Monitoring and Mining Insect Sounds in Visual Space

Yuan Hao Bilson Campana Eamonn Keogh
University of California, Riverside
{yhao, bcampana, eamonn}@cs.ucr.edu

ABSTRACT

Monitoring animals by the sounds they produce is an important and challenging task, whether the application is outdoors in a natural habitat, or in the controlled environment of a laboratory setting.

In the former case the density and diversity of animal sounds can act as a measure of biodiversity. In the latter case, researchers often create control and treatment groups of animals, expose them to different interventions, and test for different outcomes. One possible manifestation of different outcomes may be changes in the bioacoustics of the animals.

With such a plethora of important applications, there have been significant efforts to build bioacoustic classification tools. However, we argue that most current tools are severely limited. They often require the careful tuning of many parameters (and thus huge amounts of training data), they are too computationally expensive for deployment in resource-limited sensors, they are specialized for a very small group of species, or they are simply not accurate enough to be useful.

In this work we introduce a novel bioacoustic recognition/classification framework that mitigates or solves all of the above problems. We propose to classify animal sounds in the *visual space*, by treating the texture of their spectrograms as an acoustic fingerprint using a recently introduced parameter-free texture measure as a distance measure. We further show that by searching for the most representative acoustic fingerprint we can significantly outperform other techniques in terms of speed and accuracy.

Keywords

Classification, Spectrogram, Texture, Bioacoustics

1. INTRODUCTION

Monitoring animals by the sounds they produce is an important and challenging task, whether the application is outdoors in a natural habitat [4], or in the controlled environment of a laboratory setting.

In the former case the density and variety of animal sounds can act as a measure of biodiversity and of the health of the environment. Algorithms are needed here not only because they are (in the long term) cheaper than human observers, but also because in at least some cases algorithms can be more accurate than even the most skilled and motivated observers [21].

In addition to field work, researchers working in laboratory settings frequently create control and treatment groups of animals, expose them to different interventions, and test for different outcomes. One possible manifestation of different outcomes may be changes in the bioacoustics of the animals. To obtain statistically significant results researchers may have to monitor and hand-annotate the sounds of hundreds of animals for days or weeks, a formidable task that is typically outsourced to students [23].

There are also several important commercial applications of acoustic animal detection. For example, the US imports tens of billions of dollars worth of timber each year. It has been estimated that the inadvertent introduction of the Asian Longhorn Beetle (*Anoplophora glabripennis*) with a shipment of lumber could cost the US lumber industry tens of billions of dollars [22]. It has been noted that different beetle species have subtlety distinctive chewing sounds, and ultra sensitive sensors that can detect these sounds are being produced [17]. As a very recent survey of acoustic insect detection noted, "*The need for nondestructive, rapid, and inexpensive means of detecting hidden insect infestations is not likely to diminish in the near future*" [22].

With such a plethora of important applications, there have been significant efforts to build bioacoustic classification tools [4]. However, we argue that current tools are severely limited. They often require the careful tuning of many parameters (as many as eighteen [8]) and thus huge amounts of training data, they are too computationally expensive for use with resource-limited sensors that will be deployed in the field [7], they are specialized for a very small group of species, or they are simply not accurate enough to be useful.

In this work we introduce a novel bioacoustic recognition/classification framework that mitigates or solves all of the above problems. We propose to classify animal sounds in the *visual space*, by treating the texture of their spectrograms as an acoustic "fingerprint" and using a recently introduced parameter-free texture measure as a distance measure. We further show that by searching for the smallest representative acoustic fingerprint (inspired by the *shapelet* concept in time series domain [28]) in the training set, we can significantly outperform other techniques in terms of both speed and accuracy.

Note that monitoring of animal sounds in the wild opens up a host of interesting problems in sensor placement, wireless networks, resource-limited computation [7], etc. For simplicity, we gloss over such considerations, referring the interested reader to [4] and the references therein. In this work we assume all such problems have been addressed, and only the recognition/classification steps remain to be solved.

2. RELATED WORK / BACKGROUND

2.1 A Brief Review of Spectrograms

As hinted at above, we intend to do recognition/classification in the visual space, by examining the *spectrogram* of the animal sounds. As shown in Figure 1, a spectrogram is a time-varying spectral representation that shows how the spectral density of a signal varies with time.

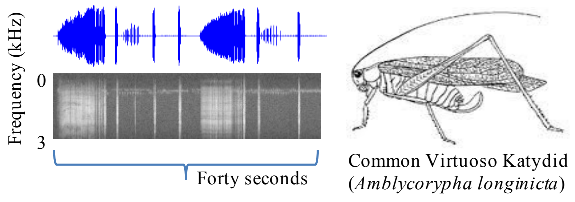

Figure 1: A spectrogram of the call of an insect. Note the highly repetitious nature of the call. In this case, capturing just two "busts" may be sufficient to recognize the insect

There is a huge amount of literature leveraging off *manual* inspection of such spectrograms; see [12] and the references therein for some examples. However, as we shall see, algorithmic analysis of spectrograms remains an open problem, and an area of active research. Beyond the problems that plague attempts to define a distance measure in any domain, including invariance to offset, scaling, uniform scaling, non-uniform warping, etc., spectrograms almost always have significant noise artifacts, even when obtained in tightly controlled conditions in a laboratory setting. One avenue of research is to "clean" the spectrograms using various techniques [2], and then apply shape similarity measures to the cleaned shape primitives. Some types of specialized cleaning may be possible; for example, removing the 60Hz noise is commonly encountered[1]. However, algorithms to robustly clean general spectrograms seem likely to elude us for the foreseeable future.

As we shall see in Section 3, our solution to this problem is to avoid any type of data cleaning or explicit feature extraction, and use the raw spectrogram directly.

[1] American domestic electricity is at 60Hz (most of the rest of the world is 50Hz) and inadequate filtering in power transformers often allows some 60Hz signal to bleed into the sound recording.

2.2 General Animal Sound Classification

The literature on the classification of animal sounds is vast; we refer the interested reader to [20][1] for useful surveys. At the highest level, most research efforts advocate extracting sets of features from the data, and using these features as inputs to standard classification algorithms such as a decision tree, a Bayesian classifier or a neural network. As a concrete representative example, consider [24], which introduces a system to recognize *Orthoptera* (the order of insects that includes grasshoppers, crickets, katydids[2] and locusts). This method requires that we extract multiple features from the signal, including *distance-between-consecutive-pulses*, *pulse-length*, *frequency-contour-of-pulses*, *energy-contour-of-pulses*, *time-encoded-signal-of-pulses*, etc. However, robustly extracting these features from noisy field recordings is non-trivial, and while these features seem to be defined for many *Orthoptera*, it is not clear that they generalize to other insects, much less to other animals. Moreover, a significant number of parameters need to be set, both for the feature extraction algorithms, and the classification algorithm.

For more complex animal sounds (essentially all non-insect animals), once again features are extracted from the raw data; however, because the temporal transitions between features are themselves a kind of meta-feature, techniques such as Hidden Markov Models are typically used to model these transitions [20][5][1]. This basic idea has been applied with varying degrees of success to birds [14], frogs and mammals [5].

One major limitation of Hidden Markov Model-based systems is that they require careful tuning of their many parameters. This in turn requires a huge amount of labeled training data, which may be difficult to obtain in many circumstances for some species.

