Burstegard: a hierarchical clustering algorithm

Tanner Prestegard and Eric Thrane
School of Physics and Astronomy, University of Minnesota, Minneapolis, MN 55455, USA[*]

Gravitational-wave excess power searches are often framed in terms of finding structure in a spectrogram, or ft-map, with a pattern recognition algorithm. Here, we describe one such pattern recognition algorithm, `burstegard`. We demonstrate the ability of this algorithm to identify structure due to simulated gravitational-wave signals in an ft-map.

I. INTRODUCTION

In the quest for the direct detection of gravitational waves, a myriad number of analysis techniques are employed. One technique involves the use of pattern recognition algorithms to process spectrograms, or frequency-time maps (ft-maps), that are created from the raw detector time-series. The success of this method depends on two things: that the statistic being displayed in ft-map form has the ability to differentiate between detector noise and a gravitational-wave signal, and that the pattern recognition algorithm is capable of identifying significant structures in the ft-map.

Since current gravitational-wave interferometers are operated at sensitivities very near potential detection thresholds, it is important that any pattern recognition algorithm utilized be as effective as possible. Therefore, a useful pattern recognition algorithm should be sensitive to signatures of actual gravitational waves, and rather insensitive to features of noise. It is also important that the algorithm be able to complete its task quickly due to the sheer amount of data that must often be processed in searches for gravitational waves. Finally, it is useful for the pattern recognition algorithm to be flexible, i.e., it should be capable of identifying features of somewhat arbitrary shape, as the signal waveform may not always be fully known or understood.

With the above goals in mind, we have developed a hierarchical clustering algorithm, called `burstegard`, that aims to group pixels in an ft-map in order to identify significant structures. In Section II, we describe the `burstegard` algorithm. In Section III, we illustrate the algorithm's capabilities by applying it ft-maps containing example waveforms, and discuss the results. Section IV contains concluding remarks.

II. METHODOLOGY

A. Algorithm

The general purpose of the `burstegard` algorithm is to identify significant features in a frequency-time map of a chosen statistic. Our specific goal is to identify structure[1] due to gravitational waves (GWs), while minimizing the computational time and overhead used. In this vein, we have developed our algorithm in the `C++` programming language, although we note that it is likely possible to implement the ideas shown here in another language, if desired.

The `burstegard` algorithm requires five user-defined parameters:

- *Pixel threshold*: the algorithm will attempt to cluster only pixels that exceed this threshold for the chosen statistic.
- *Radius for clustering*[2]: pixels separated by a distance larger than this parameter will not be clustered together. Distance is defined in units of bins in the frequency-time plane.
- *Minimum number of pixels in a cluster*: a cluster must be composed of at least this many pixels for it to be saved.
- *Time metric*: size of the "clustering ellipse" in the time direction.
- *Frequency metric*: size of the clustering ellipse in the frequency direction.

The time and frequency metrics, along with the clustering radius, define an ellipse for clustering to a particular seed pixel. Pixels are grouped into clusters based on their separation in the ft-map.

B. Findtrack add-on

When performing gravitational-wave data analysis, it is often necessary to identify and flag times and frequencies that are affected by instrumental noise. For example,

[*] prestegard@physics.umn.edu

[1] Let us note here that structure in an ft-map of a particular statistic may be caused by a variety of things, including instrumental effects - it is up to the user to define a robust and useful statistic.

[2] We note that the `burstegard` algorithm is one of several that employs a clustering radius; see also [1, 2]. However, `burstegard` differs from [1] in that it is not a density-based clustering algorithm. Rather, it is designed to target thin, line-like tracks which do not contain high-density regions.

it may be necessary to remove, or "notch" particular frequencies in an ft-map, due to vibrational modes of the detector or seismic noise. However, frequency notching can cause division of a GW signal in an ft-map into multiple pieces, and may reduce the ability of `burstegard` to identify the signal.

To account for this possibility, we have created an add-on to the algorithm, called `findtrack`, which is applied after all individual clusters have been identified. The purpose of this code is to group together multiple clusters that may be separated by a distance larger than the `burstegard` clustering radius, due to frequency notching or other removal of data caused by the presence of instrumental effects. To do this, the user must specify another parameter, the `findtrack` clustering radius, which should be larger than the original `burstegard` clustering radius.

