Autocorrelation and the Fast Fourier Transform

Elijan Jakob Mastnak
Student ID: 28181157

November 2020

Contents

Assignment

1. Analyze the various recordings of two great horned owls' calls found on the course website. Using autocorrelation and cross-correlation, compare the signals among each other, and identify which owl is heard in each recording.

2. *Optional:* Compute the autocorrelation of some other signal(s), either ones you record on your own or some other periodic signals from an existing database.

---

1 Theory

To jump right to the solution, see Section 2.

1.1 Cross-Correlation

The cross-correlation of two periodic signals $f(t)$ and $g(t)$, both with period T, is defined as

$$\phi_{fg}(\tau) = \frac{1}{T}\int_0^T f(t+\tau)g(t)\,\mathrm{d}t$$

The analogous discrete version, for two discrete signals $\{f_k\}$ and $\{g_k\}$ where $k = 0, 1, \ldots, N-1$ is

$$\phi_{fg}(n) = \frac{1}{N}\sum_{k=0}^{N-1} f_{k+n}g_k$$

The cross-correlation is thus a function of the shift n; for shifts where the signals align, the cross-correlation ϕ_{fg} is larger than at neighboring values of n.

1.2 Autocorrelation

If the signals f and g are equal, then $\phi_{fg} \equiv \phi_{ff}$ is called the signal's autocorrelation function, calculated as

$$\phi_{ff}(n) = \frac{1}{N}\sum_{k=0}^{N-1} f_{k+n}f_k \tag{1}$$

The autocorrelation measures the extent to which the signal remains similar to itself as time changes. Local extrema in the autocorrelation function reveal information about the original signal's periodicity—periodic signals have periodic autocorrelation.

For poorly correlated (e.g. noisy) signals, the autocorrelation $\phi_{ff}(n)$ decays to the square $\langle f\rangle^2$ of the signal's average value. In practice, we often work in terms of the modified autocorrelation, defined as

$$\tilde{\phi}_{ff}(n) = \frac{\phi_{ff}(n) - \langle f\rangle^2}{\phi_{ff}(0) - \langle f\rangle^2} \tag{2}$$

Note: Depending on the literature, Equation 2 is the proper expression for autocorrelation, while Equation 1 is called the *autocovariance*—see e.g. [1]. I follow this convention only loosely in this report and use autocorrelation and autocovariance interchangeably. Note that the autocorrelation in Equation 2 is just the autocovariance of a centered signal (i.e. the original signal with its mean subtracted) normalized by the variance $\sigma_f^2 = \phi_{hh}(0) - \langle h \rangle^2$. To see why the normalization factor equals the signal's variance, note that

$$\phi_{ff}(0) - \langle f \rangle^2 = \frac{1}{N} \sum_{k=0}^{N-1} (f_k - \langle f \rangle)^2 \equiv \sigma_f^2$$

Important: The formulas given so far are biased—they implicitly assume the input f is a complete representation of a periodic signal with period N (analogous to the assumption made by the discrete Fourier transform). When working with periodic functions or other functions with long-range order beyond the sample size N, it is best to work with the unbiased implementation

$$\phi_{ff}(n) = \frac{1}{N-n} \sum_{k=0}^{N-n-1} f_{k+n} f_k \tag{3}$$

1.3 Efficient Computation with the Fourier Transform

The summation-based definitions so far have time complexity $\mathcal{O}(N^2)$, which grows prohibitive for large N. There is a more efficient way to compute correlations using the fast Fourier transform (FFT) and the correlation theorem

$$\mathcal{F}\{f \star g\} = \big(\mathcal{F}f\big)^* \cdot \mathcal{F}g$$

which relates the Fourier transform of the correlation $f \star g$ to the product of the Fourier transforms $\mathcal{F}f$ and $\mathcal{F}g$; the asterisk denotes complex conjugation. Taking the relationship's inverse DFT produces

$$f \star g = \frac{1}{N}\mathcal{F}^{-1}\left[\big(\mathcal{F}f\big)^* \cdot \mathcal{F}g\right] \qquad \text{and} \qquad f \star f = \frac{1}{N}\mathcal{F}^{-1}\left[|\mathcal{F}h|^2\right] \tag{4}$$

Using the FFT, these implementations have time complexity $\mathcal{O}(N \log N)$. To find the unbiased autocorrelation of an N-sample signal $\{f_n\}_{n=1}^{N}$ with the FFT, we first zero pad the signal with N zeros to get $\{\tilde{f}_n\}_{n=1}^{2N}$ where

$$\tilde{f}_n = f_n \qquad \text{and} \qquad \tilde{f}_{n+N} = 0 \quad \text{for} \quad n = 0, 1, \ldots, N-1$$

The unbiased autocorrelation is then computed as

$$\phi_{ff}(n) = \frac{1}{N-n}\mathcal{F}^{-1}\big[|\mathcal{F}\tilde{f}|^2\big], \qquad n = 0, 1, \ldots, N-1 \tag{5}$$

1.4 Example: Practical Use of Autocorrelation

Correlation functions are useful for finding periodic signals hidden by noise. For example, consider a signal $f(t) = s(t) + n(t)$ composed of a useful signal $s(t)$ and normally-distributed random noise $n(t)$. If we assume the noise has zero mean, so that its average over a long enough period is zero, a short calculation shows

$$\begin{aligned}
\phi_{ff}(\tau) &= \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} \big[s(t) + n(t)\big] \big[s(t+\tau) + n(t+\tau)\big] \,\mathrm{d}t \\
& \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} \big[s(t)s(t+\tau) + s(t)n(t+\tau) + n(t)s(t+\tau) + n(t)n(t+\tau)\big] \,\mathrm{d}t \\
&= \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} s(t)s(t+\tau) \,\mathrm{d}t + 0 + 0 + 0 \\
&= \phi_{ss}(\tau) \qquad (6)
\end{aligned}$$

In other words, the autocorrelation process removes the effects of random noise.

2 Algorithm Implementations

2.1 Implementations with Direct Summation

The following two Python code blocks show the summation implementations of autocorrelation used in this report. Both functions directly implement Equations 2 and 3 using a nested loop; the modulo operator in line 8 of `autocor_sum_biased` assumes periodicity N in the inputted signal. Note that neither implementation is practically useful; they are included only to measure how slowly they perform relative to the FFT implementations, discussed in Subsection 2.3.

```python
 1 import numpy as np
 2 def autocor_sum_biased(signal):
 3     """ Summation implementation of 1D autocorrelation---assumes periodicity
       ↪  N in inputted signal """
 4     N = np.shape(signal)[0]  # number of samples in inputted signal
 5     autocor = np.zeros(N)  # preallocate
 6     for n in range(N):  # n indexes the autocorrelation
 7         autocor_n = 0.0  # n-th autocor value
 8         for k in range(N - n - 1):  # k is summation index
 9             autocor_n += signal[k]*signal[(k+n) % N]  # (k+n) % N wraps
           ↪  around to signal beginning (assumes periodicity N)
10         autocor[n] = autocor_n/N  # normalize with N
11     return autocor
```

```python
 1 def autocor_sum_unbiased(signal):
 2     """Summation implementation of 1D autocorrelation---doesn't assume
       ↪  periodicity; safe for use with arbitrary signals. """
 3     N = np.shape(signal)[0]  # number of samples in inputted signal
 4     autocor = np.zeros(N) # preallocate
 5     for n in range(N):  # n indexes the autocorrelation
 6         autocor_n = 0.0  # n-th autocor value
 7         for k in range(N - n - 1):  # k is summation index
 8             autocor_n += signal[k]*signal[k+n]
 9         autocor[n] = autocor_n/(N - n)  # normalize with (N_signal - n)
10     return autocor
```