Many other approaches have been attempted in the last decade. For example, in a series of papers, Dietrich et al. introduce several classification methods for insect sounds, some of which require up to eighteen parameters, and which were trained on a dataset containing just 108 exemplars [8].

It is important to note that our results are *completely* automatic. Numerous papers report high accuracies for the classification of animal sounds, but upon careful reading it appears (or it is explicitly admitted) that human effort was required to extract the right data to give to the classifier. Many authors do not seem to fully appreciate that "extracting the right data" is *at least* as difficult as the classification step.

For example, a recent paper on the acoustic classification of Australian anurans (frogs and toads) claims a technique that is "*capable to identify the*

[2] In British English, *katydids* are known as *bush-crickets*.

species of the frogs with an average accuracy of 98%." [10]. This technique requires extracting features from syllables, and the authors note, "*Once the syllables have been **properly segmented**, a set of features can be calculated to represent each syllable*" (our emphasis). However, the authors later make it clear that the segmentation is done by careful *human* intervention.

In contrast, we do not make this unrealistic assumption that all the data has been perfectly segmented. We do require sound files that are labeled with the species name, but nothing else. For example, most of the sound files we consider contain human voiceover annotations such as "*June 23th, South Carolina, Stagmomantis carolina, temperature is ...*" and many contain spurious additional sounds such as distant bird calls, aircraft, the researcher tinkering with equipment, etc. The raw unedited sound file is the input to our algorithm; there is no need for costly and subjective human editing.

2.3 Sound Classification in Visual Space

A handful of other researchers have suggested using the visual space to classify sounds (see [19][18]). However, this work has mostly looked at the relatively simple task of recognizing musical instruments or musical genres [29], etc. More recent work has considered addressing problems in bioacoustics in the visual space. In [19] the authors consider the problem of recognizing whale songs using spectrograms. The classification of an observed acoustic signal is determined by the maximum cross-correlation coefficient between its spectrogram and the specified template spectrogram [19]. However, this method is rather complicated and indirect: a "correlation kernel" is extracted from the spectrogram, the image is divided into sections which are piecewise constant, and a cross-correlation is computed from some subsets of these sections and thresholded to obtain a detection event. Moreover, at least ten parameters must be set, and it is not clear how best to set them, other than using a brute force search through the parameter space. This would require a huge amount of labeled training data. In [18] the authors propose similar ideas for bird calls. However, beyond the surfeit of parameters to be tuned, these methods have a weakness that we feel severely limits their applicability. Both these efforts (and most others we are aware of) use *correlation* as the fundamental tool to gauge similarity. By careful normalization, correlation can be made invariant to shifts of pitch and amplitude. However, because of its intrinsically linear nature, correlation cannot be made invariant to global or local differences in time (in a very slightly different context, these are called *uniform scaling* and *time warping*, respectively [9]). There is significant evidence that virtually all real biological signals have such distortions, and that unless it is explicitly addressed in the representation or classification algorithm, we are doomed to poor accuracy. As we shall show empirically in the experimental section below, our proposed method is largely invariant to *uniform scaling* and *time warping*.

2.4 A Review of the Campana-Keogh (CK) distance Measure

The CK distance measure is a recently introduced measure of texture similarity [6]. Virtually all other approaches in the vast literature of texture similarity measures work by explicitly extracting features from the images, and computing the distance between suitably represented feature vectors. Many possibilities for features have been proposed, including several variants of wavelets, Fourier transforms, Gabor filters, etc. [3]. However, one drawback of such methods is that they all require the setting of many parameters. For example, at a minimum, Gabor filters require the setting of scale, orientation, and filter mask size parameters. This has led many researchers to bemoan the fact that "*the values of (Gabor filters parameters) may significantly affect the outcome of the classification procedures...*"[3].

In contrast, the CK distance measure does not require any parameters, and does not require the user to create features of any kind. Instead, the CK measure works in the spirit of Li and Vitanyi's idea that two objects can be considered similar if information garnered from one can help compress the other [15][13]. The theoretical implications of this idea have been heavily explored over the last eight years, and numerous applications for discrete data (DNA, natural languages) have emerged.

The CK measure expands the purview of the compression-based similarity measurements to *real-valued images* by exploiting the compression technique used by MPEG video encoding [6]. In essence, MPEG attempts to compress a short video clip by taking the first frame as a template, and encoding only the *differences* of subsequent frames. Thus, if we create a trivial "video" consisting of just the two images we wish to compare, we would expect the video file size to be small if the two images are similar, and large if they are not. Assuming *x* and *y* are two equally-sized images; Table 1 shows the code to achieve this.

Table 1: The CK Distance Measure

```
function dist = CK(x, y)
dist = ((mpegSize(x,y) + mpegSize(y,x)) /( mpegSize(x,x) + mpegSize(y,y))) - 1;
```

It is worth explicitly stating that this is not pseudo code, but the *entire* actual Matlab code needed to calculate the CK measure.

The CK measure has been shown to be very effective on images as diverse as moths, nematodes, wood grains, tire tracks, etc. [6]. However, this is the first work to consider its utility on spectrograms.

2.5 Notation

In this section we define the necessary notation for our sound fingerprint finding algorithm. We begin by defining the data type of interest, a *sound sequence*:

Definition 1: A *sound sequence* is a continuous sequence $\mathbf{S} = (S_1, S_2, \ldots, S_t)$ of t real-valued data points, where S_t is the most recent value. The data points are typically generated in temporal order and spaced at uniform time intervals.

As with other researchers [19][18], we are interested in the *sound sequence* representation in the *visual space*, which is called the *spectrogram*.

Definition 2: A *spectrogram* is a visual spectral representation of an acoustic signal that shows the relationship between spectral density and the corresponding time.

A more detailed discussion of spectrograms is beyond the scope of this paper, so we refer the reader to [1] and the references therein.

We are typically interested in the *local* properties of the sound sequence rather than the *global* properties, because the entire sound sequence may be contaminated with extraneous sounds (human voice annotations, passing aircraft, etc.). Moreover, as we shall see, our ultimate aim is to find the *smallest* possible sound snippet to represent a species. A local subsection of a spectrogram can be extracted with a *sliding window*:

Definition 3: A *sliding window* (W) contains the latest w data points $(S_{t-w+1}, S_{t-w+2}, \ldots, S_t)$ in the sound sequence $\mathbf{S}$.

Within a sliding window, a local subsection of the sound sequence we are interested in is termed as a *subsequence*.

Definition 4: A *subsequence* ($\mathbf{s}$) of length m of a *sound sequence* $\mathbf{s} = (s_1, s_2, \ldots, s_t)$ is a time series $s_{i,m} = (s_i, s_{i+1}, \ldots, s_{i+m-1})$, where $1 \leq i \leq t-m+1$.

Since our algorithm attempts to find the prototype of a sound sequence $\mathbf{S}$, ultimately, a local sound subsequence $\mathbf{s}$ should be located with a distance comparison between $\mathbf{S}$ and $\mathbf{s}$, which may be of vastly different lengths. Recall that the CK distance is only defined for two images of the same size.

Definition 5: The *distance* ***d*** between a *subsequence* $\mathbf{s}$ and a longer *sound sequence* $\mathbf{S}$ is the minimum distance between $\mathbf{s}$ and all possible subsequences in $\mathbf{S}$ that are the same length as $\mathbf{s}$.

