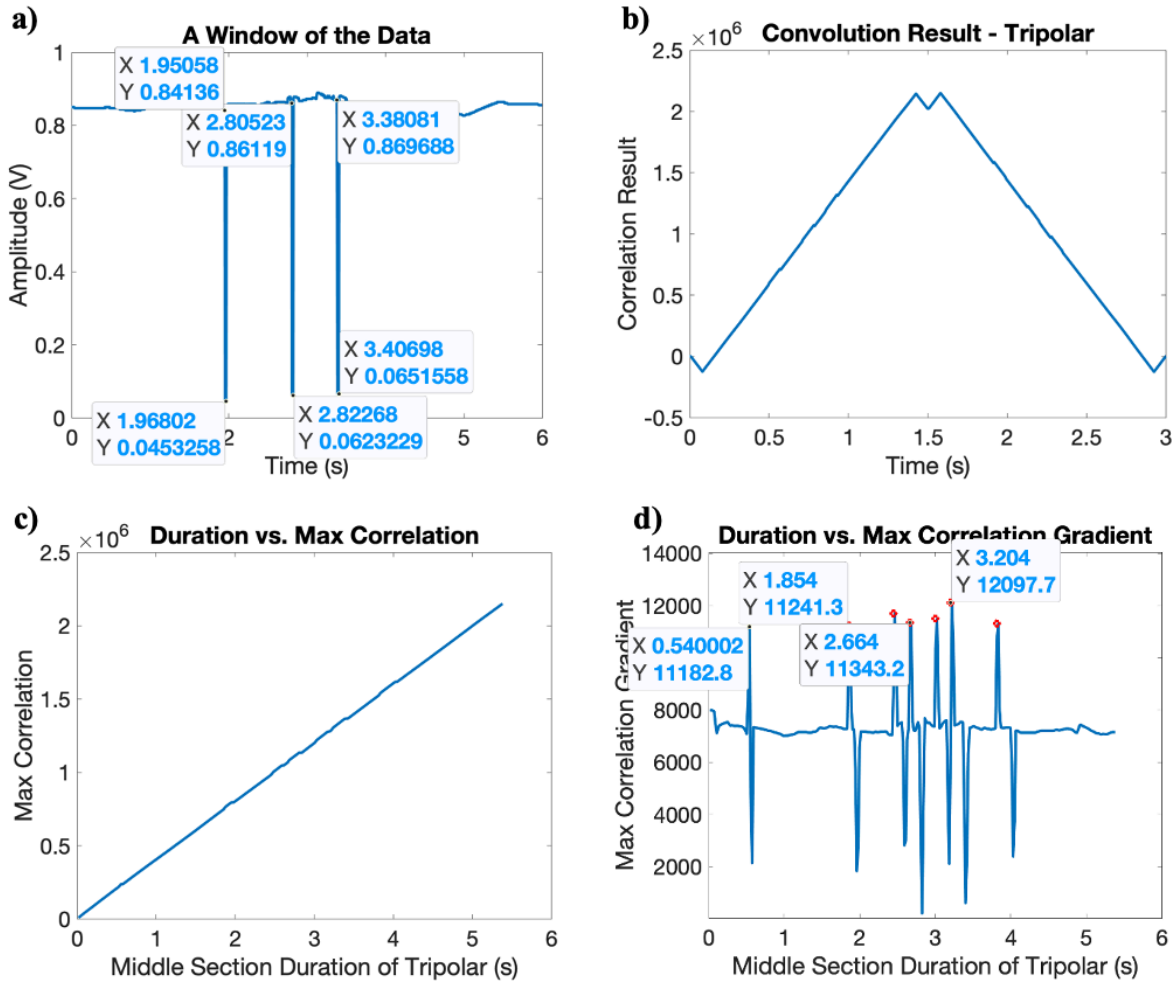


Figure 3.18: When multiple close events are in a window, the duration and maximum correlation gradient plot will show overfitted data points because it combines various events and identifies them as a multiple trapping event.