2.2 Fourier Transform Implementation

The following two code blocks show the FFT implementations of autocorrelation from Equations 4 and 5—note that the inputted signal is first zero padded to double the original length in `autocor_sum_unbiased`.

```python
from numpy.fft import fft, ifft, fftshift
def autocor_fft_biased(signal, center=True):
    """ FFT implementation of 1D autocorrelation---assumes input signal is
    ↪  periodic with period N """
    N = np.shape(signal)[0]  # number of samples in inputted signal
    if center: signal = signal - np.mean(signal)  # subtract mean
    return np.real(ifft(np.abs(fft(signal)) ** 2) / N)  # fft magic
```

```python
def autocor_fft_unbiased(signal):
    """ FFT implementation of 1D autocorrelation---zero pads signal and does
    ↪  not assume periodicity N. Safe for arbitrary signals. """
    N = np.shape(signal)[0]  # number of samples in inputted signal
    signal_zp = np.zeros(2*N)  # preallocate zero-padded signal
    signal_zp[0:N] = signal
    auto_cor = ifft(np.abs(fft(signal_zp))**2)[0:N]  # fft magic
    for n in range(N):  # normalize with N - n
        auto_cor[n] = auto_cor[n]/(N-n)
    return np.real(auto_cor)
```

The last code block shows my implementation of the cross-correlation function, which I used in Subsection 4.1 to detect owl calls in noisy signals. To ensure both Fourier transforms have the same dimensions, the inputted signals are zero padded to the same size if necessary. Zero padding didn't seem to negatively affect the end result, which makes sense—the zero-padded portions contribute zero correlation.

```python
def crosscor_fft(f, g):
    """ FFT implementation of cross-correlation of signals f and g. Note that
    ↪  f and g are zero padded to the same length. """
    N_f = np.shape(f)[0]  # number of samples in inputted signal
    N_g = np.shape(g)[0]  # number of samples in inputted signal
    if N_f == N_g:  # signals are the same length
        return np.real(ifft(np.conj(fft(f)) * fft(g)))/N_f  # fft magic
    elif N_f > N_g:  # f is longer than g
        g_padded = np.zeros(N_f)  # zero pad shorter signal to size
        g_padded[0:N_g] = g
        return np.real(ifft(np.conj(fft(f)) * fft(g_padded))) / N_f
    else:  # g is longer than f
        f_padded = np.zeros(N_g)  # zero pad shorter signal to size
        f_padded[0:N_f] = f
        return np.real(ifft(np.conj(fft(f_padded)) * fft(g))) / N_g
```

2.3 Computation Time Comparison

This section compares the computation times of the four autocorrelation implementations discussed above. I measured the average time over 5 runs that each method

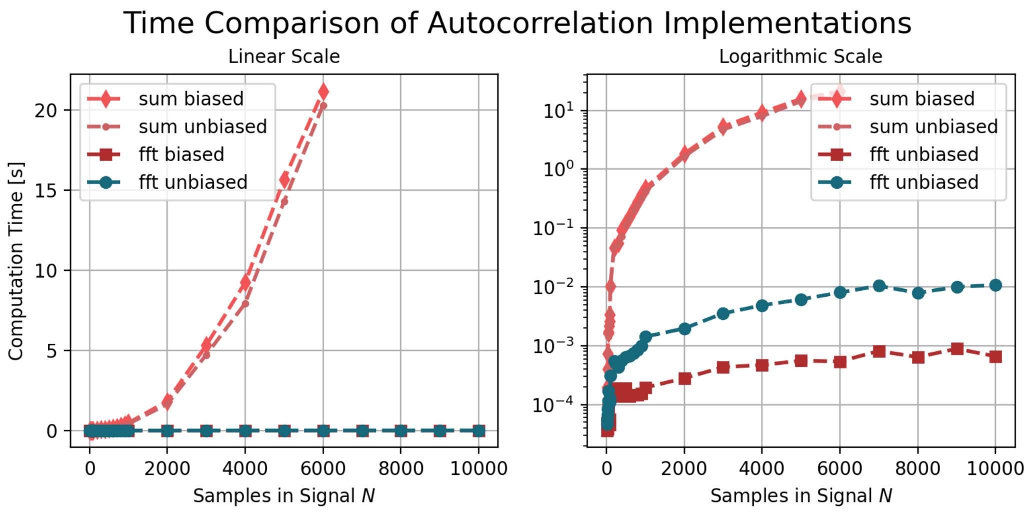

Figure 1: Time required by each implementation to compute the autocorrelation of an N-sample sinusoidal signal. Note the superiority of the FFT methods.

took to compute the autocorrelation of the sinusoidal signal $f(t) = \sin(2\pi \cdot 10 \cdot t)$ sampled at N points on the interval $[0, 10]$. Figure 1 shows the results.
Unsurprisingly, for $N \gtrsim 10^2$, the $\mathcal{O}(N \log N)$ FFT methods superlatively outperform the $\mathcal{O}(N^2)$ summation implementations. The logarithmic scale reveals that the unbiased function `autocor_fft_unbiased` is slower than `autocor_fft_biased` by a constant factor of order roughly one. This makes sense—`autocor_fft_unbiased` works with the Fourier transforms of a doubly-long zero-padded signal.

3 Some Basic Experiments Using Correlation

The goal of this section is to explore the autocorrelation and cross-correlation of simple signals like sinusoids and normally-distributed random noise before moving to the more complex, real-world signals in Section 4.

3.1 Examining Autocorrelation of Gaussian Noise

Figure 2 shows the raw and normalized autocorrelation function of normally distributed random noise with mean $\mu = 0.5$ and variance $\sigma^2 = 1$. Characteristically, the autocorrelation is maximum at zero shift and drops rapidly to zero (for the normalized autocorrelation) and to the squared average μ^2 for the raw autocorrelation.

3.2 Autocorrelation to Detect a Periodic Signal

Figure 3 shows the autocorrelation of uniformly distributed noise in the range $[-1, 1]$ superimposed on a sinusoidal function of amplitude with an exponentially decaying envelope. The autocorrelation capably detects the signal's periodicity even as the signal-to-noise ratio exponentially increases. Note that the pure signal is shown only for reference; the autocorrelation is computed using the noisy gray signal.

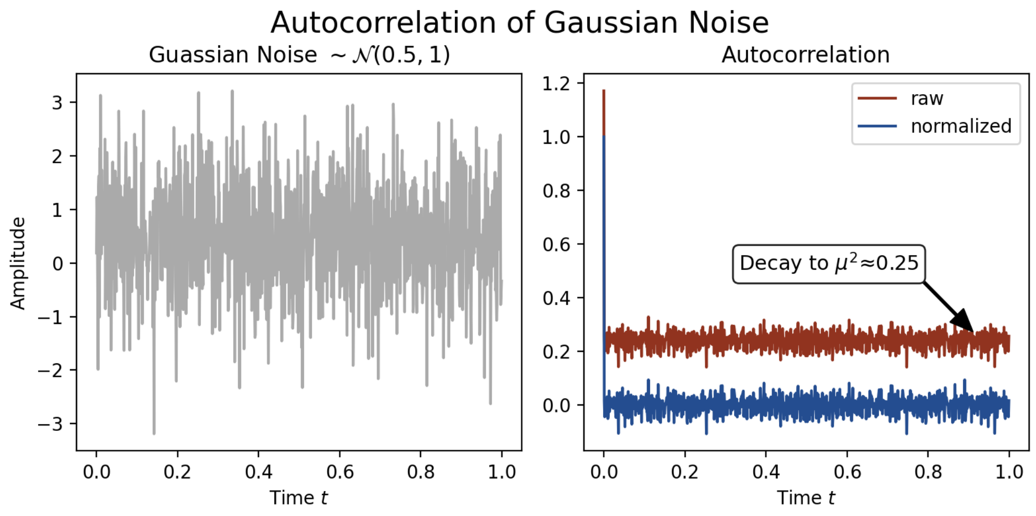

Figure 2: Characteristics of the autocorrelation function of normally distributed random noise. Note the maximum value at zero shift and raw autocorrelation's decay to the mean noise value μ^2 at large shifts. The normalized autocorrelation decays to zero, indicating zero correlation, as expected for noise.