Our algorithm needs some evaluation mechanism for splitting datasets into two groups (*target class*, denoted as ***P***, *everything else*, denoted as ***U***). We use the classic machine learning idea of *information gain* to evaluate candidate splitting rules. To allow discussion of *information gain*, we must first review *entropy*:

Definition 6: The *entropy* for a given *sound sequence* dataset $\mathbf{D}$ is $E(\mathbf{D}) = -p(X)\log(p(X)) - p(Y)\log(p(Y))$, where X and Y are two classes in $\mathbf{D}$, $p(X)$ is the proportion of objects in class X and $p(Y)$ is the proportion of objects in class Y.

The *information gain* is for a given splitting strategy and is just the difference in entropy before and after splitting. More formally:

Definition 7: The *information gain* of a partitioning of dataset $\mathbf{D}$ is:

$$Gain = E(\mathbf{D}) - E'(\mathbf{D}),$$

where $E(\mathbf{D})$ and $E'(\mathbf{D})$ are the entropy before and after partitioning $\mathbf{D}$ into $\mathbf{D_1}$ and $\mathbf{D_2}$, respectively.

$$E'(\mathbf{D}) = f(\mathbf{D_1})E(\mathbf{D_1}) + f(\mathbf{D_2})E(\mathbf{D_2}),$$

where $f(\mathbf{D_1})$ is the fraction of objects in $\mathbf{D_1}$, and $f(\mathbf{D_2})$ is the fraction of objects in $\mathbf{D_2}$.

As noted above, we wish to find a sound fingerprint such that most or all of the objects in ***P*** of the dataset have a subsequence that is similar to the *fingerprint*, whereas most of the sound sequences in ***U*** do not. To find such a *fingerprint* from all possible candidates, we compute the distance between each candidate and every subsequence of the same size in the dataset, and use this information to sort the objects on a number line, as shown in Figure 2. Given such a linear ordering, we can define the best splitting point for a given sound fingerprint:

Definition 8: Given an annotated (by one of two classes, ***P*** and ***U***) linear ordering of the objects in $\mathbf{D}$, there exists *at most*[3] $|\mathbf{D}|$-1 distinct splitting points which divide the number line into two distinct sets. The splitting point which produces the largest information gain is denoted as the *best splitting point*.

In Figure 2 we illustrate the *best splitting point* with a bold/yellow vertical line.

We are finally in a position to define the sound fingerprint using the above definitions:

Definition 9: The *sound fingerprint* for a species is the *subsequence* from ***P***, together with its corresponding *best splitting point*, which produces the largest information gain when measured against the universe set ***U***.

Note that we may expect ties, which must be broken by some policy. We defer a discussion of tie-breaking policies to later in this section.

[3] Note that there can be duplicate values in the ordering.

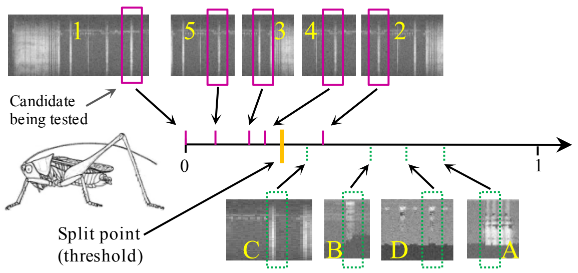

Figure 2: A candidate sound fingerprint, the boxed region in spectrogram 1, is evaluated by finding its nearest neighbor subsequence within both *P* and the four representatives of *U* and then sorting all objects on a number line

We can concretely illustrate the definition of a sound fingerprint using the example shown in Figure 2. Note that there are a total of nine objects, five from ***P***, and four from ***U***. This gives us the entropy for the unsorted data of:

$$[-(5/9)\log(5/9)-(4/9)\log(4/9)] = 0.991$$

If we used the split point shown by the yellow/bold vertical bar in Figure 2, then four objects from ***P*** are the only four objects on the left side of the split point. Of the five objects to the right of the split point we have four objects from ***U*** and just one from ***P***. This gives us an entropy of:

$$(4/9)[-(4/4)\log(4/4)]+(5/9)[-(4/5)\log(4/5)-(1/5)\log(1/5)] = 0.401$$

Thus, we have an information gain of 0.590 = 0.991-0.401. Note that our algorithm will calculate the information gain many times as it searches through the candidate space, and ties are very likely. Thus, we must define a tie-breaking policy. Here we have several options. The intuition is that we want to produce the maximum separation ("margin") between the two classes. We could measure this margin by the absolute distance between the *rightmost* positive and the *leftmost* universe distances. However, this measure would be very brittle to a single mislabeled example. To be more robust to this possibility (which frequently occurs in our data) we define the margin as the absolute distance between the *medians* of two classes. Figure 3 illustrates this idea.

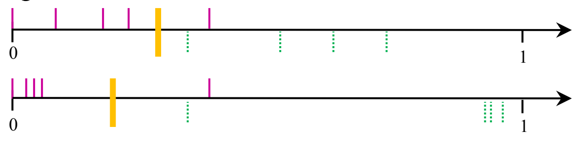

Figure 3: Two order lines that have the same information gain. Our tie-breaking policy reflects the intuition that the *top* line achieves less separation than the bottom line

Even though these two fingerprints have the same information gain of 0.590 as the example shown in Figure 2, the bottom one is preferable, because it achieves a larger margin between ***P*** and ***U***.

Before moving on, we preempt a possible question from the reader. Why optimize the information gain, rather than just optimizing the tie-breaking function itself? The answer is twofold. Just optimizing the tie-breaking function allows pathological solutions that do not generalize well. More critically, as we shall see later, optimizing the information gain will allow admissible pruning techniques that can make our algorithm two orders of magnitude faster.

3. SOUND FINGERPRINTS

As the dendrogram we will later show in Figure 5 hints at, the CK measure can be very accurate in matching (carefully extracted) examples of animal sounds. However, our task at hand is much more difficult than this. We do *not* have carefully extracted prototypes for each class, and we do not have a classification problem where every sound must correspond to some animal we have previously observed.

Rather, for each species we have a collection of sound files which contain within them one or more occurrences of a sound produced by the target. We do not know how long the animal call is, or how many occurrences of it appear in each file. Moreover, since most of the recordings are made in the wild, we must live with the possibility that some of the sound files are "contaminated" with other sounds. For example, a twenty-second recording of a frog we examined also contains a few seconds of Strigiform (owl) calls and several cricket chirps.

In addition, as we later use our sound fingerprints to monitor audio streams we must generally expect that the vast majority of sounds are not created by any of the target species, and thus we have a large amount of data that *could* produce false positives.

3.1 The Intuition of Sound Fingerprints

We begin by xpanding on the intuition behind sound fingerprints. For ease of exposition we will give examples using discrete text strings, but the reader will appreciate that we are really interested in streams of real-valued sounds. Assume we are giving a set of three observations that correspond to a particular species, let us say *Maua affinis* (a cicada from South West Asia):

Ma = {`rrbbcxcfbb`, `rrbbfcxc`, `rrbbrrbbcxcbcxcf`}

We are also given access to the universe of sounds that are known *not* to contain examples of a *Maua affinis*.

¬*Ma* = {`rfcbc`, `crrbbrcb`, `rcbbxc`, `rbcxrf`,..,`rcc`}

In practice, the universe set may be so large that we will just examine a small fraction of it, perhaps just sounds that are likely to be encountered and could be confused for the target insect. Our task is to monitor an audio stream (or examine a large offline archive) and flag any occurrences of the insect of interest.