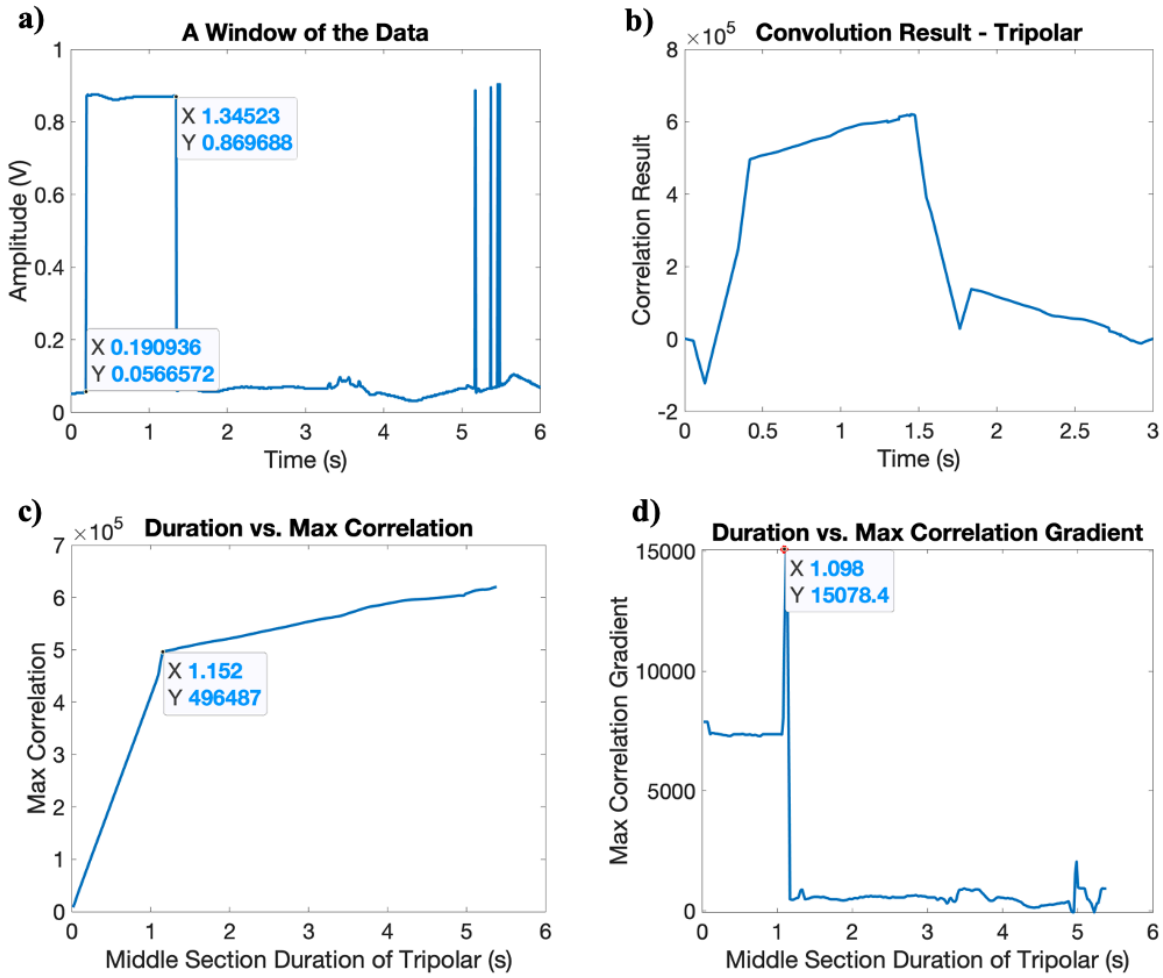


Figure 3.19: A single event with multiple smaller events in a window has a duration of 1.154 seconds. The peak shown in the duration and maximum correlation gradient plot is 1.098 seconds, resulting in a difference of 0.056 seconds. This is also an example showing that if a window contains an event with a longer duration than the other events, the smaller events will be excluded and not identified in the duration and maximum correlation gradient plot.

*3.1.4 Cumulative Results*

The convolutional filtering method is still under development and requires further work. Therefore, we will only compare the results processed with 'AutoStepfinder' and the Sinc bandpass filter followed by 'findpeak.' We used the 'pdist2' function in MATLAB to measure the similarity between the manually selected data, which is set as the standard, and the data processed by the two

methods. Lower distance values indicate higher similarity, while higher distance values indicate lower similarity.

Comparing result obtained from the optical transmission data:

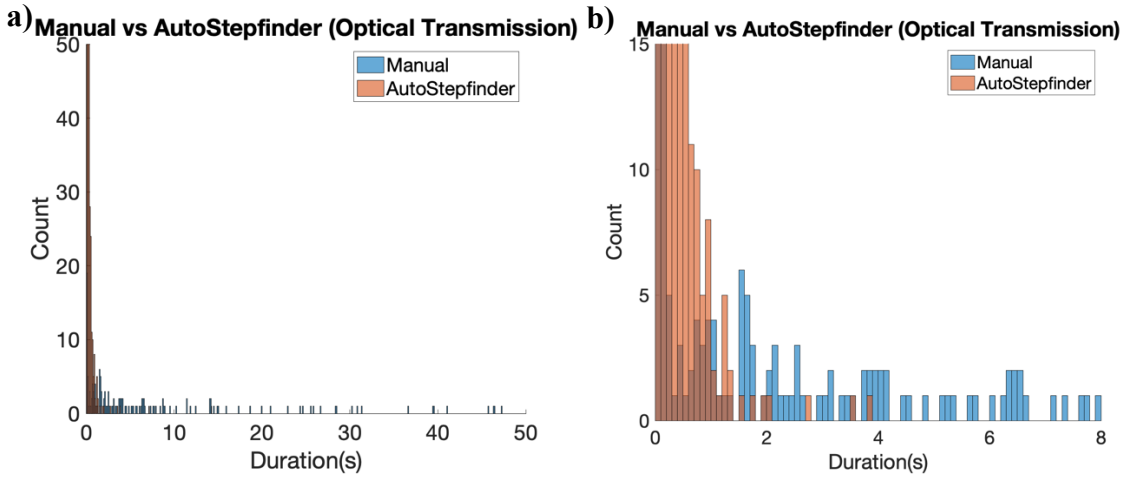


Figure 3.20: a) A histogram of step duration distribution of the manually selected and autostepfinder optical tramission data. b) A zoomed-in view of the histogram shown in (a), focusing on a specific range of values to provide more detailed information. The distance value comparing the manually selected data and data output from 'AutoStepfinder' is 178.23.

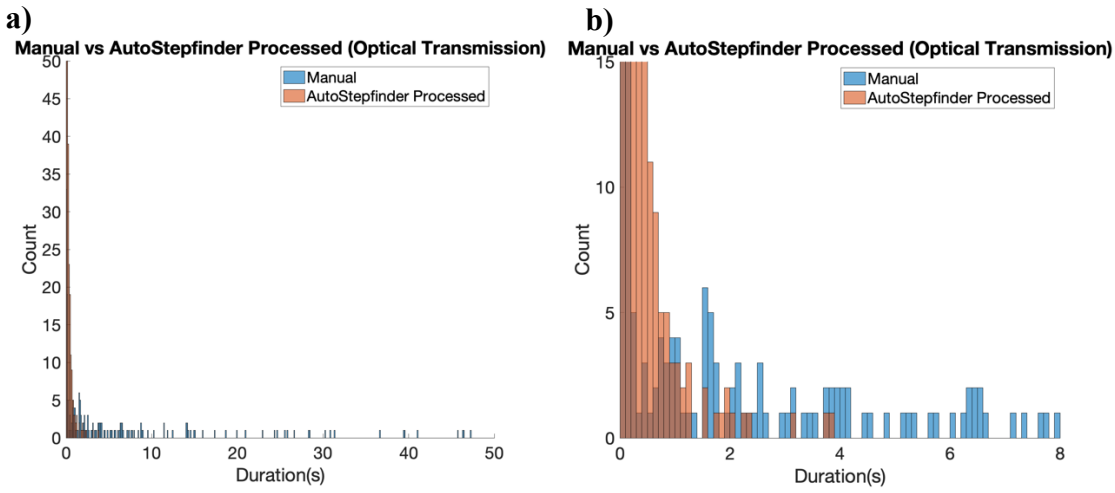


Figure 3.21: a) A histogram of step duration distribution of the manually selected data and post processed autostepfinder optical tramission data with a threshold varies between 0.5 V and 0.6 V. b) A zoomed-in view of the histogram shown in (a), focusing on a specific range of values to

provide more detailed information. The distance value comparing the manually selected data and post processed data ouput from ‘AutoStepfinder’ is 70.55.