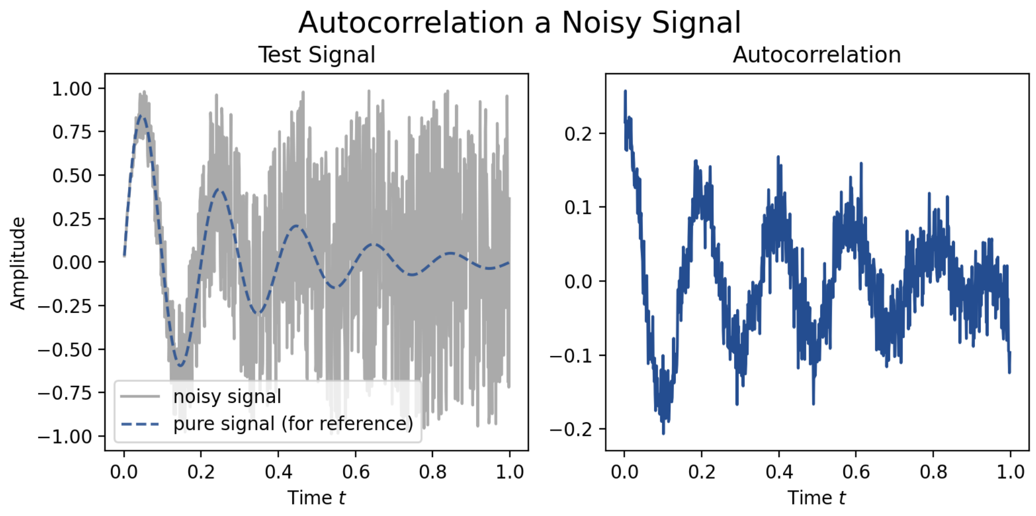

Figure 3: Autocorrelation of an exponentially decaying sine function (the pure signal is shown in dotted blue for reference) masked with uniformly distributed noise. The autocorrelation detects periodicity even at a large SNR, when doing so visually would be difficult.

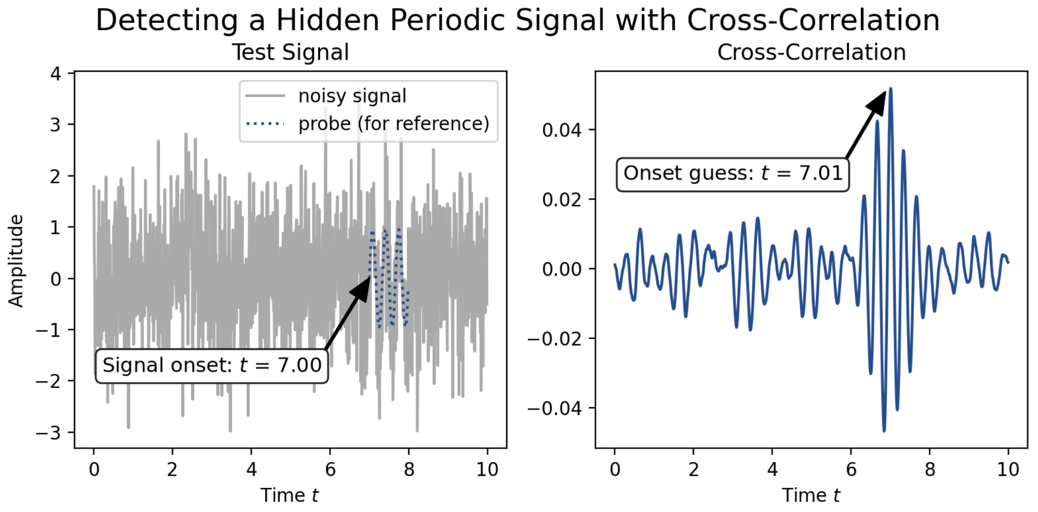

Figure 4: Using cross-correlation to detect the onset of a weak periodic signal within normally distributed random noise $\sim \mathcal{N}(0, 1)$. The signal onset corresponds the the cross-correlation's maximum. The correct position of the "probe" signal is shown in dashed blue for reference.

3.3 Cross-Correlation to Detect a Periodic Signal

In this example, I first superimposed a short, weak sine wave on normally distributed random noise with mean $\mu = 0$ and variance $\sigma^2 = 1$, shown in Figure 4. I then used the pure sine signal as a cross-correlation "probe" to detect the onset of the weak periodic signal within the noisy signal. Note that, as before, the pure probe signal is shown only for reference; the autocorrelation is calculated on the noisy gray signal. I found the results quite impressive—if the probe signal were not retrospectively highlighted for reference, visually identifying its position within the noisy signal seems hopeless, but the cross-correlation function does so definitively. I used a similar approach in Subsection 4.1 to detect owl calls.

4 Identifying Eurasian Eagle Owls with Correlation

First, some background: we are given the "clean", noiseless calls of two different Eurasian eagle owls in 44 100 Hz, single-channel `wav` format, along with four records of the same two owls in noisy natural environments. The goal is to detect which owl in the noisy recordings is which based on the "clean" reference calls. For orientation, a quick description of each recording follows below, while Figure 5 shows the corresponding waveforms in the time domain.

- `bubomono.wav` and `bubo2mono.wav`: 5-second clips containing the first and second owl's call, respectively. The calls themselves last from about 0.5 s to 1.0 s; the rest of each file is faint background noise.
- `bubo1-short.wav` and `bubo2-short.wav`: 0.5 s trimmed versions of `bubomono.wav` and `bubo2-short.wav` containing only the owl calls.

- `mix.wav`: Light background noise including crickets and the sound of a person walking through brush and dry leaves. The owl call is heard clearly.
- `mix1.wav`: Moderate background noise including a burbling creek and cricket song. The owl call is heard well, but not as clearly as in `mix.wav`.
- `mix2.wav`: Loud background noise including a running river and gentle crickets. The owl call is faint.
- `mix22.wav`: Dominant background noise from a running river. The owl call is hardly audible.

Each `mix` file is a 5-second recording with the owl call near the 0.5 s mark.