Clearly it would be quite naive to examine the data for *exact* occurrences of the three positive examples we have been shown, even under a suitably flexible measure such as edit distance. Our positively labeled data is only guaranteed to have one or more samples of the insect call, and it may have additional sections of sounds from other animals or anthropogenic sounds before and/or after it.

Instead, we can examine the strings for shorter substrings that seem diagnostic of the insect. The first candidate template that appears promising is T_1 = **rrbb**, which appears in every *Ma* insect example. However, this substring also appears in the second example in $\neg$*Ma*, in c**rrbb**rcb, and thus this pattern is not unique to *Maua affinis.*

We could try to *specialize* the substring by making it longer; if we use T_2 = **rrbbc**, this does not appear in $\neg$*Ma*, removing that false positive. However, **rrbbc** only appears in two out of three examples in *Ma*, so using it would incur a risk of false negatives. As it happens, the substring template T_3 = **cxc** *does* appear in all examples in *MA* at least once, and never in $\neg$*Ma*, and is thus the best candidate for a prototypical template for the class.

As the reader may appreciate, the problem at hand is *significantly* more difficult that this toy example. First, because we are dealing with real-value data we cannot do simple tests for equality; rather, we must also learn an accept/reject threshold for the template. Moreover, we generally cannot be sure that every example in the positive class really has one true high-quality example call from the target animal. Some examples could be mislabeled, of very low quality, or simply atypical of the species for some reason. Furthermore, we cannot be completely sure that ***U*** does not contain any example from ***P***. Finally, because strings are discrete, we only have to test all possible substrings of length one, then of length two, etc, up to the length of the shortest string in the target class. However, in the real-valued domain in which we must work, the search space is *immensely* larger. We may have recordings that are minutes in length, sampled at 44,100Hz.

Thus far we have considered this problem abstractly: is it really the case that small amounts of spurious sounds can dwarf the similarity of related sounds? To see this we took six pairs of recording of various *Orthoptera* and visually determined and extracted one-second similar regions. The group average hierarchical clustering of the twelve snippets is shown in Figure 4.

The results are very disappointing, given that only one pair of sounds is correctly grouped together, in spite of the fact that human observers can do much better.

We believe this result is exactly analogous to the situation elucidated above with strings. Just as **rrbbcxcfbb** must be stripped of its spurious prefix and suffix to reveal **cxc**, the pattern that is actually indicative of the class, so too must we crop the irrelevant left and right edges of the spectrograms.

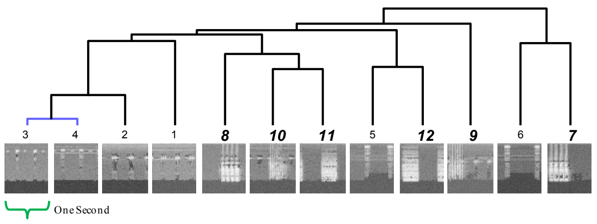

Figure 4: A clustering of six pairs of one-second recordings of various katydids and crickets using the CK texture measure. Only one species pair {3,4} is correctly grouped. Ideally the pairs {1,2}, {5,6}, {7,8}, {9,10} and {11,12} should also be grouped together

For the moment, let us do this by hand. As the resulting images may be of different lengths, we have to slightly redefine the distance measure. To compute the distance between two images of different lengths, we slide the shorter one along the longer one (i.e. definition 5), and report the minimal distance. Figure 5 shows the resulting clustering.

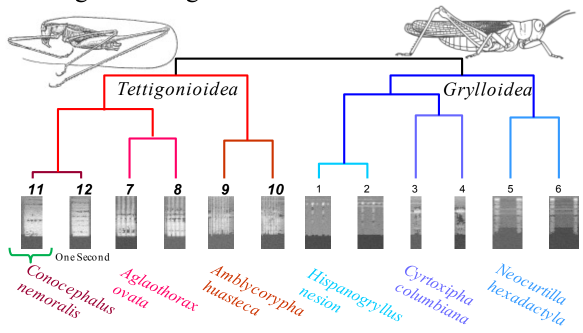

Figure 5: A clustering of the same data used in Figure 4, after trimming irrelevant prefix and suffix data. All pairs are correctly grouped, and at a higher level the dendrogram separates katydids and crickets

The trimming of spurious data produces a dramatic improvement. However, it required careful human inspection. In the next section we will show our novel algorithm which can do this automatically.

3.2 Formal Problem Statement and Assumptions

Informally, we wish to find a snippet of sound that is most representative of a species, on the assumption that we can use this snippet as a template to recognize future occurrences of that species. Since we cannot know the exact nature of the future data we must monitor, we will create a dataset which contains representatives of ***U***, non-target species sounds.

Given this heterogeneous dataset ***U***, and dataset ***P*** which contains only examples from the "positive" species class, our task reduces to finding a subsequence of one of the objects in ***P*** which is close to at least one subsequence in *each* element of ***P***, but far from all subsequences in *every* element of ***U***. Recall that Figure 2 shows a visual intuition of this.

This definition requires searching over a large space of possibilities. How large of a space? Suppose the dataset ***P*** contains a total of *k* sound sequences. Users have the option to define the minimum and maximum (L_{min}, L_{max}) length of sound fingerprint candidates. If they decline to do so we default to L_{max}= infinity and to L_{min}= 16, given that 16 by 16 is the smallest size video MPEG-1 is defined for [6]. Assume for the moment that the following relationship is true:

$$L_{max} \leq min(M_i)$$

That is to say, the longest sound fingerprint is no longer than the shortest object in ***P***, where M_i is the length of S_i from ***P***, $1 \leq i \leq k$.

The total number of sound fingerprint candidates of all possible lengths is then:

$$\sum_{l=L_{\min}}^{L_{\max}} \sum_{S_i \in \{P\}} (M_i - l + 1),$$

where *l* is a fixed length of a candidate. It may appear that we must test every integer pixel size from L_{min} to L_{max}; however, we know that the "block size" of MPEG-1 [6] in a CK measure is eight-by-eight pixels, and pixels remaining after tiling the image with eight-by-eight blocks are essentially ignored. Thus, there is no point in testing non-multiples of eight image sizes. As a result, the above expression can be modified to the one below:

$$\sum_{l=L_{\min}+8(i-1)}^{L_{\max}} \sum_{S_i \in \{P\}} (M_i - l + 1), i = 1, 2, ..., \lceil (L_{\max} - L_{\min})/8 \rceil$$

While this observation means we can reduce the search space by a factor of eight, there is still a *huge* search space that will require careful optimization to allow exploration in reasonable time.

For concreteness, let us consider the following small dataset, which we will also use as a running example to explain our search algorithms in the following sections. We created a small dataset with ***P*** containing ten two-second sound files from *Atlanticus dorsalis* (Gray shieldback), and ***U*** containing ten two-second sound files from other random insects. If we *just* consider fingerprints of length 16 (i.e. L_{min}= L_{max} = 16), then even in this tiny dataset there are 830 candidate fingerprints to be tested, requiring 1,377,800 calls to the CK distance function.

3.3 A Brute-Force Algorithm

For ease of exposition, we begin by describing the brute force algorithm for finding the sound fingerprint for a given species and later consider some techniques to speed this algorithm up.