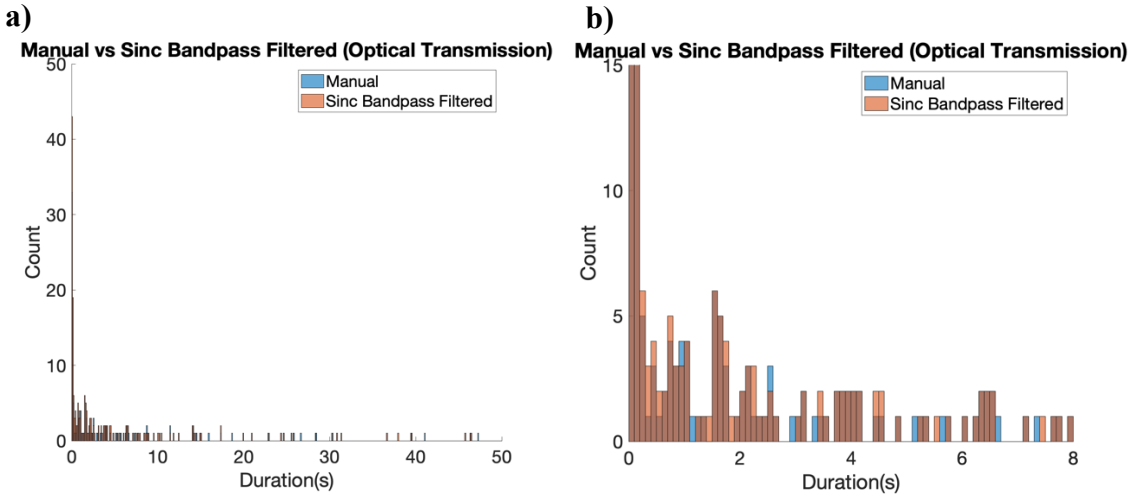


Figure 3.22: a) A histogram of step duration distribution of the manually selected data and Sinc bandpass filtered optical transmission data. b) A zoomed-in view of the histogram shown in (a), focusing on a specific range of values to provide more detailed information. The distance value comparing the manually selected data and Sinc bandpass filtered data is 13.42.

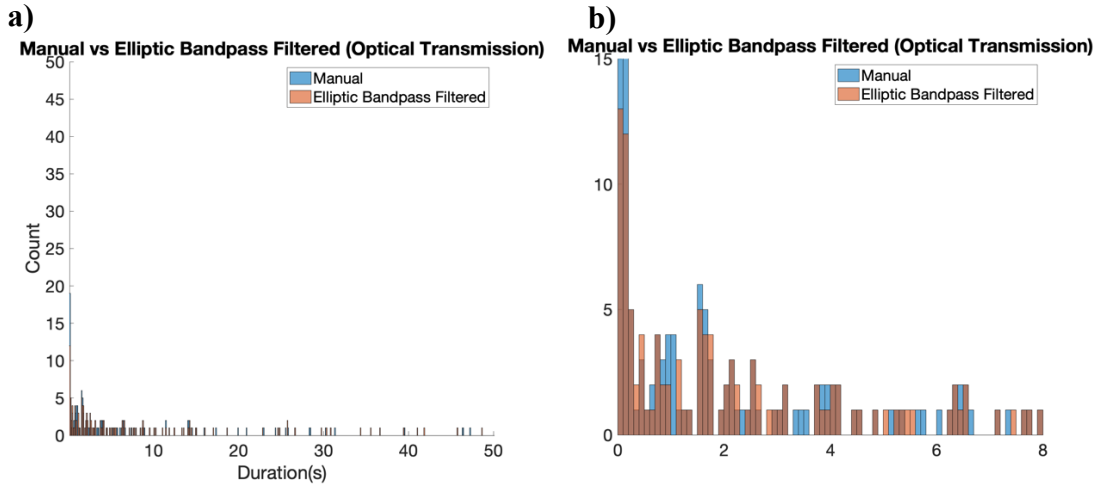


Figure 3.23: a) A histogram of step duration distribution of the manually selected data and Elliptic bandpass filtered optical transmission data. b) A zoomed-in view of the histogram shown in (a), focusing on a specific range of values to provide more detailed information. The distance value comparing the manually selected data and Elliptic bandpass filtered data is 26.53.

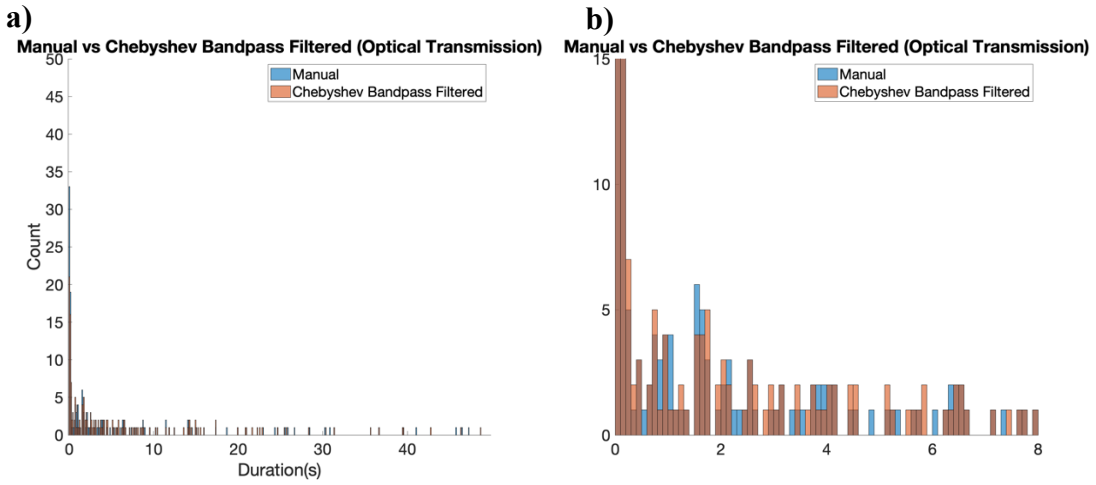


Figure 3.24: a) A histogram of step duration distribution of the manually selected data and Chebyshev bandpass filtered optical transmission data. b) A zoomed-in view of the histogram shown in (a), focusing on a specific range of values to provide more detailed information. The distance value comparing the manually selected data and Chebyshev bandpass filtered data is 21.70.