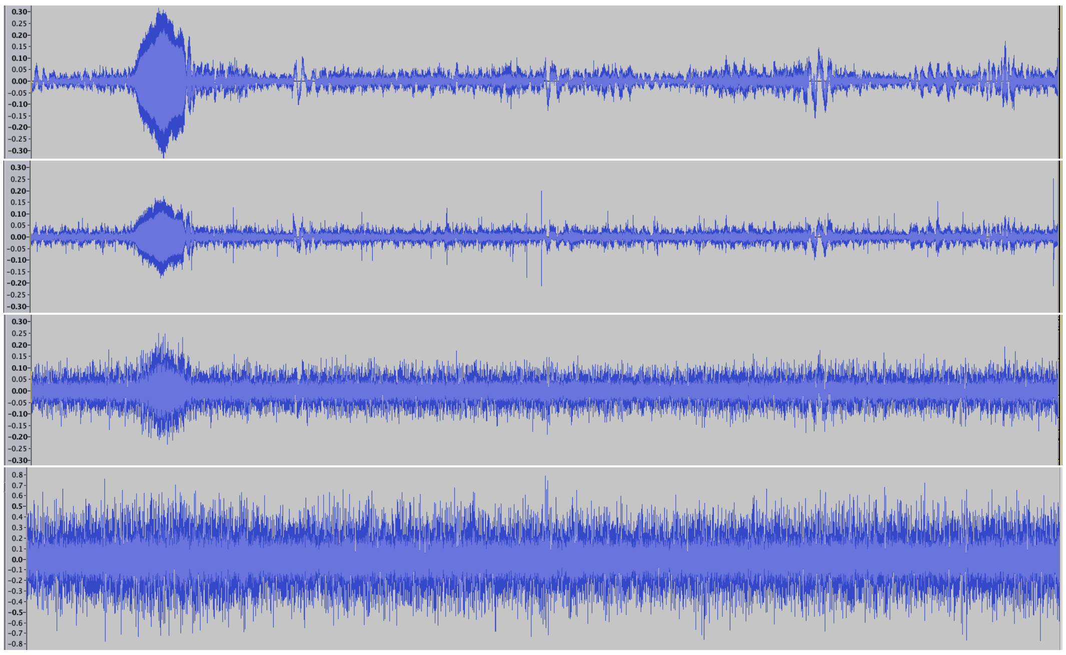

Figure 5: Time-domain waveforms, in order from top to bottom, of the recordings `mix`, `mix1`, `mix2`, `mix22`. Note how the owl call—the short peak near the beginning of each clip—is progressively dominated by background noise in successive recordings.

4.1 Cross-Correlation Analysis

This is the first of two approaches to identifying the owl in each noisy signal. The premise is simple: I used the "clean" reference recordings `bubo1.wav` and `bubo2.wav` as cross-correltion "probes" to analyze each `mix` recording. The reference recording producing a stronger cross-correlation response would identify the owl in each noisy signal. The method worked impressively—see Figures 6 and 7 for results. For perspective, in the noisiest recording `mix22`, the listener is hard-pressed to make out a hoot in the first place, let alone identify the correct owl—just look at the last waveform in Figure 5, which appears to be pure noise. But the cross-correlation in Figure 7 performs the identification capably. I was impressed, to say the least!

4.2 Spectral Analysis

This approach works by identifying the characteristic frequency of each owl from the clean reference calls and then searching for the presence of the characteristic frequency in the spectrum of each noisy recording.

4.2.1 Spectrum of Pure Owl Calls

First, I plotted the spectrum of the reference signals `bubo1-short.wav` and `bubo2-short.wav`. Each owl call displays a clear and distinct fundamental frequency—380 Hz for `bubo1` and 335 Hz for `bubo2`—along with higher harmonics at integer multiples of the fundamental. Figure 8 shows both spectra in linear and logarithmic scale; the latter is useful when identifying the higher harmonics.

4.2.2 Analysis of Signal and Autocorrelation Spectrum

The premise of this method is straightforward: compute the spectrum of each noisy signal and search for the presence of a fundamental frequency from Figure 8 that would uniquely identify the owl, either 380 Hz for `bubo1` or 335 Hz for `bubo2`. There is a slight caveat—I found used the spectra of both the raw signal and the signal's autocorrelation function.

Why use the autocorrelation's spectrum? My thinking was: each noisy signal contains a characteristic owl frequency, which should be picked up by the autocorrelation function (the autocorrelation of a periodic signal is periodic with the same period), with a slight bonus—the autocorrelation should remove random noise from each noisy signal (see e.g. the derivation in Equation 6). Figure 9 shows how the autocorrelation of the noisy signal `mix1.wav` is indeed periodic; the other `mix` files all display periodicity, and I left them out to avoid over-cluttering the report with figures.

The spectral approach appears to correctly identify the owl in each recording, although the results are not as definitive as when using cross-correlation detection. Note that the autocorrelation's spectrum is indeed less noisy than the raw signal's spectrum, but only up to a constant factor of order 1. I was impressed by both spectra picking up nearly identical frequencies, indicating that the autocorrelation function successfully preserves the dominant frequencies in the noisy signal.

4.3 Quick Summary of Results

Table 4.3 summarizes the identification results for each of the three methods used in the report. All of methods agree on their identification for each noisy file.

Method	`mix`	`mix1`	`mix2`	`mix22`
Cross-Correlation	`bubo1`	`bubo1`	`bubo1`	`bubo2`
Signal Spectrum	`bubo1`	`bubo1`	`bubo1`	`bubo2`
Autocorrelation Spectrum	`bubo1`	`bubo1`	`bubo1`	`bubo2`

