SURF Interim Reports

Examining Change Points in LIGO Hanford Data over Observing Run 4 to Improve Detector Performance

Iain Morton[1], Dr. Ansel Neunzert[2], and Camilla Compton[2]

[1]Department of Physics, Seton Hall University
[2]LIGO Hanford Observatory

September 23, 2024

DCC: T2400227
California Institute of Technology
LIGO Hanford Observatory
National Science Foundation

Contents

1 Abstract

LIGO data is imperfect, largely due to ongoing phenomena that interfere with continuous gravitational wave detection. As such, investigating certain disruptions in this gravitational wave detection is important for run analysis. This is true for both analyzing narrow spectrum artifacts over long-range duration searches and for refining existing documentation, which tags potential reasons for losing lock. More specifically, change-point detection methods derived from theoretical digital signal processing is a proposed method for automating line detection over LIGO noise. Mathematical and computational efficiency for this change-point detection is introduced and programmed.

Existing documentation which correlate certain reasons for lockloss requires continual refinement. Over the O3 run, a considerable number of locklosses were tagged for unknown reasons. Tagging is handled by the Python Locklost tool, which requires plugins for tagging events. This project involves developing a glitch plugin which establishes an amplitude threshold within a certain frame prior to the lockloss event itself. Furthermore, we are also developing a plugin which checks for ASC saturation. These practices will ensure for further refinement for detecting and analyzing various reasons for lockloss.

2 Narrow Spectral Artifacts

2.1 Background and Motivation

Previously, we have provided a formal definition of the usage and motivation of the cost-function and discrepancy to analyze change-point methods in incoming noise. As defined previously, my project's aim is to utilize python to detect change points in detected LIGO noise. This is motivated to determine the behavior and persistence of particular combs over the duration of several dates over the O4a run.

Identical methods for using cost-functions in the form of the sum of least square has remained identical. However, an additional discrepancy has since been changed for simpler plotting and data detection. Our purpose was to keep the window-sliding technique introduced in the change-point detection methods for signal processing, but the discrepancy curve was then changed to be plotted between 0 and 1, which allows for an easier threshold to be established to the discrepancy data. As such, our initial discrepancy plot was depicted as:

$$D(y_{a,t}, y_{t,b}) = c(y_{a,b}) - c(y_{a,t}) - c(y_{t,b}) \tag{1}$$

However, this discrepancy function, when applied over a comb, outputted various line heights over certain frequencies, which became difficult for an appropriate threshold to be established. Hence, the discrepancy function has been modified to:

$$D(y_{a,t}, y_{t,b}) = 1 - \frac{(c(y_{a,t}) + c(y_{t,b}))}{c(y_{a,b})} \tag{2}$$

Which made the development of a threshold possible for each frequency.

Furthermore, previous methods of incorporating a provided penalty function to a criterion function (see Interim Report 1) has been considered for the purpose of isolating segmentations of said change-points. The penalty function has been defined as:

$$Pen_{l0}(\tau) := \beta|\tau| \tag{3}$$

However, the smoothing parameter (β) in (3) is assumed with an established data model and a known variance. Given how the analyzed combs do not have a previously determined variance amongst change-points, it was necessary to disregard establishing the penalty function to the criterion, and consider alternative methods.

One such alternative method is to establish a discrepancy curve (defined above) for plotting, in which the peak prominence package from SciPy may isolate and distinguish between slope-points and peak-points in the discrepancy array for each comb range. Following this, a threshold may be introduced to then distinguish the exact moments of said change-points from the line-height graphs. This will then be able to be compared to the respective dates on which said change points occur.

2.2 Final Progress

Following the failure of properly implementing the smoothing parameter to the penalty function, this hence rendered the method of the criterion function inefficient for our particular purposes. As such, the immediate next steps were choosing a particular comb range for O4a line heights. The past chosen frequencies were 24.4 hertz and 24.5 hertz, respectively. This has since changed to 6.977 hertz, with frequency combs varying with the following equation:

$$f_n = f_0 + n * \delta f \tag{4}$$

Where f_n refers to the tooth frequency, f_0 is the offset (from 0 hertz), n is an integer, and δf is the a spacing. The 6.977 hertz frequency was implemented with $n = 26$, and then stored in an array for each n multiplication of 6.977 hertz.

This was then plotted, with the line height of each frequency over each date in O4a depicted in the below figure. Visual change points are immediately noticeable around November 2023, which was noted and compared to the subsequent graphs. As accomplished before for the 24.4 and 24.5 hertz frequencies, the line height for each integer multiple of the 6.977 hertz frequency was then inputted into an implemented cost function in python, and was then stored into a new array.

The discrepancy curve was then calculated from the cost functions of the comb range, and then plotted with an established window size. The labeled plot below depicts this calculation. It is important to note the change points in the discrepancy curve visually align with the change points in the line height graph. Both are plotted over the same dates as well.