The brute force algorithm is described in Table 2. We are given a dataset **D**, in which each sound sequence is labeled either class ***P*** or class ***U***, and a user defined length L_{min} to L_{max} (optional: we default to the range sixteen to infinity).

The algorithm begins by initializing *bsf_Gain*, a variable to track the best candidate encountered thus far, to zero in line 1. Then all possible sound fingerprint candidates $S_{k,l}$ for all legal subsequence lengths are generated in the nested loops in lines 2, 4, and 5 of the algorithm.

As each candidate is generated, the algorithm checks how well each candidate $S_{k,l}$ can be used to separate objects into class ***P*** and class ***U*** (lines 2 to 9), as illustrated in Figure 2. To achieve this, in line 6 the algorithm calls the subroutine *CheckCandidates*() to compute the information gain for each possible candidate. If the information gain is larger than the current value of *bsf_Gain*, the algorithm updates the *bsf_Gain* and the corresponding sound fingerprint in lines 7 to 9. The candidate checking subroutine is outlined in the algorithm shown in Table 3.

Table 2: Brute-Force Sound Fingerprint Discovery

```
SoundFP_Discovery(D, L_min, L_max)
Require: A dataset D (P and U) of sound sequence's spectrogram, user defined minimum length and maximum length of sound fingerprint
Ensure: Return the sound fingerprint

1   bsf_Gain ← 0
2   For i ← 1 to |P| do {every spectrogram in P}
3       S ← P_i
4       For l ← L_min to L_max do {every possible length}
5           For k ← 1 to |S| - l + 1 do {every start position}
6               gain ← CheckCandidates(D, S_k,l)
7               If gain > bsf_Gain
8                   bsf_Gain ← gain
9                   bsfFingerprint ← S_k,l
10              EndIf
11          EndFor
12      EndFor
13  EndFor
14  Return bsfFingerprint
```

In the subroutine *CheckCandidates*(), shown in Table 3, we compute the *order line* ***L*** according to the distance from the *sound sequence* to the candidate computed in *minCKdist*() procedure, which is shown in Table 4. In essence, this is the procedure illustrated in Figure 2. Given ***L***, we can find the optimal split point (definition 8) in lines 10 to 15 by calculating all possible splitting points and recording the best.

While the splitting point can be any point on the positive real number line, we note that the information gain cannot change in the region between any two adjacent points. Thus, we can exploit this fact to produce a finite set of possible split positions. In particular, we need only test |**D**|-1 locations.

In the subroutine *CheckCandidates*() this is achieved by only checking the mean value (the "halfway point") of each pair of adjacent points in the distance ordering as the possible positions for the split point. In *CheckCandidates*(), we call the subroutine *minCKdist*() to find the best matching subsequence for a given candidate under consideration.

Table 3: Check the Utility of Single Candidate

```
CheckCandidates (D or Dist, candidate $S_{k,l}$)
Require: A dataset D of spectrogram (or distance ordering), sound fingerprint candidate $S_{k,l}$
Ensure: Information Gain gain
1   $L \leftarrow \emptyset$
2   If first input is D
3    For j ← 1 to |D| do {compute distance of every spectrogram to the candidate sound fingerprint $S_{k,l}$}
4       $dist \leftarrow minCKdist(D_j, S_{k,l})$
5       insert $D_j$ into L by the key dist
6    EndFor
7   Else
8      $dist \leftarrow Dist$
9   EndIf
10    $I(D) \leftarrow$ new information gain after split computed by def 7
11    For split ← 1 to |D|-1do
12       Count $N_1$, $N_2$ for both the partitions
13       $I'(D).split \leftarrow$ new information gain after split computed by def 7
14       $gain(D) = max(I(D) - I'(D).split)$
15    EndFor
16  Return gain(D)
```

We do this for every spectrogram in **D**, including the one from which the candidate was culled. This explains why in each order line at least one subsequence is at zero (c.f. Figure 2 and Figure 7).

In *minCKdist*() (Table 4), we use the CK measure [6] as the distance measurement between a candidate fingerprint and a generally much longer spectrogram.

Table 4: Compute Minimum Subsequence CK Distance

```
minCKdist ($D_j$, candidate $S_{k,l}$)
Require: A sound sequence's spectrogram $D_j$, sound fingerprint candidate $S_{k,l}$
Ensure: Return the minimum distance computed by CK
1   $minDist \leftarrow Infinity$
2    For $i \leftarrow 1$ to $|D_{j,i}| - |S_{k,l}| + 1$ do {every start position}
3       $CKdist \leftarrow CK(D_{j,i}, S_{k,l})$
4       If $CKdist < minDist$
5           $minDist \leftarrow CKdist$
6       EndIf
7    EndFor
8   Return minDist
```

In Figure 6 we show a trace of the brute force algorithm on the *Atlanticus dorsalis* problem.

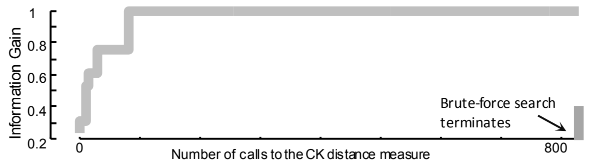

Figure 6: A trace of value of the *bsf_Gain* variable during brute force search on the *Atlanticus dorsalis* dataset. Only sound fingerprints of length 16 are considered here for simplicity

Note that the search continues even after an information gain of one is achieved in order to break ties. The 1,377,800 calls to the CK function dominate the overall cost of the search algorithm (99% of the CPU time is spent on this) and require approximately 8 hours. This is not an unreasonable amount of time, considering the several days of effort needed for an entomologist to collect the data in the field. However, this is a tiny dataset. We wish to examine datasets that are orders of magnitude larger. Thus, in the next section we consider speedup techniques.

3.4 Admissible Entropy Pruning

The most expensive computation in the brute force search algorithm is obtaining the distances between the candidates and their nearest matching subsequences in each of the objects in the dataset. The information gain computations (including the tie breaking computations) are inconsequential in comparison. Therefore, our intuition in speeding up the brute force algorithm is to eliminate as many distance computations as possible.

Recall that in our algorithm, we have to obtain the annotated linear ordering of all the candidates in ***P***. As we are incrementally doing this, we may notice that a particular candidate looks very unpromising. Perhaps when we are measuring the distance from the current candidate to the first object in ***U*** we find that it is a *small* number (recall that we want the distances to ***P*** to be small and to ***U*** large), and when we measure the distance to the next object in ***U*** we again find it to be small. Must we continue to keep testing this unpromising candidate? Fortunately, the answer may be "no". Under some circumstances we can admissibly prune unpromising fingerprints; without having to check all the objects in the universe ***U***.

The key observation is that we can cheaply compute the *upper bound* of the current partially computed linear ordering at any time. If the *upper bound* we obtain is less than the *best-so-far* information gain (i.e. the ***bsf_Gain*** of Table 2), we can simply eliminate the remaining distance computations in ***U*** and prune this particular fingerprint candidate from consideration.

To illustrate this pruning policy, we consider a concrete example. Suppose that during a search the *best-so-far* information gain is currently 0.590, and we are incrementally beginning to compute the sound fingerprint shown in Figure 2. Assume that the partially computed linear ordering is shown in Figure 7. We have computed the distances to all five objects in ***P***, and to the first two objects in ***U***.