Comparing result obtained from the optical reflection data:

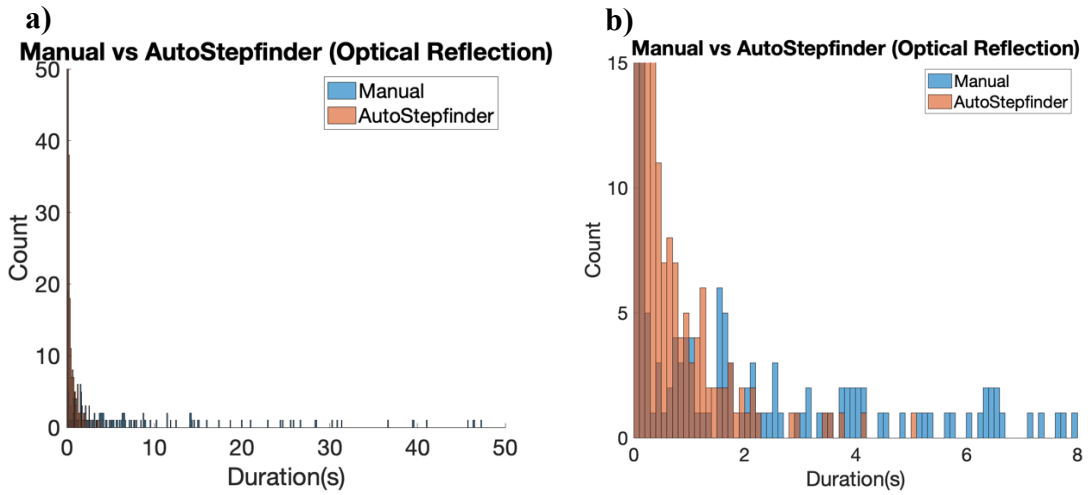


Figure 3.25: a) A histogram of step duration distribution of the manually selected and autostepfinder optical reflection data. b) A zoomed-in view of the histogram shown in (a), focusing on a specific range of values to provide more detailed information. The distance value comparing the manually selected data and data output from 'AutoStepfinder' is 68.14.

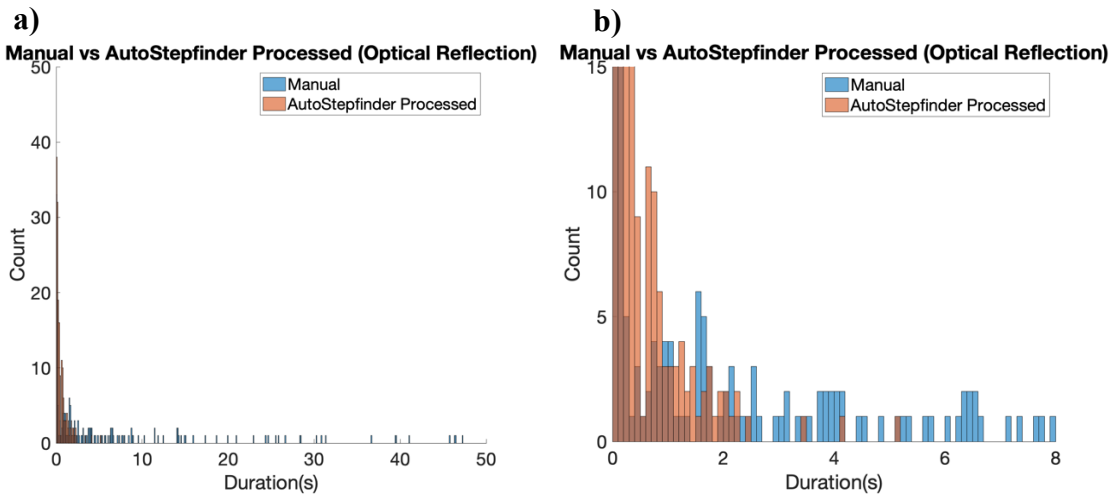


Figure 3.26: a) A histogram of step duration distribution of the manually selected data and post processed autostepfinder optical reflection data with a threshold varies between 0.5 V and 0.6 V. b) A zoomed-in view of the histogram shown in (a), focusing on a specific range of values to provide more detailed information. The distance value comparing the manually selected data and post processed data ouput from 'AutoStepfinder' is 25.22.