Following the 6.977 hertz comb, this then indicated that our discrepancy curve could now be applied over each comb for both vetted and unvetted lines over O4. As indicated by figure 4, however, the high line count over O4 made the discrepancy plot more difficult to visually interpret. As such, it was necessary to implement a "threshold" value over the discrepancy plot. This was manually chosen as 0.9, for practical purposes.

The next step was to record each discrepancy value that passed the set threshold for each line in a text document with its corresponding date. Given that the recorded O4 vetted and unvetted lines had descriptions attached to that date, this was also iterated through and recorded in the same text file. As such, the project safely concluded with the ability for any user to access this text document and monitor any frequency that occurred over O4.

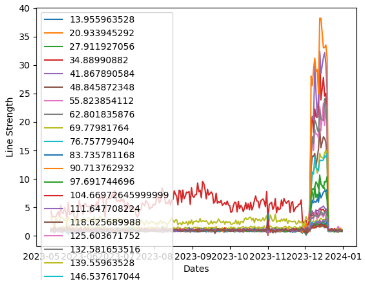

Figure 1: Line Height vs. Dates for 6.977 Hertz Comb

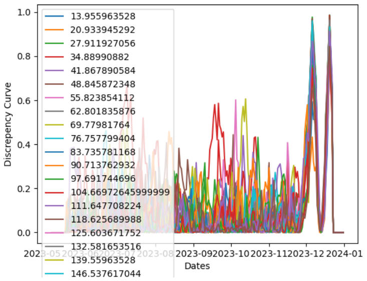

Figure 2: Discrepancy for 6.977 Hertz Comb

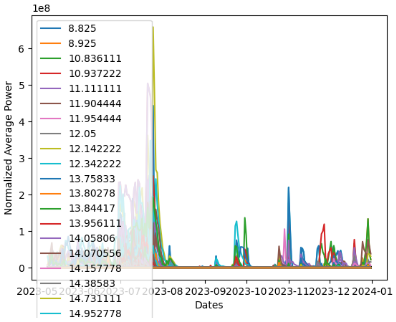

Figure 3: Line Height vs. Dates for O4 Vetted and Unvetted Lines

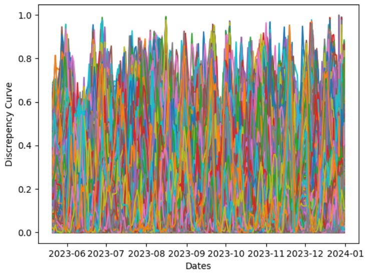

Figure 4: Discrepancy for O4 Vetted and Unvetted Lines

2.3 Challenges and Potential Solutions

One of the underlying challenges of the alternative method is the development of a proper threshold after the arrangement of peak prominence. Determining this threshold must be based nearly entirely on visual interpretation of the change points in the shown line height curve. As such, this manual input may prove to be problematic due to inaccurate visual approximations.

3 Lockloss Tagging and Plugin Development

3.1 Background and Motivation

Lockloss occurs when the suspended mirrors deviate heavily from their nominal, causing them to lose control and disrupting data analysis. Lockloss events themselves are monitored via the Locklost tool plugins, which determine correlating factors to surrounding phenomena which could explain the reason for losing lock. For example, seismic activity may impact mirror deviation.

One phenomena occurring prior to lockloss are glitches. Glitches may be defined as an unknown oscillation in the channel monitoring the output to the lowest suspended mass of the ETMX suspension. There is not an apparent cause for such glitches, and they do not appear to correlate to any any other correlating tag. As such, the establishment of a detection window for these glitches of 2 seconds to 100 milliseconds prior to the lockloss event was necessary.

Furthermore, because a variety of channels may detect a lockloss at differing times from each other, it was also necessary to develop a plugin which would tag for the input mode cleaner (IMC) losing lock at either the same time as other main channels, or 50 milliseconds after. Another instance of channel refinement may also be to establish saturation thresholds for any ASC channels.

3.2 Final Progress

As established in interim report 1, there were complicating factors in the development of an appropriate window-range for the glitch detection. As such, there was a refinement of this window. With strict monitoring from the 2 second to 100 millisecond range prior to lockloss, the threshold was then implemented to check for glitches both above and below the zero-axis. As such, this will not only determine a window prior to lockloss, but furthermore introduce a detection of said glitch with an appropriate threshold. When this threshold is passed, this would then be tagged with "ETM_GLITCH", which was promptly incorporated within a refinement plugin of the Locklost tool. This was refined to only tag when two thresholds are passed, with the initial signal being larger than $3 * 10^5$ counts and then staying within $\pm 1 * 10^5$ counts for 60ms.