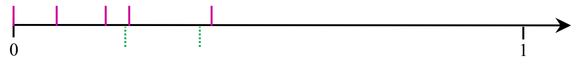

Figure 7: The order line of all the objects in *P* and just the first two objects in *U*

Is it possible that this candidate will yield a score better than our *best-so-far*? It is easy to see that the most optimistic case (i.e., the upper bound) occurs if all of the remaining objects in ***U*** map to the far right, as we illustrate in Figure 8.

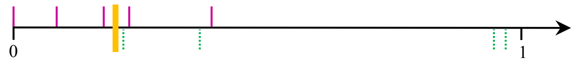

Figure 8: The logically best possible order line based on the distances that have been calculated in Figure 7. The best split point is shown by the yellow/heavy line

Note that of the three objects on the left side of the split point, all three are from ***P***. Of the six objects on the right side, two are from ***P*** and four are from ***U***. Given this, the entropy of the hypothetical order line shown in Figure 8 is:

$$(3/9)[-(3/3)\log(3/3)]+(6/9)[-(4/6)\log(4/6)-(2/6)\log(2/6)] = 0.612$$

Therefore, the best possible information gain we could obtain from the example shown in Figure 7 is just 0.612, which is lower than the *best-so-far* information gain. In this case, we do not have to consider the ordering of the remaining objects in ***U***. In this toy example we have only pruned two invocations of the *CheckCandidates*() subroutine shown in Table 3. However, as we shall see, this simple idea can prune more than 95% of the calculations for more realistic problems.

The formal algorithm of admissible entropy pruning is shown in Table 5. After the very first sound fingerprint candidate check, for all the remaining candidates, we can simply insert *EntropyUBPrune*() in line 4 of Table 3, and eliminate the remaining CK distance and information gain computation if the current candidate satisfies the pruning condition, as we discussed in this section. *EntropyUBPrune*() takes the *best-so-far* information gain, current distance ordering from class ***P*** and class ***U***, and remaining objects in ***U***, and returns the fraction of the distance measurements computed to see how much elimination we achieved.

Table 5: Entropy Upper Bound Pruning

EntropyUBPrune (*U_m, currentDist, $S_{k,l}$, bsf_Gain*)	
Require: A sound sequence's spectrogram U_m, current distance ordering, *sound fingerprint* candidate $S_{k,l}$, best-so-far information gain	
Ensure: Return fraction of distance computations in U	
1	fraction ← 0
2	counter ← 0
3	rightmostDist ← largest distance value in currentDist + 1
4	bestDist ← Add rightmostDist for U_m to currentDist
5	*gain* ← *CheckCandidates*(*bestDist*, $S_{k,l}$)
6	**If** *gain* >*bsf_Gain*
7	Return **False** and increment counter
8	**Else**
9	Return **True**
10	**EndIf**
11	**Return** fraction ← counter/|*U*|, *gain*

We can get a hint as to the utility of this optimization by revisiting the *Atlanticus dorsalis* problem we considered above. Figure 9 shows the difference entropy pruning makes in this problem.

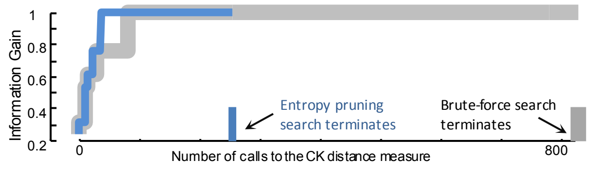

Figure 9: A trace of the *bsf_Gain* variable during brute force and entropy pruning search on the *Atlanticus dorsalis* dataset

Note that not only does the algorithm terminate earlier (with the exact same answer), but it *converges* faster, a useful property if we wish to consider the algorithm in an anytime framework [27].

3.5 Euclidean Distance Ordering Heuristic

In both the brute force algorithm and the entropy-based pruning extension introduced in the last section, we generate and test candidates; from left to right; and top to bottom based on the given lexical order of the objects' label (i.e., the file names used by the entomologist).

There are clearly other possible orders we could use to search, and it is equally clear that for entropy-based pruning, some orders are better than others. In particular, if we find a candidate which has a relatively high information gain early in the search, our pruning strategy can prune much more effectively.

However, this idea appears to open a "*chicken and egg*" paradox. How can we know the best order; until we have finished the search? Clearly, we cannot. However, we do not need to find the *optimal* ordering; we just need to encounter a relatively good candidate relatively early in the search. Table 6 outlines our idea to achieve this. We simply run the entire brute force search using the Euclidean distance as a proxy for the CK distance, and sort the candidates based on the information gain achieved using the *Euclidean distance*.

Concretely, we can insert *EuclideanOrder*() between lines 4 and 5 in Table 2 to obtain a better ordering to check all the candidates.

Table 6: Euclidean Distance Measure Order Pruning

EuclideanOrder (*D, minLen, maxLen*)	
Require: A dataset *D* (*P* and *U*) of sound sequence's spectrogram, user defined minimum/maximum length of *sound fingerprint*	
Ensure: Return the *new order of candidates*	
1	Replace CK measure with Euclidean distance measure
2	*newGain* ← *CheckCandidates* (*D or Dist, candidate $S_{k,l}$*)
3	*newOrder* ← sort the candidates by decreasing *newGain*
4	**Return** *newOrder*

Running this preprocessing step adds some overhead; however, it is inconsequential because the Euclidean distance is at least two orders of magnitude faster than the CK distance calculation. For this idea to work well, the Euclidean distance must be a good proxy for the CK distance calculation. To see if this is the case, we randomly extracted 1,225 pairs of insect sounds and measured the distance between them under both measures, using the two values to plot points in a 2D scatter plot, as shown in Figure 10. The results suggest that Euclidean distance is a very good surrogate for CK distance.

To measure the effect of this reordering heuristic we revisited our running example shown in Figure 6/ Figure 9.

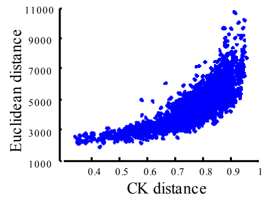

Figure 10: The relationship between Euclidean and CK distance for 1,225 pairs of spectrograms

The Euclidean distance reordering heuristic is shown in Figure 11.

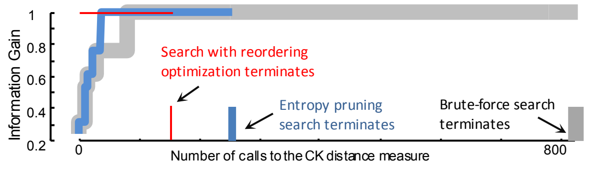

Figure 11: A trace of value of the *bsf_Gain* variable during brute force, entropy pruning, and reordering optimized search on the *Atlanticus dorsalis* dataset

As we can see, our heuristic has two positive effects. First, the absolute time to finish (with the identical answer as a brute force search) has significantly decreased. Secondly, we converge on high quality solution faster. This is a significant advantage if we wanted to cast the search problem as an anytime algorithm [27]. As impressive as the speedup results are, as we shall show in the next section, they are pessimistic due to the small size of our toy problem.

4. EXPERIMENTAL EVALUATION

We have created a supporting webpage [11], which contains all code/data used in this work. Moreover, the webpage contains addition experiments, along with videos and sounds files that allow the interested reader to get a better appreciation of the scale and complexity of the data we are working with.