The `findtrack` add-on works by calculating the distance between particular corners of two clusters - if this distance is sufficiently small, those clusters are grouped together. In the case of a "chirp-down"-type signal, `findtrack` tries to connect clusters by considering the distances between the top left and bottom right corners of each cluster (see Fig. 1 for an example of this type of signal). For a "chirp-up"-type signal, `findtrack` tries to connect clusters using the top right and bottom left corners of each cluster. We note here that `findtrack` may only be useful in the limit of large signal power. For signals of marginal power, `findtrack` may increase the background for such a search at a rate greater than the potential improvement in signal sensitivity.

An example of a signal waveform affected by frequency notching, and the effect of `findtrack` on it can be seen in Fig. 1.

C. Clustering Details

A general outline of the `burstegard` algorithm is as follows:

1. Identify all above-threshold pixels in the ft-map.
2. Label each pixel with a "seed number" based on its location in the ft-map, beginning in the upper left corner and running left to right, top to bottom.
3. Set the pixel with the lowest seed number as the seed and try to cluster nearby pixels to it.
4. When the seed has added all nearby above-threshold pixels to the cluster, pick another pixel from the cluster to be the seed and repeat the clustering process.
5. When all pixels in a cluster have been used as the seed, the cluster is complete (it cannot be connected to anything else).
6. Since the cluster cannot be connected to any other clusters (except by using `findtrack` at a later time), all pixels in the cluster are removed from the list of pixels for further clustering.
7. If the number of pixels in this cluster is greater than or equal to the required amount, the cluster is saved.
8. Repeat this process of identifying clusters until all pixels in the ft-map have been used as the seed at some point.
9. Try to group together any saved clusters using `findtrack`.
10. For each saved cluster, calculate the overall "cluster" value for a chosen statistic and return the cluster that has the maximum value of this statistic.

A flow chart of the process is shown in Fig. 2 for illustrative purposes.

III. EXAMPLES

For demonstrative purposes, we consider the cross-power-based Stochastic Transient Analysis Multi-detector Pipeline (STAMP) [3]. STAMP utilizes the cross-correlation of two spatially separated GW interferometers to construct an unbiased estimator $\hat{Y}(t;f)$ of the GW power at a particular time and frequency.

$$\hat{Y}(t;f) = \mathrm{Re}\left[Q_{IJ}(t;f;\hat{\Omega})\hat{C}_{IJ}(t;f)\right] \tag{1}$$

Here, $\hat{C}_{IJ}(t;f)$ is the cross-correlation time-series of the two detectors, and $Q_{IJ}(t;f;\hat{\Omega})$ is a filter function that accounts for the efficiency of the detector pair and the phase delay between them.

An estimator for the variance of $\hat{Y}(t;f)$, called $\hat{\sigma}^2(t;f)$, can be constructed by using the auto-power in each detector at adjacent times. Using this variance, we can define the STAMP cross-correlation SNR $\equiv \hat{Y}/\hat{\sigma}$. Recalling that $\hat{Y}(t;f)$ is an estimator for the GW power in a particular frequency bin at a particular time, we can use the set of $\hat{Y}$ to create a spectrogram for this statistic. Analogously, we can do the same for $\hat{\sigma}$ and the cross-correlation SNR. The cross-correlation SNR is designed so that GW signals will manifest themselves as groups of pixels with positive, relatively large values in the corresponding ft-map. The interested reader should refer to [3] for more details on STAMP formalism.

To test the performance of `burstegard` with STAMP, we inject simulated GW waveforms into Gaussian Monte Carlo noise and try to recover them. In all of the examples shown here, we utilize the following parameters for