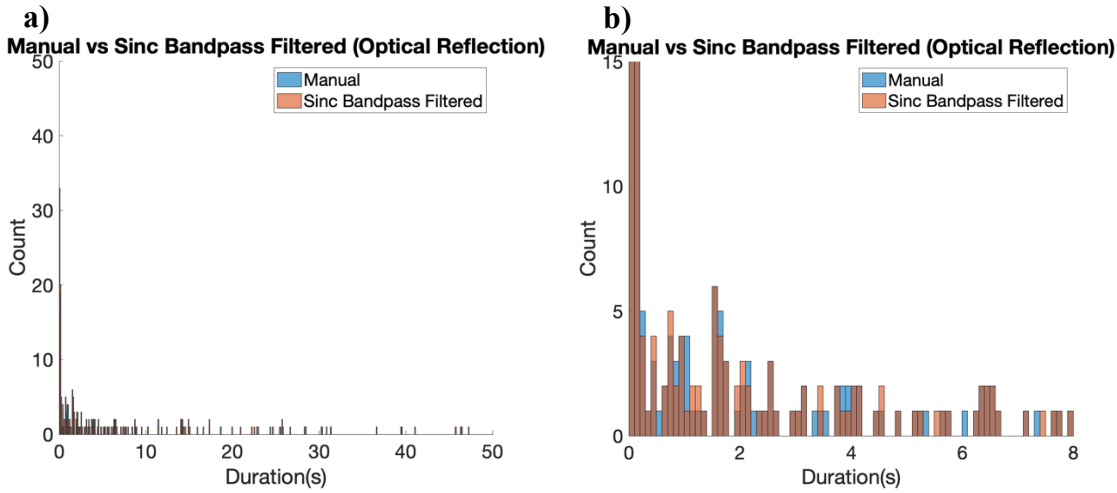


Figure 3.27: a) A histogram of step duration distribution of the manually selected data and Sinc bandpass filtered optical reflection data. b) A zoomed-in view of the histogram shown in (a), focusing on a specific range of values to provide more detailed information. The distance value comparing the manually selected data and Sinc bandpass filtered data is 20.22.

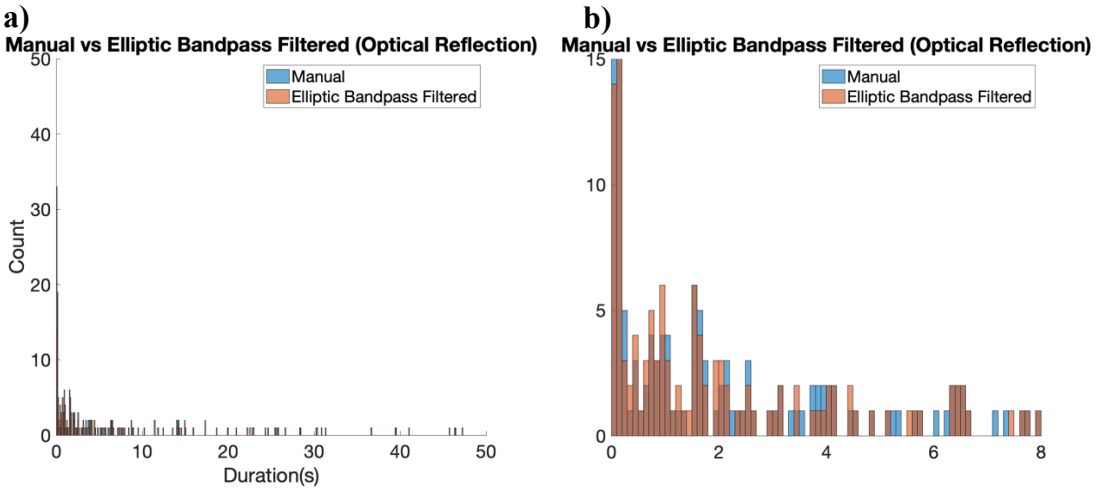


Figure 3.28: a) A histogram of step duration distribution of the manually selected data and Elliptic bandpass filtered optical reflection data. b) A zoomed-in view of the histogram shown in (a), focusing on a specific range of values to provide more detailed information. The distance value comparing the manually selected data and Elliptic bandpass filtered data is 27.53.

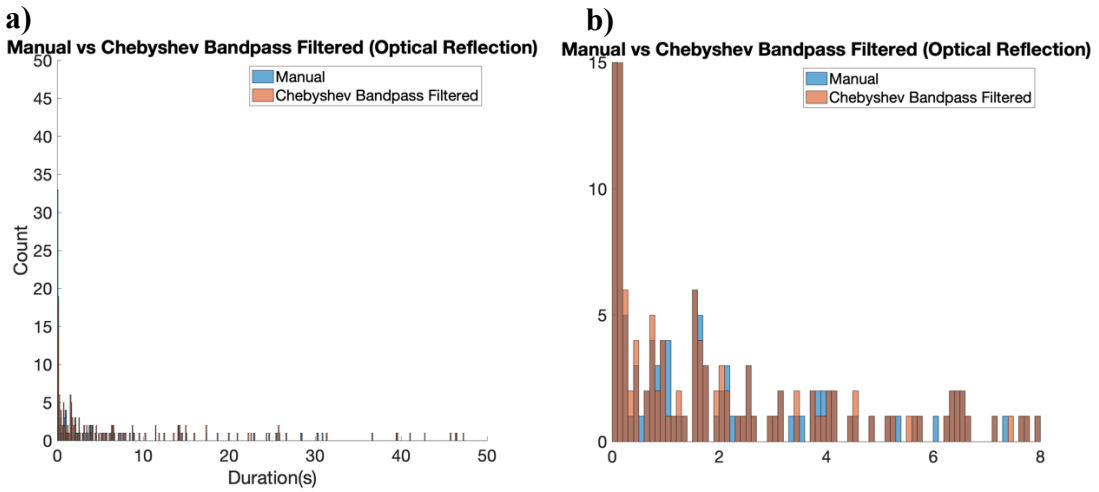


Figure 3.29: a) A histogram of step duration distribution of the manually selected data and Chebyshev bandpass filtered optical reflection data. b) A zoomed-in view of the histogram shown in (a), focusing on a specific range of values to provide more detailed information. The distance value comparing the manually selected data and Chebyshev bandpass filtered data is 26.12.

| | Optical Transmission Data | Optical Reflection Data |
|---|---|---|
| **AutoStepfinder** | 178.23 | 68.14 |
| **AutoStepfinder with Post-processing** | 70.55 | 25.22 |
| **Butterworth Bandpass Filtered Data** | 26.78 | 27.78 |
| **Elliptic Bandpass Filtered Data** | 26.53 | 27.53 |
| **Chebyshev Bandpass Filtered Data** | 21.7 | 26.12 |
| **Sinc Bandpass Filtered Data** | 13.42 | 20.22 |

Table 3.1: Comparison of pairwise distance values across different data processing methods.

# CHAPTER 4

# DISCUSSION AND CONCLUSIONS

Manually selecting data points is time-consuming because each dataset contains a different number of events, and human error can occur during the selection process. Therefore, there is a need to find a way to process SANE optical data accurately and efficiently. We tested different methods, (1) 'AutoStepfinder,' (2) Sinc bandpass filtering, and other filtering methods, followed by 'findpeak,' and (3) a convolutional filter method.