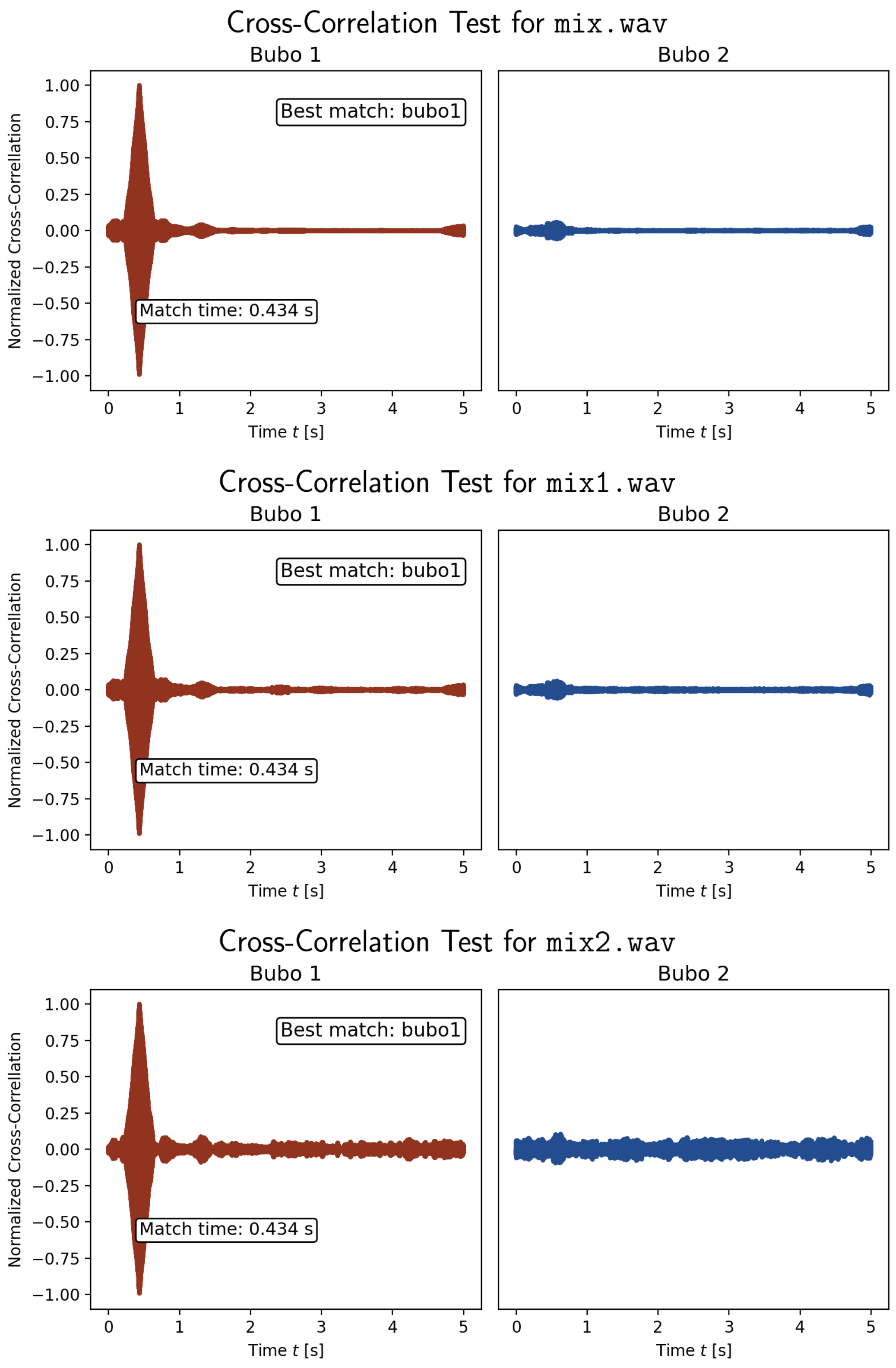

Figure 6: Identifying the owl in the noisy recordings `mix`, `mix1` and `mix2` via cross-correlation with the clean reference calls.

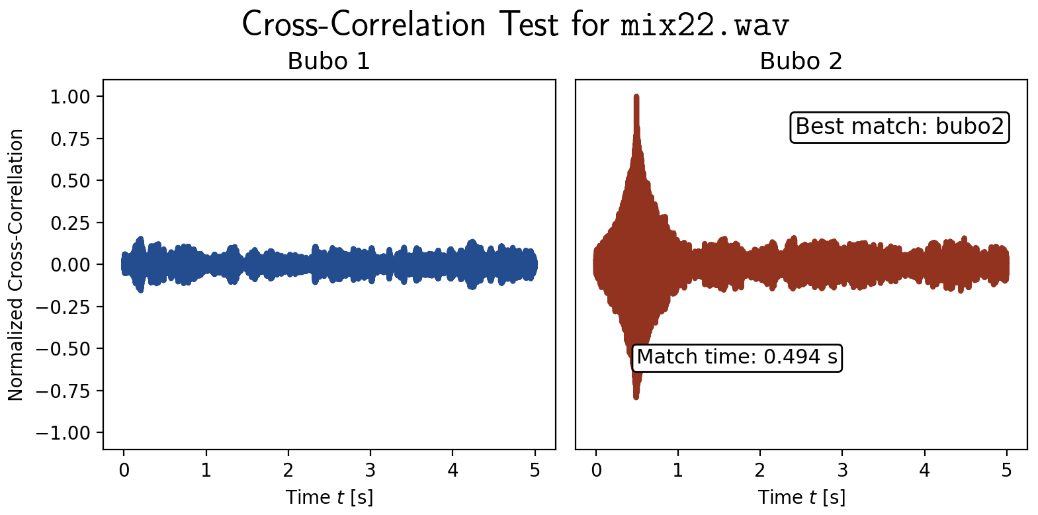

Figure 7: Identifying the owl `bubo2` in the noisy recording `mix22` using cross-correlation with the clean reference calls.

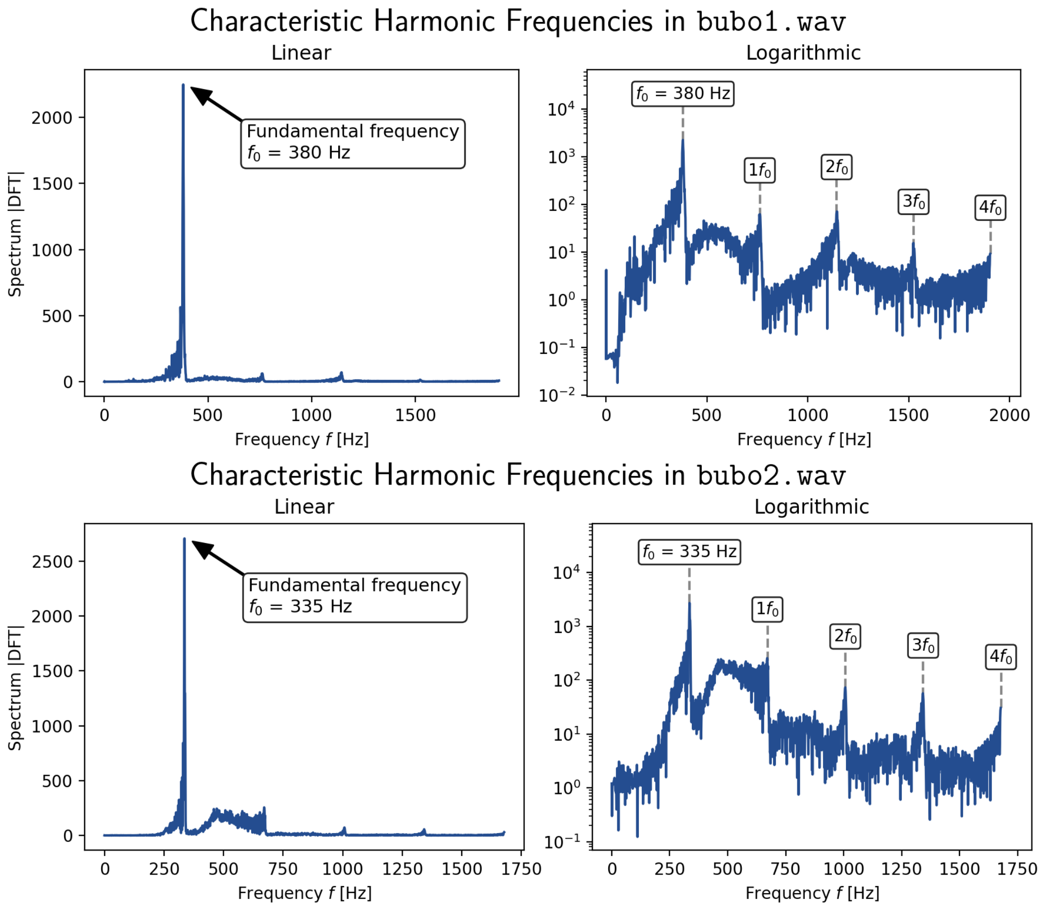

Figure 8: Both owl calls display clear and distinct fundamental frequencies—either 380 Hz for `bubo1` or 335 Hz for `bubo2`. These frequencies are used later as "acoustic fingerprints" to identify the owls in noisy recordings.