4.1 CK as a Tool for Taxonomy

We begin by noting that beyond the utility of our ideas for monitoring wildlife, the CK measure may be useful as a taxonomic tool. Consider the insect shown in Figure 12. As noted in a National Geographic article, "*the sand field cricket (Gryllus firmus) and the southeastern field cricket (Gryllus rubens) look nearly identical and inhabit the same geographical areas*" [25]. Thus, even if handling a living specimen, most entomologists could not tell them apart without resorting to DNA analysis.

We suspected that we might be able to tell them apart by sound[4]. While we do not have enough data to do forceful and statistically significant experiments, we can do two tentative tests. As shown in Figure 12, we projected twenty-four examples from the two species into two-dimensional space using multi-dimensional scaling, and we also clustered eight random examples, four from each class.

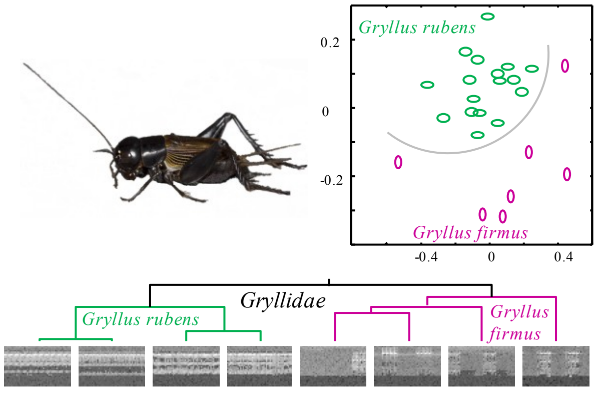

Figure 12: *top left*) An insect found in Florida: is it a *G. rubens* or *G. firmus*? *top right*) Projecting one-second snippets of songs from both insects into 2D space suggests they are almost linearly separable, a possibility reflected by their clusterability (*bottom*)

The results suggest that these congeneric[5] species are almost linearly separable in two-dimensional space (they *are* linearly separable in three-dimensional space).

4.2 Insect Classification

There are currently no benchmark problems for insect classification. Existing datasets are either too small to make robust claims about accuracy, or were created by authors unwilling to share their data. To redress this we created and placed into the public domain a large classification dataset [11]. The data consists of twenty species of insects, eight of which are *Gryllidae* (crickets) and twelve of which are *Tettigoniidae* (katydids)[6]. Thus, we can treat the problem as either a twenty-class *species* level problem, or two-class *genus* level problem. For each class we have ten training and ten testing examples. It is important to note that we assembled these datasets *before* attempting classification, explicitly to avoid cherry-picking. Note that because of convergent evolution, mimicry and the significant amounts of noise in the data (which were collected in the field) we should not expect perfect accuracy here. Moreover, this group of insects requires some *very* subtle distinctions to be made; for example, *Neoconocephalus bivocatus*, *Neoconocephalus retusus*, and *Neoconocephalus maxillosus* are obviously in the same genus, and are *visually* indistinguishable at least to our untrained eye.

[4] In brief, it is well known that the acoustic behavior of insects is important in insect speciation, especially for sympatric speciation, where new species evolve from a single ancestral species while inhabiting the same geographic region [26].

[5] Species belonging to the same genus are *congeneric*.

[6] A full description of the data is at [11].

Likewise, we have multiple representatives from both the *Belocephalus* and *Atlanticus* genera.

We learned twenty sound fingerprints using the algorithm in Section 3. We then predicted the testing exemplars class label by sliding each fingerprint across it and recording the fingerprint that produced the minimum value as the exemplar's nearest neighbor. The classification accuracies are shown in Table 7.

Table 7: Insect Classification Accuracy

	species-level problem		genus-level problem	
	default rate	fingerprint	default rate	fingerprint
10 species	0.10	0.70	0.70	0.93
20 species	0.05	0.44	0.60	0.77

The results are generally impressive. For example, in the ten-class species-level problem the default accuracy rate is only 10%, but we can achieve 70%. It is worth recalling the following when considering these results.

- The testing data does not consist of carefully extracted *single* utterances of an insect's call. Rather, it consists of one or two-minute sound files known to contain at least one call, together with human voice annotations and miscellaneous environmental sounds that can confuse the classification algorithm.
- As noted above, our dataset has multiple congeneric species; that, at least to our eyes and ears, look and sound identical. This is an *intrinsically hard* problem.
- The reader can be assured that the results are not due to overfitting, because we did not *fit* any parameters in this experiment[7]. These are "black box" results.
- We can do a little better by weighting the nearest neighbor information with the threshold information (which we ignore in the above). Since this *does* introduce a (weighting) parameter to be tuned, in the interest of brevity, given page limits and our already excellent results, we defer such discussions to future work.

4.3 Monitoring with Sound Fingerprints

To test our ability to monitor an audio stream in real time for the presence of a particular species of insects, we learned the sound fingerprints for three insect species of insects native to Florida. In each case we learned from training sets consisting of ten insects.

To allow visual appreciation of our method, as shown in Figure 13 we produced an eight-second sequence of audio by concatenating snippets of four different species, including holdout (i.e. *not* seen in the training set) examples from our three species of interest. While each fingerprint has a different threshold, for simplicity and visual clarity we show just the averaged threshold.

[7] The minimum fingerprint length is set to 16, a hard limit due to the way MPEG is coded. The maximum length is set to infinity.

As we can see in Figure 13, this method achieves three true positives, and more remarkably, no false positives. Recall that the CK distance measure exploits the compression technique used by MPEG video encoding, which is among the most highly optimized computer code available. Thus, we can do this monitoring experiment in real time, even on an inexpensive laptop.

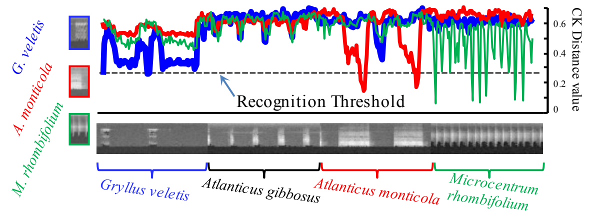

Figure 13: (Image best viewed in color) *far left*) Three insect sound fingerprints are used to monitor an eight-second audio clip. In each case, the fingerprint distance to the sliding window of audio dips below the threshold as the correct species sings, but not when a different species is singing

4.4 Scalability of Fingerprint Discovery

Recall the experiments shown in Section 3, when our toy example had only ten objects in both ***P*** and ***U***. We showed a speedup of about a factor of five, although we claimed this is pessimistic because we expect to be able to prune more aggressively with larger datasets. To test this, we reran these experiments with a more realistically-sized ***U***, containing 200 objects from other insects, birds, trains, helicopters, etc. As shown in Figure 14, the speedup achieved by our reordering optimization algorithm is a factor of 93 in this case.

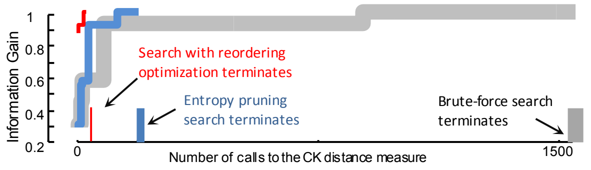

Figure 14: A trace of value of the *bsf_Gain* variable during brute force, entropy pruning, and reordering optimized search on the *Atlanticus dorsalis* dataset with the 200-object universe

4.5 Archived Experiments

Page limitations make it difficult to show all our extensive empirical work. We urge the interested reader to visit [11], where we have more than one-hundred additional experiments. Among the questions we consider are: Do sound fingerprints work for other vocal animals such as frogs/birds? (*yes*); is our method robust to mislabeled training data? (so long as the majority of data in ***P*** is correctly labeled, *yes*); are we robust to noisy environments, such as aircraft noise in the background as a monitored cricket chirps? (so long as the target signal is a significant fraction of the overall signal, *yes*).