'AutoStepfinder' was initially tested, and we observed that due to noise in the data, the algorithm tends to overfit by identifying noise as steps, resulting in small durations, as seen in the histogram above. The number of iterations can be adjusted by analyzing the S-curve plot, which identifies the iteration with the highest S-value, as mentioned in Chapter Two. Although this adjustment reduces the amount of noise being identified as steps, it does not significantly alter the overall results. In this thesis, we focused on processing single-level events. The observation of 'AutoStepfinder' suggests that it can identify precise steps, which could be useful for multi-step event observations, but it greatly overestimates short (few hundred millisecond) duration steps as real signal and there is no software setting that can avoid that from happening.

The Sinc bandpass filtered data yielded the highest similarity to the manually selected data compared to all other methods compared in this work. However, there are multiple limitations to the Sinc bandpass filtering method. One limitation is that it can filter out data points, leading to inaccurate step duration analysis. After the initial analysis of the first step as the start or end of an event, the code assumes that the following data point continues from the first. For example, if the first data point is identified as the starting point of an event, the second data point will automatically be identified as the end of the event. If a data point is missing because it was filtered

out, the rest of the analysis will be inaccurate. Another limitation is that since we set a threshold for the 'findpeak' function to detect peaks, this assumes that every peak is higher than the threshold. The Sinc bandpass filtered method will not be able to identify a peak when it is close to the noise level or baseline.

As shown in the Results section, there are multiple scenarios when using the convolutional filtering method. This method requires fewer assumptions compared to the Sinc bandpass filtering method. The duration and amplitude of the reference wave are adjustable, meaning it does not require prior knowledge on the data. We were able to identify peaks from the duration and maximum correlation gradient plot, allowing us to determine the duration of different events. Although the peaks help us identify the duration of trapping events, if multiple events occur within the same window, peaks from the overfitted duration time points will appear in the duration and maximum correlation plot, e.g. Figure 3.11 and 3.12. We have not found a significant feature that allows us to distinguish the actual event duration from the overfitted duration without visualizing the plots. Another issue is that the output data points do not clearly indicate whether a duration corresponds to a single event, a starting section of an event split across different windows (resulting for splitting time-series data to pieces due to computer memory limitations), or the middle of an event. The convolution step within the code is also computationally intensive, requiring significant computational power to process in MATLAB. Overall, while we found features that suggest possible durations of trapping events, we cannot yet determine whether the output duration is the actual trapping duration or an overfitted duration that incorrectly identifies the start and end of a trapping event. More work is needed to develop and optimize the convolutional filtering method.

In summary, all methods examined in this work were able to provide step duration data, but they need further refinement and optimization. The data obtained from 'AutoStepfinder'

includes an overestimate smaller duration step-changes because it identifies noise as steps, leading to inaccuracy of the data histograms. It is interesting that this is meant to be a powerful, general purpose software for analyzing step data, but it was written to look for steps in fluorescence data, where small-amplitude, rapid signal fluctuations could be part of the real signal. However, in our case, the background fluctuations were cause by other nanobeads that were not at the center of the SANE sensor's optical trap but were transiently near enough to contribute to signal fluctuations. These fluctuations are uncommon to other typical time-series data types and therefore the 'AutoStepfinder' algorithm could not address them, unless we had worked on some strong pre-filtering step or spending significant time to adjust this open-source algorithm instead.

Instead, for the next step of this work we decided to move into a pre-filtering data approach that makes step-changes easily identifiable. The data processed with the Sinc bandpass filter followed by 'findpeak' showed the most promising results of all the different filter types tested. However, this method has limitations too, due to the threshold setting for peak detection, which prevents the detection of peaks close to the noise level. In addition, the threshold determination step was empirical, and we did not find an easy way to make this selection obvious. If future work, more efforts need to be placed on analyzing the signal's frequency content, e.g. by time-frequency analysis to gain more detailed understanding of the frequency content variations in these signals with the purpose of identifying a data-driven threshold for these analyses.

Lastly, the convolutional filter method allowed us to identify step durations when a single event occurred within a time window of time-series data. We identified an asymmetric filter kernel that was successful in identifying single steps as long as they were not too close, e.g. fewer than a few hundred milliseconds, form another step. However, an improved method is needed to accurately identify the start and end of step-like events when these occur close to each other, and

in the future on top of each other (multi-step scenario). In addition, because the convolution step is computationally intensive, data needs to be split is short (multi-second) pieces, where an event begins at the end of one data set and ends in the next. To accommodate for this scenario, this method needs to have a contingency for bridging analysis data between time windows.

For future work, we aim to develop software using a convolutional neural network (CNN) to classify data obtained from SANE sensors into different analyte categories. CNNs have been successfully developed and used to identify abnormalities in ECG signals [9]. It is time-consuming and labor-intensive to do signal processing and feature extraction to output optimal results from SANE sensor data. Previous studies have shown that CNNs can yield high classification accuracy, sensitivity, and specificity, helping researchers avoid manual examination of ECG signals and reducing human errors [9]. We can expand our training dataset as our database grows and potentially include artificially generated data [10]. The algorithm will consist of multiple convolutional and pooling layers to extract complex patterns of SANE sensor data obtained from different analytes.