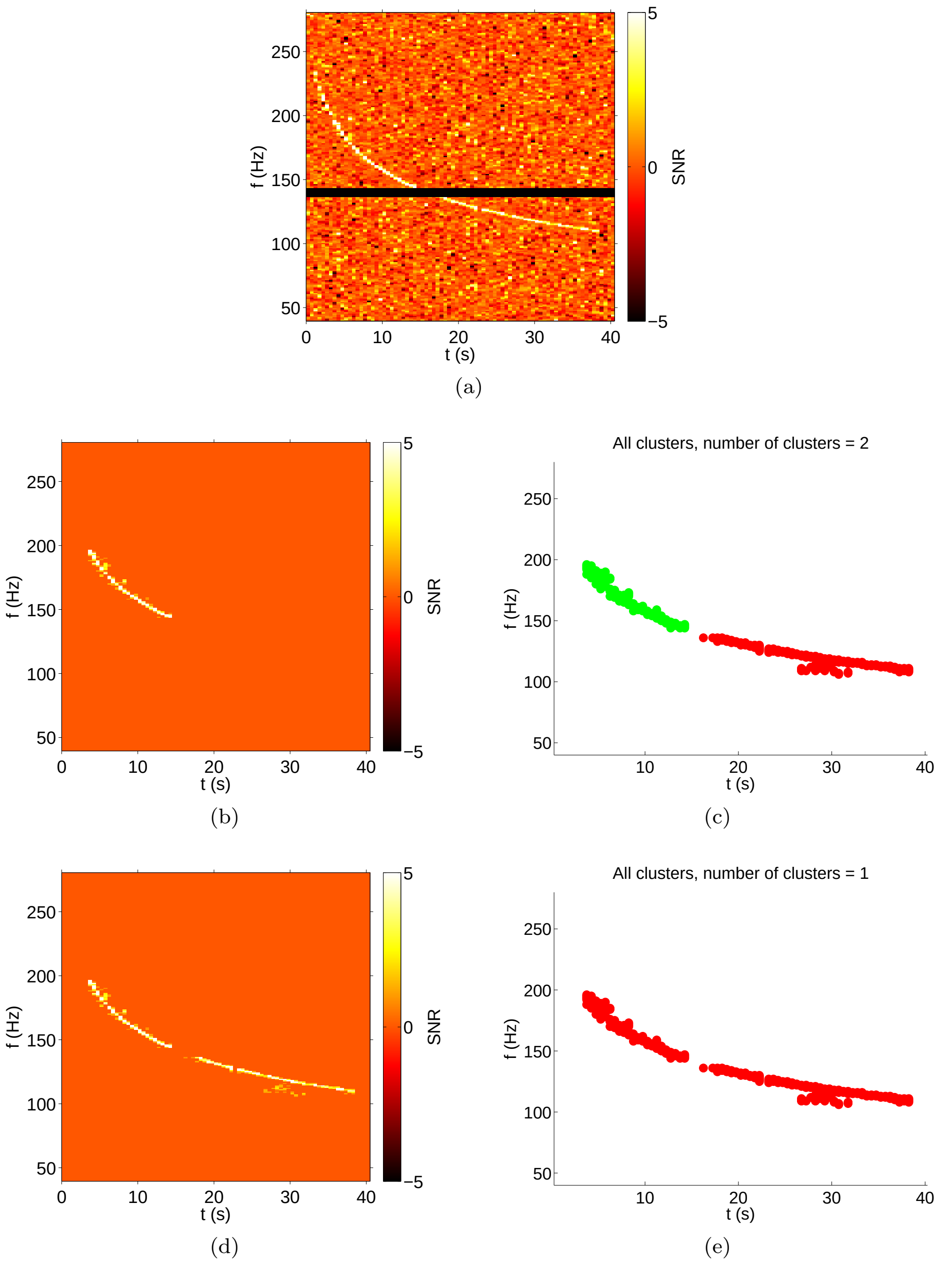


FIG. 1: Example of a simulated GW signal, with frequencies from 137 Hz to 143 Hz notched. Top: ft-map of cross-correlation SNR, showing the waveform and the notched frequencies. Middle-left: `burstegard` recovery without `findtrack`. Middle-right: all clusters identified by `burstegard` without `findtrack`. Different colors indicate independent clusters. Bottom-left: `burstegard` recovery with `findtrack`. Bottom-right: all clusters identified by `burstegard` with `findtrack`. See Section III for more information on the signal waveform and the analysis pipeline used in this example.

`burstegard`, chosen based on testing with STAMP[3]:

- *Pixel threshold:* 0.75
- *Radius for clustering:* 2
- *Minimum number of pixels in a cluster:* 80
- *Time metric:* 1
- *Frequency metric:* 1

The first waveform, displayed in Fig. 3, is based on GW emission due to instabilities in the accretion torus

[3] A more systematic determination of the optimal parameters is possible, and may be done in a similar fashion to previous studies done using a different clustering algorithm with STAMP[1, 4, 5].