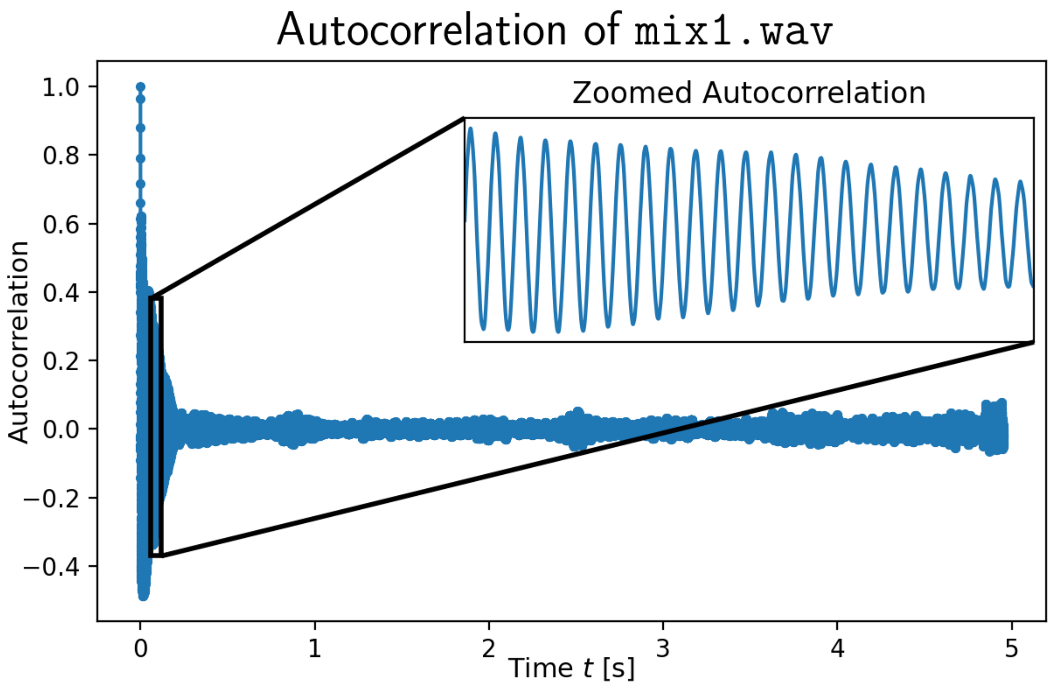

Figure 9: The autocorrelation function of `mix1.wav`. The zoomed portion is intended to show how the autocorrelation function is periodic, presumably at the characteristic frequency of the owl call, which forms the basis of the spectral analysis in Figures 11 and 10.

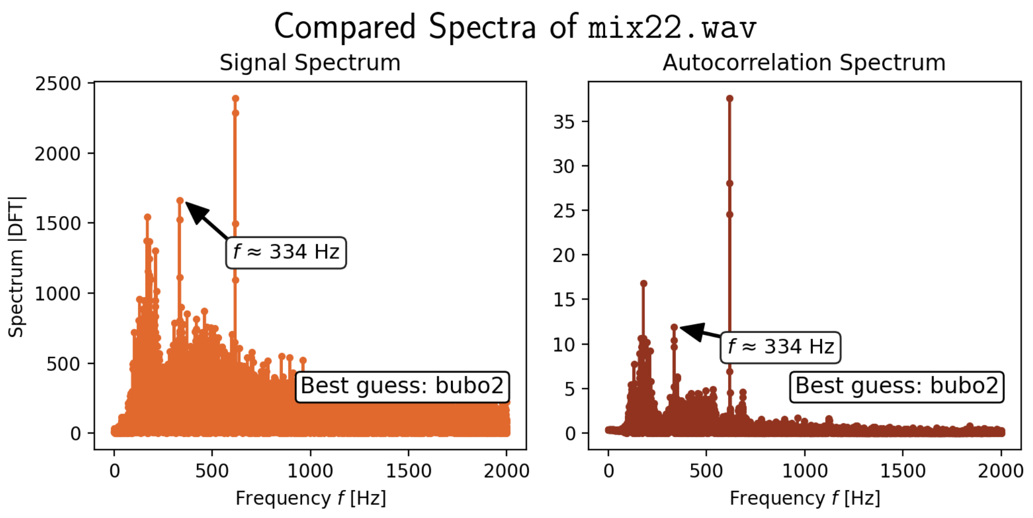

Figure 10: The spectra of the raw noisy signal and the signal's autocorrelation function both display the 335 Hz characteristic frequency of `bubo2` in the recording `mix22`.

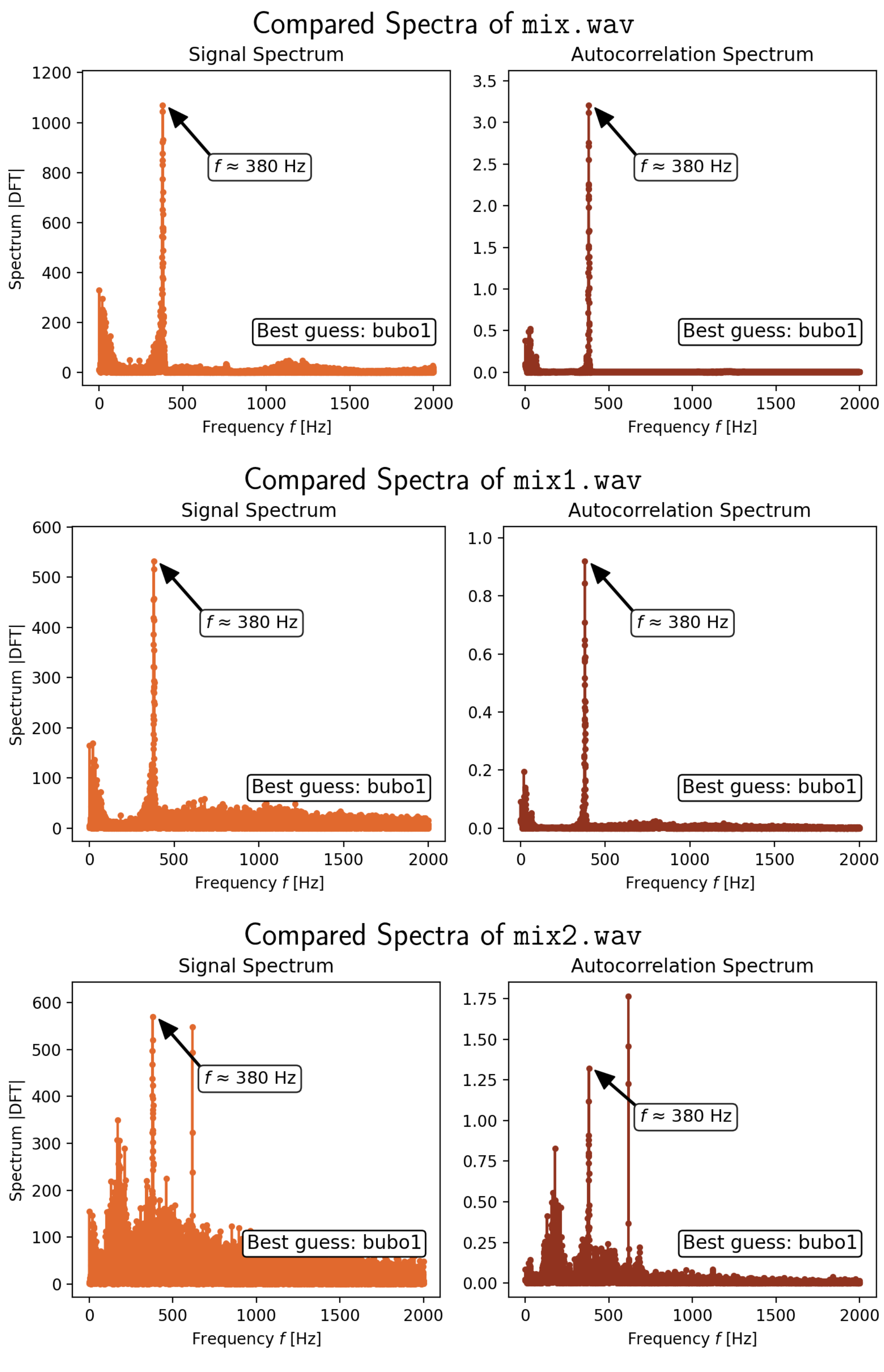

Figure 11: The spectra of the raw noisy signal and the signal's autocorrelation function both display the characteristic frequency 380 Hz characteristic frequency of `bubo1` in the recordings `mix`, `mix1` and `mix2`.