# APPENDIX A

# CODE FOR OPTICAL DATA PROCESSING

Appendix A Content

Sinc Bandpass Filtered Method Followed by Peak Finding

```
clc;
close all
clear;
close all

%% Load and define data
% Load .abf file
fileName = '2022_07_19_0003 ORL W2C14 equimolar 1 aM new prep.abf';
[data0, si, ~] = abfload(sprintf(fileName), 'start', 0, 'sweeps', 'a');

% Create time vector
SamplePeriod = si * 1e-6;
contime = [0:SamplePeriod:(length(data0)*SamplePeriod - 1*SamplePeriod)];
% Add time vector to abfdata
abftime = [contime' data0];
% Sampling frequency
fs = 500000;
% Nyquist frequency
fn = fs/2;
% Optical transmission
Optical = abftime(:, 4);
% Optical reflection
OpticalRef = abftime(:, 5);
Time = [0:SamplePeriod:(length(Optical)*SamplePeriod - 1*SamplePeriod)];

%% Pre-processing
% Calculate optical signal to between 0 and 1
MinOptical = min(Optical);
MaxOptical = max(Optical);
NormSmoothedOptical = (Optical - MinOptical)/(MaxOptical - MinOptical);

% Median filter
SmoothedOptical = medfilt1(NormSmoothedOptical, 17501, 'omitnan');

% Detrend
dt_SmoothedOptical = detrend(SmoothedOptical);

% Zero out the mean
mean_SmoothedOptical = mean(dt_SmoothedOptical);
Removal_DC_Freq = dt_SmoothedOptical - mean_SmoothedOptical;

%% Compute the FFT
FFTOptical = fft(SmoothedOptical);
FFTOptical_amp = abs(FFTOptical/length(SmoothedOptical));
FFTOptical_shift = fftshift(FFTOptical_amp);
FFTOptical_singleside = FFTOptical_shift(length(FFTOptical)/2+1:end);
Frequency = (0:length(FFTOptical)/2-1)*fs/length(FFTOptical);
Frequency = Frequency';
% figure;
% plot(Frequency, FFTOptical_singleside)
% axis([0 0.1 0 0.5])
% xlabel('frequency')
```

```
% ylabel('amplitude')

%% Sinc bandpass filter
fcL = 0.039; % 0.033
fcH = 0.04; % 0.043
N = 45;
% fir1(n, Wn, ftype): n (filter order), Wn (frequency constraints), ftype
(filter type), window type
h2 = fir1(N, [fcL fcH], 'bandpass', kaiser(N+1, 0.5));
SincData = filtfilt(h2, 1, SmoothedOptical);
figure;
plot(Time, SmoothedOptical, Time, WindowSincData, 'LineWidth', 3)
title('Sinc Bandpass Filtered Data', 'FontSize', 27)
xlabel('Time (s)', 'FontSize', 24)
ylabel('Amplitude (V)', 'FontSize', 24)
legend('Pre-Processed', 'Sinc Filtered')
ax = gca;
set(ax, 'FontSize', 24);

% Compute frequency response of the Sinc filter
[h2, w2] = freqz(h2, 1, 8000);
dB = mag2db(abs(h2));
figure;
plot(w2/pi, dB, 'LineWidth', 3);
title('Sinc Bandpass Filter Frequency Response', 'FontSize', 27);
xlabel('Normalized Frequency (\times\pi rad/sample)', 'FontSize', 24);
ylabel('Magnitude (dB)', 'FontSize', 24);
grid on;
ax = gca;
set(ax, 'FontSize', 24);

% Find peaks
[peakValues, peakLocations] = findpeaks(SincData, 'MinPeakHeight', 0.07);
peakTimes = peakLocations * 2e-6;
figure;
plot(Time, SincData, 'LineWidth', 1.5);  % Plot the original data
hold on;
plot(peakTimes, peakValues, 'rv', 'MarkerFaceColor', 'r', 'LineWidth', 3);  %
Mark the peaks in red
hold off;
title('Detected Peaks', 'FontSize', 27);
xlabel('Time (s)', 'FontSize', 24);
ylabel('Amplitude (V)', 'FontSize', 24);
ax = gca;
set(ax, 'FontSize', 24);

% Remove the unwanted data
i = 1;
while i < length(peakTimes)
    if (peakTimes(i+1) - peakTimes(i)) < 0.003
        peakTimes(i+1) = [];
    else
        i = i + 1;
    end
end

% Odd time points
```

```
oddpeakTime = peakTimes(1:2:end);
% Even time points
evenpeakTime = peakTimes(2:2:end);
```