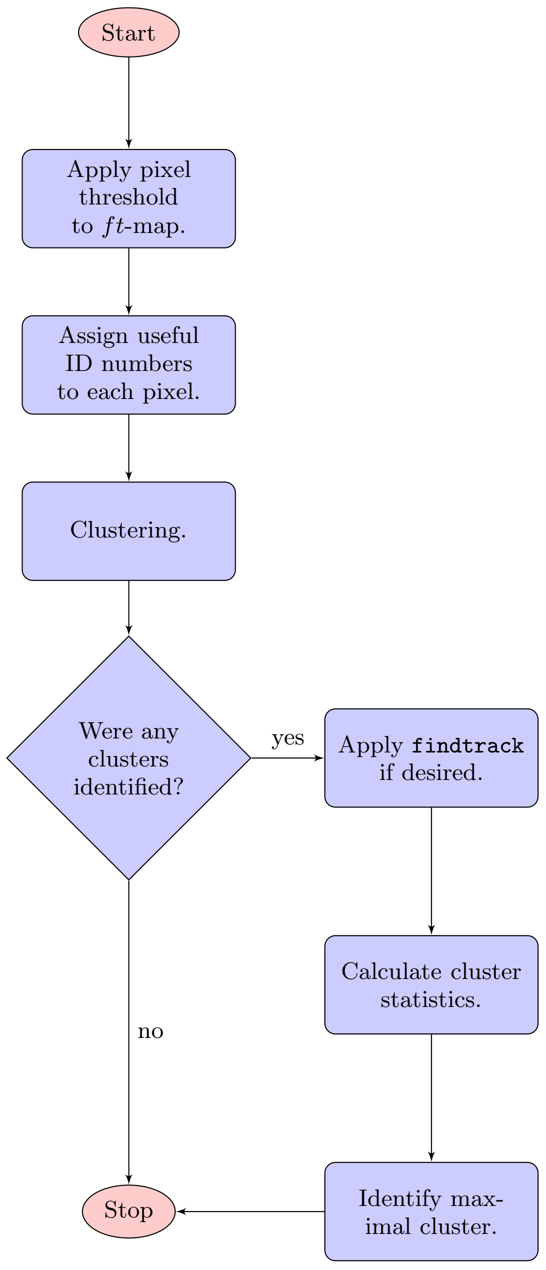

Start
Apply pixel threshold to ft-map.
Assign useful ID numbers to each pixel.
Clustering.
Were any clusters identified?
yes
Apply findtrack if desired.
Calculate cluster statistics.
no
Identify maximal cluster.
Stop

FIG. 2: Flow chart describing the `burstegard` algorithm.

of a black hole [6–8]. This waveform manifests itself as a "chirp-down" track of $\mathcal{O}(100\,\mathrm{s})$ in an ft-map of STAMP cross-correlation SNR.

Comparing the two plots in the top row of Fig. 3, we can see that nearly all of the track corresponding to the ADI signal is recovered. A portion of the tail is not recovered, likely because it dips below the imposed SNR threshold. We also note that few other noise pixels are added to the track. Finally, the bottom plot in Fig. 3 shows only one other identified cluster, which indicates that pure noise is unlikely to form clusters that would be identified by `burstegard`.

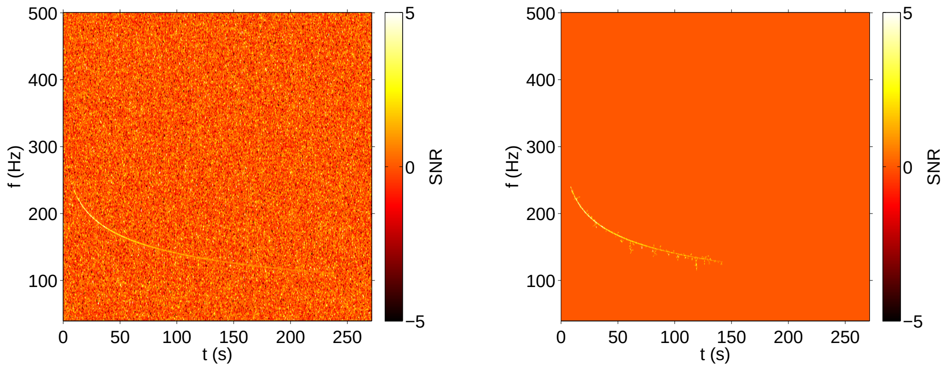
The second waveform tested here is a simulation of the inspiral of an eccentric binary black hole system [9]. This signal is visible as a "chirp-up" of $\mathcal{O}(10\,\mathrm{s})$ in Fig. 4. It is evident that `burstegard` is able to accurately reconstruct the GW signal. It is also important to note that, other than the signal cluster, no other clusters are identified.

From the examples shown, we can draw some conclusions about the effectiveness of the `burstegard` algorithm. In each case, a significant portion of the signal is recovered by `burstegard`, so we conclude that the algorithm is capable of identifying thin, "line-like" curves, and thicker, "cluster-like" groups of pixels. We can also conclude that, due to the small number of extraneous clusters recovered, the `burstegard` algorihm is insensitive to structures arising due to noise fluctuations. We note again that both of these statements are highly dependent on a proper choice of parameters for the `burstegard` clustering algorithm.