Moreover, at [11] we have embedded sound and video files to allow a more direct appreciation of the subtlety of the distinctions our system can make.

5. CONCLUSION AND FUTURE WORK

In this work we have introduced a novel bioacoustic recognition/classification framework. We feel that unlike other work in this area, our ideas have a real chance to be adopted by domain practitioners, because our algorithm is essentially a "black box", requiring only that the expert can label some data. We have shown through extensive experiments that our method is accurate, robust and efficient enough to be used in real time in the field.

In future work we will explore expanding the representational power of sound fingerprints with logical operators, such as classifying a sound as feline if we hear a "hiss" **OR** a "mew". We are also beginning to explore the spatiotemporal data mining problems inherent in monitoring large sites with multiple sensors.

ACKNOWLEDGMENTS

We thank the Cornell Lab of Ornithology for donating much of the data used in [16], Jesin Zakaria for help projecting snippets of sounds into 2D space, and Dr. Agenor Mafra-Neto for entomological advice. This work was funded by NSF awards 0803410 and 0808770.

6. REFERENCES

[1] R. Bardeli, *Similarity search in animal sound databases*, IEEE Transactions on Multimedia, vol. 11, no. 1, pp. 68–76, 2009.

[2] Y. Beiderman, Y. Azani, Y. Cohen, C. Nisankoren, M. Teicher, V. Mico, , J. Garcia, Z. Zalevsky, *Cleaning and quality classification of optically recorded voice signals*, Recent Patents on Signal Proc' 6-11, 2010.

[3] F. Bianconi, A. Fernandez, *Evaluation of the effects of Gabor filter parameters on texture classification*. Pattern Recognition 40(12), 3325–35 (2007).

[4] D. T. Blumstein et.al, *Acoustic monitoring in terrestrial environments using microphone arrays: applications, technological considerations, and prospectus*, J. Appl Ecol 48:758–767, 2001.

[5] J. C. Brown, P. Smaragdis, *Hidden Markov and Gaussian mixture models for automatic call classification*. J. Acoust. Soc. Am, (6):221–22, 2009.

[6] B. J. L. Campana, E. J. Keogh, *A compression-based distance measure for texture*, Statistical Analysis and Data Mining 3(6): 381-398 (2010)

[7] T. Dang, N. Bulusu, W. C. Feng, W. Hu, *RHA: A Robust Hybrid Architecture for Information Processing in Wireless Sensor Networks*, In 6th ISSNIP 2010.

[8] C. Dietrich, F. Schwenker, G. Palm, *Classification of Time Series Utilizing Temporal and Decision Fusion, Proceedings of Multiple Classifier Systems (MCS)*, pp 378-87, 2001.

[9] A. Fu, E. Keogh, L. Lau, C. A. Ratanamahatana, R. C.-W. Wong, *Scaling and time warping in time series querying*. VLDB J. 17(4): 899-921 (2008).

[10] N. C. Han, S. V. Muniandy, J. Dayou, *Acoustic classification of Australian anurans based on hybrid spectral-entropy approach*, Applied Acoustic, 72(9): 639-645, 2011

[11] Y, Hao. Animal sound fingerprint Webpage. *www.cs.ucr.edu/~yhao/animalsoundfingerprint.html*

[12] T. E. Holy, Z. Guo, *Ultrasonic songs of male mice*, PLoS Biol 3:e386, 2005.

[13] E. Keogh, S. Lonardi, C. A. Ratanamahatana, L. Wei, S. Lee, J. Handley, *Compression-based data mining of sequential data*, DMKD. 14(1): 99-129, 2007.

[14] J. A. Kogan, D. Margoliash, *Automated recognition of bird song elements from continuous recordings using dynamic time warping and hidden markov models: a comparative study*, J. Acoust. Soc. Am. 103(4):2185–219, 1998.

[15] M. Li, X. Chen, X. Li, B. Ma, P. Vitanyi, *The similarity metric*, Proc'of the 14th Symposium on Discrete Algorithms, pp: 863 -72, 2003.

[16] Macaulay Library, Cornell Lab of Ornithology, *www.macaulaylibrary.org/index.do*

[17] R. Mankin, D. Hagstrum, M. Smith, A. Roda, M. Kairo, *Perspective and Promise: a Century of Insect Acoustic Detection and Monitoring*, Amer. Entomol. 57: 30-44.

[18] M. Marcarini, G. A. Williamson, L. de S. Garcia, *Comparison of methods for automated recognition of avian nocturnal flight calls*, ICASSP 2008: 2029-32.

[19] D. K. Mellinger, C. W. Clark, *Recognizing transient low-frequency whale sounds by spectrogram correlation*, J. Acoust. Soc. Am., 107: 6, pp. 3518-29, 2000.

[20] D. Mitrovic, M. Zeppelzauer, C. Breiteneder, *Discrimination and Retrieval of Animal Sounds*, In Proc. of IEEE Multimedia Modelling Conference, Beijing, China, 339-343, 2006.

[21] A. Celis-Murillo, J. L. Deppe, M. F. Allen, *Using soundscape recordings to estimate bird species abundance, richness, and composition*, Journal of Field Ornithology, 80, 64–78, 2009.

[22] D. J. Nowak, J. E. Pasek, R. A. Sequeira, D. E. Crane, V. C. Mastro, *Potential effect of Anoplophora glabripennis on urban trees in the United States*, Journal of Entomology. 94(1): 116-122, 2001.

[23] J. B. Panksepp, K. A. Jochman, J. U. Kim, J. J. Koy, E. D. Wilson, Q. Chen, C. R. Wilson, G. P. Lahvis, *Affiliative behavior, ultrasonic communication and social reward are influenced by genetic variation in adolescent mice*, PLoS ONE 4:e351 (2007).

[24] K. Riede, F. Nischk, C. Thiel, F. Schwenker, *Automated annotation of Orthoptera songs: first results from analysing the DORSA sound repository*, Journal of Orthoptera Research 15(1), 105-113, 2006.

[25] J. Roach, *Cricket, Katydid Songs Are Best Clues to Species' Identities. National Geographic News*, (URL) *news.nationalgeographic.com/news/2006/09/060905-crickets.html*

[26] M. M. Wells, C. S. Henry, *Songs, reproductive isolation, and speciation in cryptic species of insect: a case study using green lacewings*, In Endless Forms: species and speciation, Oxford Univ. Press, NY, 1998.

[27] X. Xi, K.Ueno, E. Keogh, D.J Lee, *Converting non-parametric distance-based classification to anytime algorithms*. Pattern Anal. Appl. 11(3-4): 321-36 (2008).

[28] L.Ye and E.Keogh. *Time series shapelets: a new primitive for data mining*. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD, pages 947-956, 2009.

[29] G. Yu, J.-J. Slotine, *Audio classification from time-frequency texture*, IEEE ICASSP, 2009, pp. 1677-80.