The Convolutional Filtered Method

```
clc;
close all
clear;
close all
tic

%% Load and define data
% Load .abf file
fileName = '2022_07_19_0003 ORL W2C14 equimolar 1 aM new prep.abf';
[data0, si, ~] = abfload(sprintf(fileName), 'start', 0, 'sweeps', 'a');

% Create time vector
SamplePeriod = si * 1e-6;
contime = [0:SamplePeriod:(length(data0)*SamplePeriod - 1*SamplePeriod)];
% Add time vector to abfdata
abftime = [contime' data0];
% Sampling frequency
fs = 500000;
% Nyquist frequency
fn = fs/2;
% Optical transmission
Optical = abftime(:, 4);
% Optical reflection
OpticalRef = abftime(:, 5);
Time = [0:SamplePeriod:(length(Optical)*SamplePeriod - 1*SamplePeriod)];

%% Pre-processing
% Calculate optical signal to between 0 and 1
MinOptical = min(Optical);
MaxOptical = max(Optical);
NormSmoothedOptical = (Optical - MinOptical)/(MaxOptical - MinOptical);

% Median filter
SmoothedOptical = medfilt1(NormSmoothedOptical, 17501, 'omitnan');

% Detrend
dt_SmoothedOptical = detrend(SmoothedOptical);

% Zero out the mean
mean_SmoothedOptical = mean(dt_SmoothedOptical);
Removal_DC_Freq = dt_SmoothedOptical - mean_SmoothedOptical;

%% Chop data into different windows
% Length of each window 6s
windowLength = 3000000;
numWindows = floor(length(Removal_DC_Freq) / windowLength);
OpticalWindows = cell(numWindows, 1);

% Create each window variable
for k = 1:numWindows
```

```
    startIdx = (k - 1) * windowLength + 1;
    endIdx = k * windowLength;
    if endIdx > length(Removal_DC_Freq) % Check to avoid indexing beyond the
data in the last section
        endIdx = length(Removal_DC_Freq);
    end
    OpticalWindows{k} = Removal_DC_Freq(startIdx:endIdx); % Assign section to
cell array
end

Window = OpticalWindows{13,1};
Time = [0:SamplePeriod:(length(Window)*SamplePeriod - 1*SamplePeriod)];
figure;
plot(Time, Window, 'LineWidth', 3);
title('A Window of the Data', 'FontSize', 27)
xlabel('Time (s)', 'FontSize', 24)
ylabel('Amplitude (V)', 'FontSize', 24)
ax = gca;
set(ax, 'FontSize', 24);

%% Convolution
% Define the reference wave lengths
Lengths = 1:10000:length(Window);
Amplitude = 1;
NegAmplitude = -1;
MarginDuration = 2500;

MaxCorrArray = [];
BestMaxCorr = -Inf;
BestLength = 0;

for Len = Lengths
    % Generate tripolar reference wave
    ReferenceOptical = zeros(1, Len);
    firstThirdLength = floor(Len/20);
    middleSectionLength = Len - 2*firstThirdLength;
    ReferenceOptical(MarginDuration+1:firstThirdLength) = NegAmplitude;
    ReferenceOptical(firstThirdLength+1:firstThirdLength+middleSectionLength) =
Amplitude;
    ReferenceOptical(firstThirdLength+middleSectionLength+1:end-
MarginDuration) = NegAmplitude;

    % Perform convolution
    [ConvolutionResult, Lags] = xcorr(Window, ReferenceOptical);
    [MaxCorr, MaxCorrIndex] = max(ConvolutionResult);
    MaxCorrArray = [MaxCorrArray, MaxCorr]; % Append the current MaxCorr to
the array
end

LagIndices = linspace(0,3,length(ConvolutionResult));

% Plot convolution result
figure;
plot(LagIndices, ConvolutionResult, 'LineWidth', 3);
title('Convolution Result - Tripolar', 'FontSize', 27)
ylabel('Correlation Result', 'FontSize', 24);
xlabel('Time (s)', 'FontSize', 24);
```

```
ax = gca;
set(ax, 'FontSize', 24);

% Plot Duration vs Max Correlation
FirstDuration = Lengths/20;
MidDuration = Lengths - 2*FirstDuration;
MidDuration = MidDuration';
MidDuration = MidDuration*2e-6;
MaxCorrArray = MaxCorrArray';
figure;
plot(MidDuration(2:end), MaxCorrArray(2:end), 'LineWidth', 3);
title('Duration vs. Max Correlation', 'FontSize', 27);
xlabel('Middle Section Duration of Tripolar (s)', 'FontSize', 24);
ylabel('Max Correlation', 'FontSize', 24);
ax = gca;
set(ax, 'FontSize', 24);

%% Looking for the peaks in the duration vs. max correlation gradient plot
MaxCorr_gradient = gradient(MaxCorrArray(2:end));
[peakValues, peakLocations] = findpeaks(MaxCorr_gradient,
'MinPeakProminence',5000); % Adjust 'MinPeakProminence' as needed
peakTimes = peakLocations * 2e-6;
figure;
plot(MidDuration(2:end), MaxCorr_gradient, 'LineWidth', 3)
hold on;
plot(MidDuration(peakLocations), peakValues, 'or', 'LineWidth', 3);
xlabel('Middle Section Duration of Tripolar (s)', 'FontSize', 27)
ylabel('Max Correlation Gradient','FontSize', 24)
title('Duration vs. Max Correlation Gradient', 'FontSize', 24)
ax = gca;
set(ax, 'FontSize', 24);

timeElapsed = toc;
```

REFERENCES

[1] Liu K, Feng J, Kis A, Radenovic A. Atomically thin molybdenum disulfide nanopores with high sensitivity for DNA translocation. ACS Nano. 2014 Mar 25;8(3):2504-11. doi: 10.1021/nn406102h. Epub 2014 Feb 18. PMID: 24547924.

[2] Raillon C, Cousin P, Traversi F, Garcia-Cordero E, Hernandez N, Radenovic A. Nanopore detection of single molecule RNAP-DNA transcription complex. Nano Lett. 2012 Mar 14;12(3):1157-64. doi: 10.1021/nl3002827. Epub 2012 Mar 1. PMID: 22372476.

[3] Kowalczyk SW, Hall AR, Dekker C. Detection of local protein structures along DNA using solid-state nanopores. Nano Lett. 2010 Jan;10(1):324-8. doi: 10.1021/nl903631m. PMID: 19902919.

[4] Dekker C. Solid-state nanopores. Nat Nanotechnol. 2007 Apr;2(4):209-15. doi: 10.1038/nnano.2007.27. Epub 2007 Mar 4. PMID: 18654264.

[5] Raza MU, Peri SSS, Ma LC, Iqbal SM, Alexandrakis G. Self-induced back action actuated nanopore electrophoresis (SANE). Nanotechnology. 2018 Oct 26;29(43):435501. doi: 10.1088/1361-6528/aad7d1. Epub 2018 Aug 3. PMID: 30073973.

[6] Bast Labs. (n.d.). *EVENTPRO: Nanopore data analysis app*. EventPro | BAST LABS: Nanopore Force Spectroscopy. Retrieved March 4, 2022, from http://bastlabs.org/eventpro.html.

[7] Loeff L, Kerssemakers JWJ, Joo C, Dekker C. *AutoStepfinder*: A fast and automated step detection method for single-molecule analysis. Patterns (N Y). 2021 Apr 30;2(5):100256. doi: 10.1016/j.patter.2021.100256. PMID: 34036291; PMCID: PMC8134948.

[8] M. A. Hossain, "Performance analysis of barker code based on their correlation property in multiuser environment", Int. J. Inf. Sci. Techn., vol. 2, no. 1, pp. 27-39, Jan. 2012.

[9] Jahmunah V, Ng EYK, San TR, Acharya UR. Automated detection of coronary artery disease, myocardial infarction and congestive heart failure using GaborCNN model with ECG signals. Comput Biol Med. 2021 Jul;134:104457. doi: 10.1016/j.compbiomed.2021.104457. Epub 2021 May 7. PMID: 33991857.

[10] Cheng J, Zou Q, Zhao Y. ECG signal classification based on deep CNN and BiLSTM. BMC Med Inform Decis Mak. 2021 Dec 28;21(1):365. doi: 10.1186/s12911-021-01736-y. PMID: 34963455; PMCID: PMC8715576.