Following this, an additional measure was needed to check whether certain IMC channels detect a lockloss at the same time as the IFO. In order to determine this particular change, code was developed to compare the refined GPS times of both events, and tag the IMC with a "SAME" tag if it loses lock at the same detected GPS time of the main IFO, or within 50 milliseconds. This was then promptly tested on all of O4, which led to no detection. However, the tag was triggered over the O3 run, which indicates the tag works correctly. This also indicates the channels lose lock at separate times, especially over the entirety of O4.

The next step was monitoring the entirety of tag counts over the O4 run, including glitch detection and "SAME". This was included in a summary plot, shown below. Furthermore, their counts over the O4 run were also recorded.

Given how successful tagging occurred for O4, we then decided to check for all events over the beginning of O3 to a specified date within O3 and run a summary plot over all tags. This led to an interesting result in which the "SAME" tag appeared far more times in the chosen O3 pool than in O4. Unknown tags remained relatively high. This is indicated in plot figure 7.

In O4, 14% of locklosses were tagged “ETM_GLTICH” but out of the analyzed locklosses in O3, only 4% were tagged “ETM_GLTICH”. Over O4, 0.4% of the locklosses were tagged ”SAME”, with a significant increase of the ”SAME” tag over part of O3, with 23% of all tags were ”SAME”.

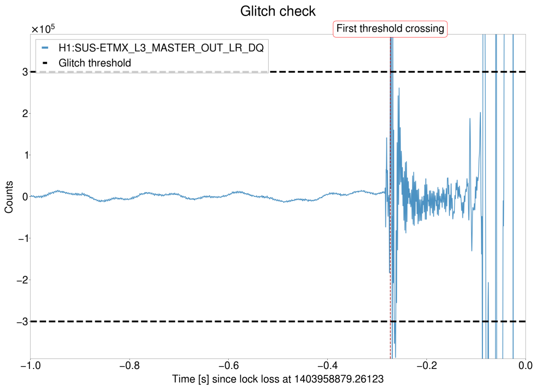

Figure 5: Glitch Detection for event 1403958880

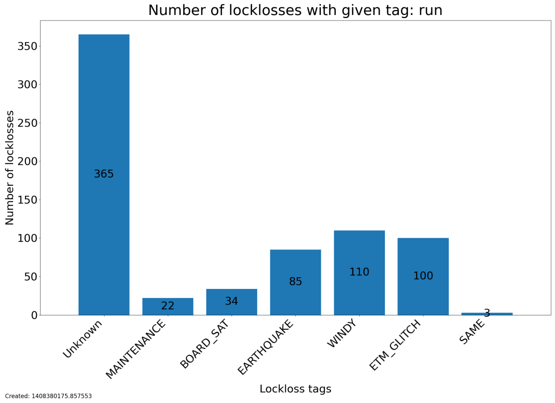

Figure 6: Cumulative Tags over O4 run

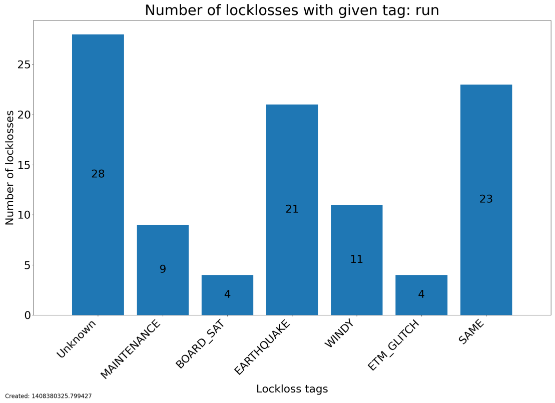

Figure 7: Total Lockloss Tags over Part of O3

3.3 Challenges and Potential Solutions

As indicated in Figures 6 and 7, clear discrepancies in tagging between part of the O3 and all of O4. Specifically, with significant differences for ”ETM_GLITCH” and ”SAME”. An investigation over these results may be to determine the correlation between the causes of glitches prior to lockloss.

4 References

References

[1] C. Truong, L. Oudre, N. Vayatis, Selective review of offline change point detection methods, Signal Processing University Paris, 2019 pp. 1-11

[2] A. Buikema, et. al, Sensitivity and performance of the Advanced LIGO detectors in the third observing run, Physical Review D, American Physical Society, 2020, pp. 1

[3] R.Abbot, et al., GWTC-3: Compact Binary Coalescences Observed by LIGO and Virgo during the Second Part of the Third Observing Run, LIGO Scientific Collaboration, Virgo Collaboration, and KAGRA Collaboration, American Physical Society, 2023, pp. 1

[4] Jameson Rollins Locklost Tool, September 29, 2015, GitHub, GitHub repository, https://github.com/james.rollinson/locklost