5 Extra: Extracting Guitar Notes from a Noisy Background

As an extra, I recorded (in 44 100 Hz sample rate, single-channel, `wav` format) individual reference samples of three notes played on an acoustic guitar—the `E3`, `A3` and `G4` notes to use as reference “probe signals”, and then recorded the same notes played in the sequence

```
E3    G4    A3
```

with the notes spaced at roughly one second intervals. I'll refer to this recording at `sequence-clean.wav`.[1]

Next, inspired by the `bubo` recordings, I superimposed various nature sounds (songbirds, ocean waves crashing on a beach, and a running river) on the clean signal to generate the noisy files `sequence-waves.wav`, `sequence-birds-river.wav`, and `sequence-birds-waves.wav`.[2] To push the limits of correlation detection, I increased the noise amplitude well above the actual signal amplitude (at least on par with the noisiest owl recording `mix22.wav`)—in `sequence-birds-waves`, for example the `E` and `A` notes are hardly audible.

I then performed an analogous cross-correlation analysis to those shown and discussed in Figure 4 and Subsection 4.1; the goal was to detect the onset of each note in the noisy signal. Figures 12 through 14 show the results. The method works well, generally detecting the note's onset time to withing a few milliseconds (the reference value was determined from `sequence-clean.wav` in the audio editing and waveform analysis program `Audacity`). Worth noting, however, is that the method failed for the first `E3` note in the recordings `sequence-waves.wav` and `sequence-birds-waves.wav`. Whether the failure is a fault of the cross-correlation method in principle or my overzealous attempts to push the method to its limit by making the `E3` note excessively faint relative to the dominating background noise is up to debate—I encourage the reader to give the recordings `sequence-birds-waves.wav` and `sequence-waves.wav` a listen and decide for him or herself.

[1] As a side note, I intentionally played the notes as string harmonics at the 12th fret to produce a weaker signal (and thus push the limits of the correlation detection), but that is a non-essential musical detail.

[2] The samples are included, along with the usual source code, in the `zip` file accompanying the report.

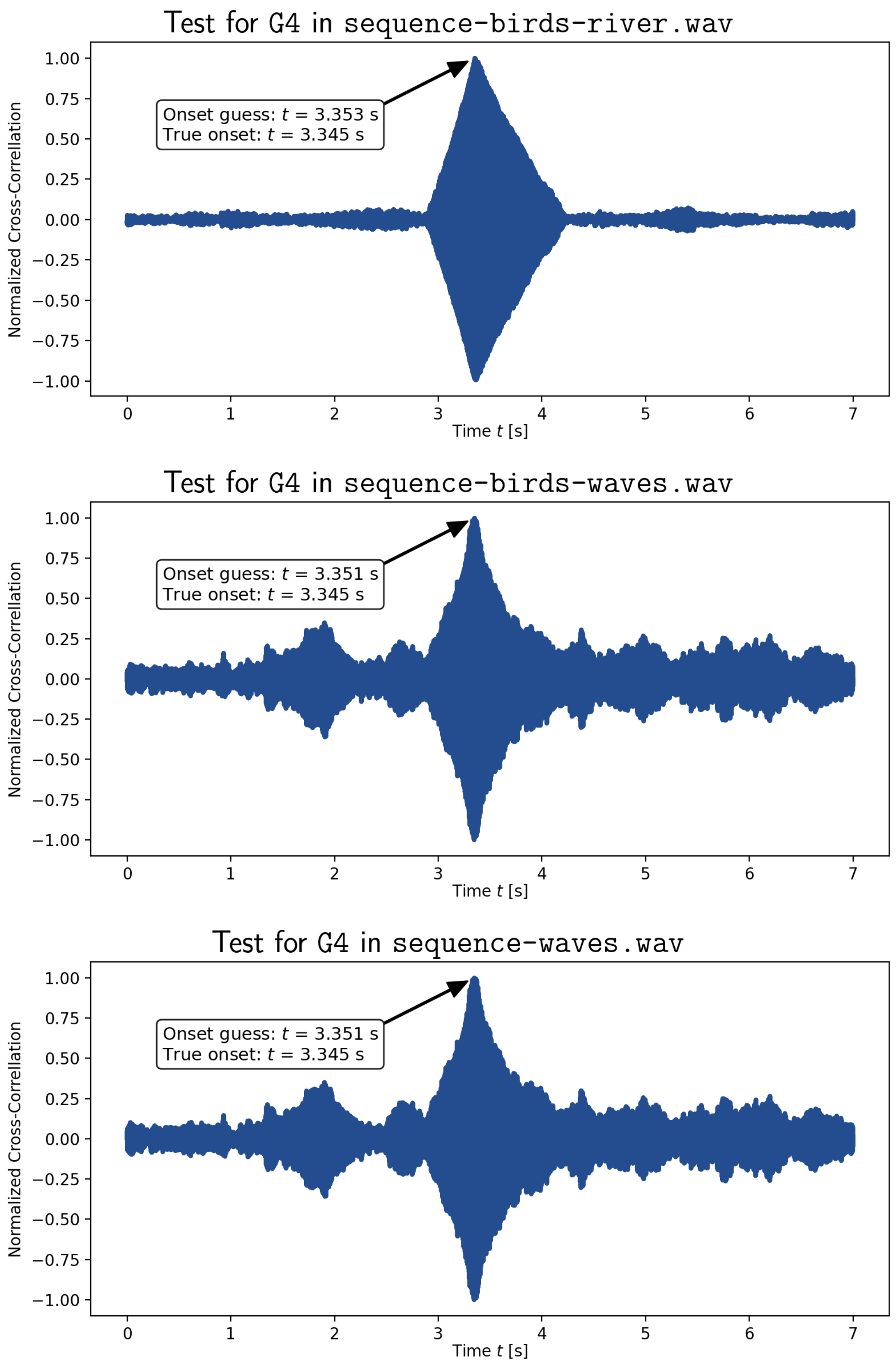

Figure 12: Using cross-correlation to detect the onset of a faint `G4` note in a noisy background.