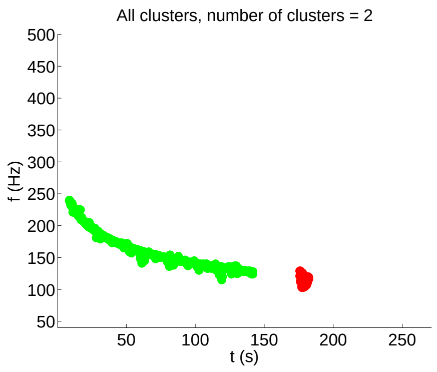
IV. CONCLUSIONS

There is strong motivation for using pattern recognition techniques to identify GW signals in ft-maps and clustering is a particularly useful method, due to its flexibility. We have shown that our clustering algorithm is capable of recovering various GW signals and that it is insensitive to noise. Although the examples shown were based on a particular cross-correlation statistic, we expect that `burstegard` would be effective at identifying GW signals for any well-motivated statistic presented in the form of an ft-map.

[1] Khan R and Chatterji S 2009 *Class. Quantum Grav.* **26** 155009
[2] Raffai P, Frei Z, Márka Z and Márka S 2007 *Class. Quantum Grav.* **24** S457
[3] Thrane E, Kandhasamy S, Ott C D *et al.* 2011 *Phys. Rev. D* **83** 083004
[4] Kandhasamy S 2011 URL `www.ldas-sw.ligo.caltech.edu/ilog/pub/groups/stochastic/logs/2011/images/05/05/shivaraj-1306425734.pdf`
[5] Kandhasamy S 2012 URL `www.ldas-sw.ligo.caltech.edu/ilog/pub/groups/stochastic/logs/2012/images/02/28/shivaraj-1331057458.pdf`
[6] van Putten M 2002 *Astrophys. J. Lett.* **575** 71–74
[7] van Putten M 2001 *Phys. Rev. Lett.* **87** 091101
[8] van Putten M 2008 *Astrophys. J. Lett.* **684** 91
[9] Csizmadia P, Debreczeni G, Rácz I and Vasuth M 2012 *Submitted to Class. Quantum Grav.*

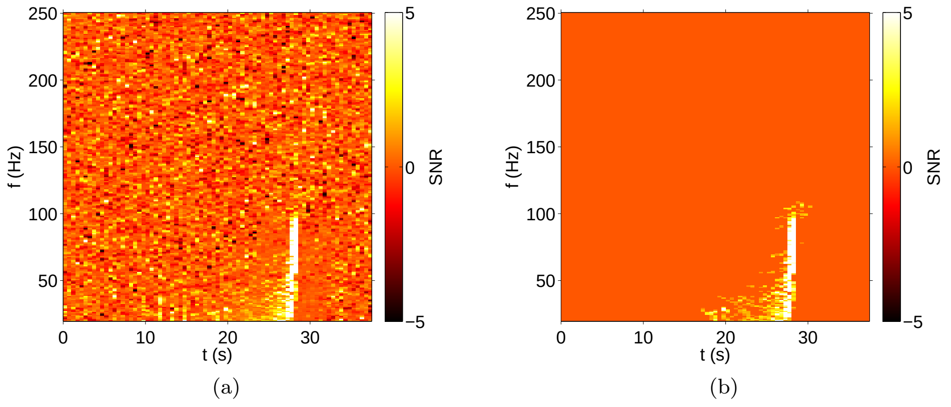


(a) (b)



(c)

FIG. 3: Plots of a simulated accretion disk instability waveform injected into Gaussian Monte Carlo noise. Top-left: SNR ft-map, including injected signal. Top-right: largest cluster recovered by `burstegard`. Bottom: all clusters identified by `burstegard`.



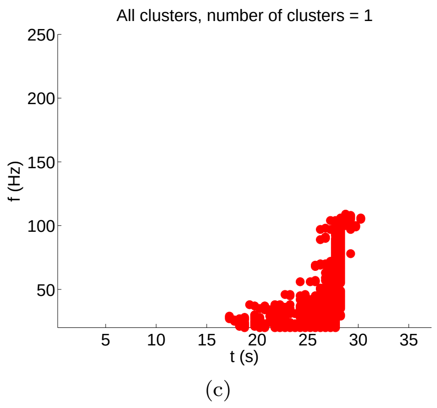


FIG. 4: Plots of a simulated eccentric binary black hole waveform injected into Gaussian Monte Carlo noise. Top-left: SNR ft-map, including injected signal. Top-right: largest cluster recovered by `burstegard`. Bottom: all clusters identified by `burstegard`. Different colors indicate independent clusters.