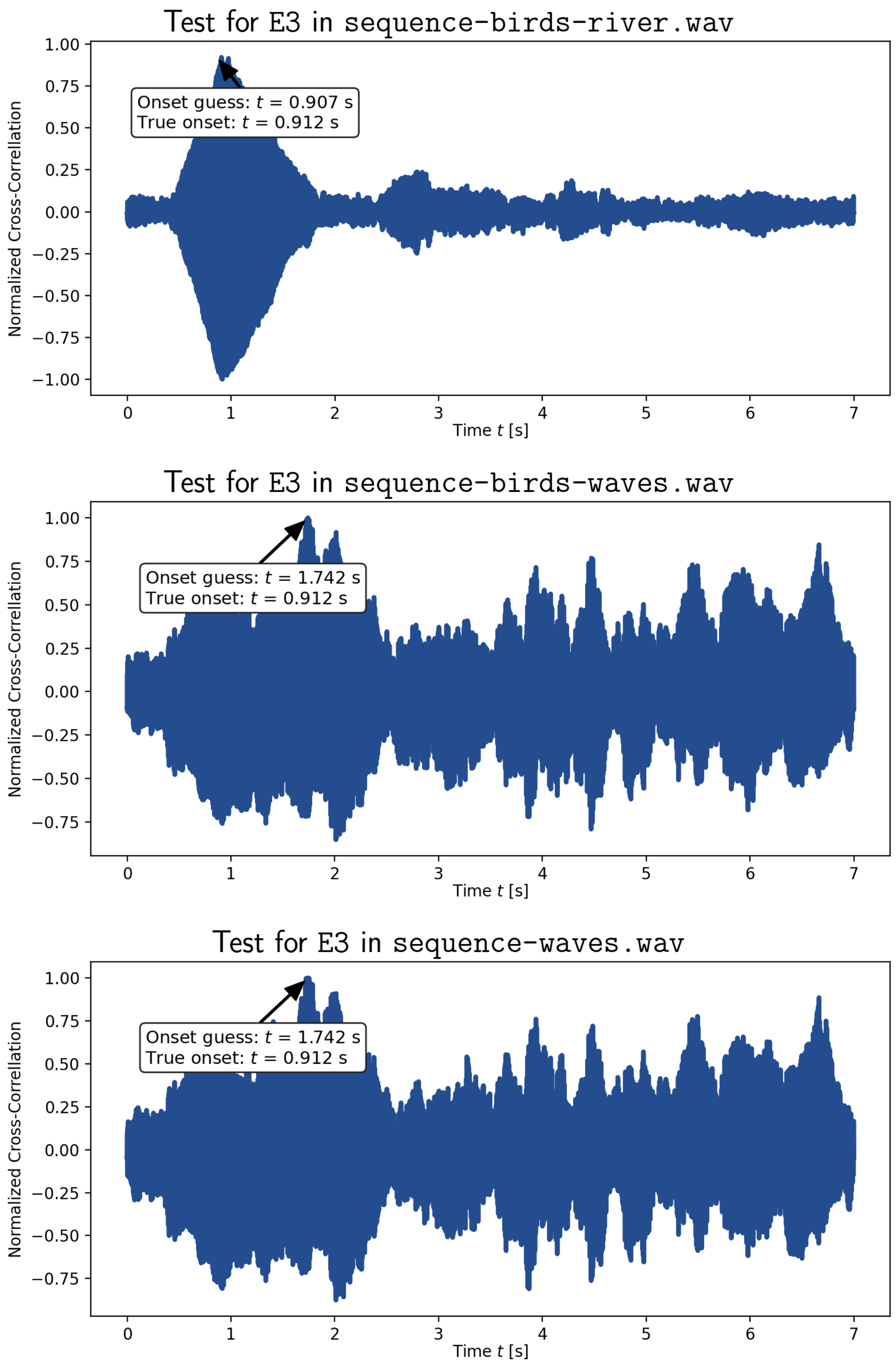

Figure 13: Using cross-correlation to detect the onset of a faint `E3` note in a noisy background. Note the method's failure in the last two recordings, but keep in mind that the `E3` note in particular is excessively faint.

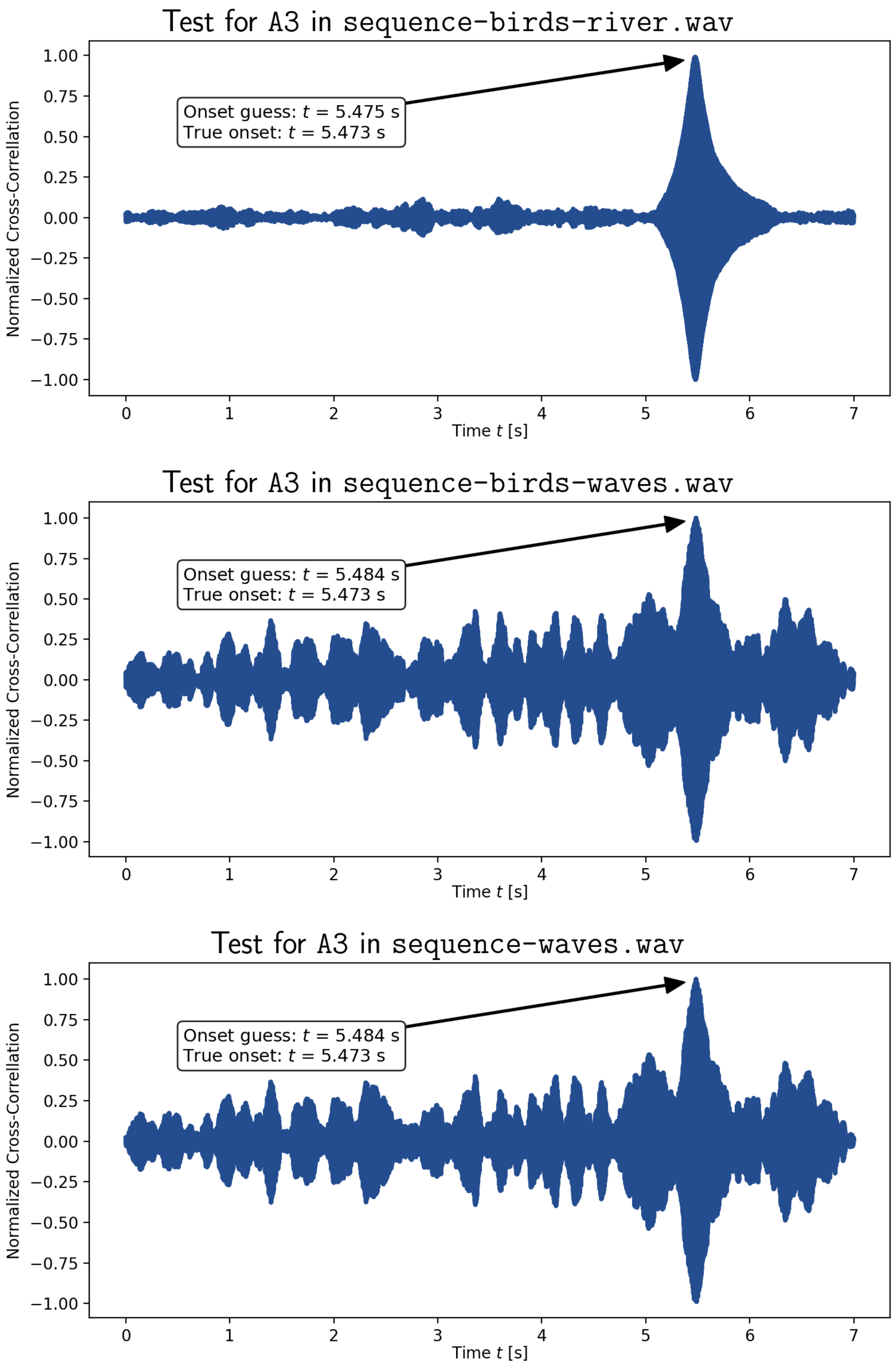

Figure 14: Using cross-correlation to detect the onset of a faint `A3` note in a noisy background.

References

[1] Wikipedia contributors. “Autocorrelation.” *Wikipedia, The Free Encyclopedia.* 25 October 2020. https://en.wikipedia.org/wiki/Autocorrelation#Normalization

[2] Derek Rowell. Lecture 22. In *Signal Processing - Continuous and Discrete—MIT Course No. 2.161.* Cambridge MA, 2008. https://ocw.mit.edu/courses/mechanical-engineering/2-161-signal-processing-continuous-and-discrete-fall-2008/lecture-notes/lecture_22.pdf

[3] Thibauld Nion. “Autocorrelation functions.” 2012. https://etudes.tibonihoo.net/literate_musing/autocorrelations.html.

[4] Martin Horvat. “Avtokorelacijska funckcija”. 2005. https://ucilnica.fmf.uni-lj.si/pluginfile.php/62334/course/section/6936/avtokorelacije.nb.pdf.