J Insect Behav
DOI 10.1007/s10905-012-9361-5

Monitoring and Mining Animal Sounds in Visual Space

Yuan Hao · Bilson Campana · Eamonn Keogh

Revised: 18 October 2012 / Accepted: 19 October 2012

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), are either too computationally expensive for deployment in resource-limited sensors, specialized for a very small group of species, or 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 sonograms 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 · sonogram · texture · bioacoustics · acoustic monitoring

Y. Hao (✉) · B. Campana · E. Keogh
Department of Computer Science & Engineering, University of California, Riverside, Riverside, CA, USA
e-mail: yhao@cs.ucr.edu

B. Campana
e-mail: bcampana@cs.ucr.edu

E. Keogh
e-mail: eamonn@cs.ucr.edu

Published online: 13 November 2012

Introduction

Monitoring animals by the sounds they produce is an important and challenging task, whether the application is outdoors in a natural habitat (Blumstein 2001), 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 (Celis-Murillo et al. 2009; Schmidt et al. 2011).

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 (Panksepp et al. 2007).

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 (Nowak et al. 2001). It has been noted that different beetle species have subtle distinctive chewing sounds, and ultra sensitive sensors that can detect these sounds are being produced (Mankin et al. 2011). 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*" (Nowak et al. 2001).

With such a plethora of important applications, there have been significant efforts to build bioacoustic classification tools for animals (Blumstein 2001). However, we argue that most current tools are severely limited. They often require the careful tuning of many parameters (as many as 18 (Dietrich et al. 2001)) and thus huge amounts of training data, are too computationally expensive for deployment with resource-limited sensors that will be deployed in the field (Dang et al. 2010), are specialized for a notably small group of species, or are simply not accurate enough to be useful.

Our aim is to 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 sonograms 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 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 (Dang et al. 2010), etc. For simplicity, we gloss over such considerations, referring the interested reader to (Blumstein 2001) 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.

Related Work/Background

A Brief Review of Sonograms

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

There is a huge amount of literature leveraging off *manual* inspection of such sonograms; see (Holy and Guo 2005) and the references therein for some examples. However, as we shall see, algorithmic analysis of sonograms remains an open problem and thus 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., sonograms almost always have significant noise artifacts, even when obtained in tightly controlled conditions in a laboratory setting (Schmidt et al. 2011). One avenue of research is to “clean” the sonograms using various techniques (Beiderman et al. 2010), and then apply shape similarity measures to the cleaned shape primitives. Some types of specialized cleaning may be possible; for example, removing the 60 Hz noise is commonly encountered (American domestic electricity is at 60 Hz (most of the rest of the world is 50 Hz) and inadequate filtering in power transformers often allows some 60 Hz signal to bleed into the sound recording). However, algorithms to robustly clean general sonograms are 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 to use the raw sonogram directly.

General Animal Sound Classification

The literature on the classification of animal sounds is vast; we refer the interested reader to (Mitrovic et al. 2006; Bardeli 2009) for useful surveys. At the highest level, most research efforts advocate the extraction of sets of features from the data, and the use of these features as inputs for standard classification algorithms such as a decision tree, a Bayesian classifier or a neural network. As a concrete representative example, consider (Riede et al. 2006), which introduces a system to recognize *Orthoptera* (the

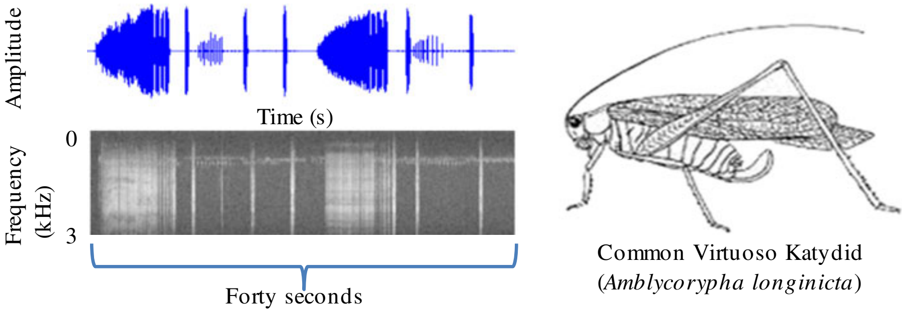

Fig. 1 A sonogram of the call of an insect. Note the highly repetitious nature of the call. In this case, capturing just two “busts” may be sufficient to recognize the insect

order of insects that includes grasshoppers, crickets, katydids (In British English, katydids are known as *bush-crickets*) 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 whehter they generalize to other insects, much less to other animals. Moreover, a significant number of parameters need to be set, for both the feature extraction algorithms and the classification algorithms.

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 a kind of meta-feature themselves, techniques such as Hidden Markov Models are typically used to model these transitions (Mitrovic et al. 2006; Brown and Smaragdis 2009; Bardeli 2009). This basic idea has been applied with varying degrees of success to birds (Kogan and Margoliash 1998), frogs, and mammals (Brown and Smaragdis 2009).

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 18 parameters, and which are trained on a dataset containing just 108 exemplars (Dietrich et al. 2001).

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 is required for extracting the right data to give to the classifier. Many authors do not 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 species of the frogs with an average accuracy of 98 %.*" (Han et al. 2011). 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 noise, the researcher tinkering with equipment, etc. The raw, unedited sound file is the input for our algorithm; there is no need for costly and subjective human editing.

Sound Classification in Visual Space

A handful of other researchers have suggested using the visual space to classify sounds (see (Mellinger and Clark 2000; Marcarini et al. 2008)). However, this work has mostly looked at the relatively simple task of recognizing musical instruments or musical genres (Yu and Slotine 2009), etc. More recent work has considered addressing problems in bioacoustics in the visual space. In (Mellinger and Clark 2000) the authors consider the problem of recognizing whale songs using sonograms. The classification of an observed acoustic signal is determined by the maximum cross-correlation coefficient between its sonogram and the specified template sonogram (Mellinger and Clark 2000). However, this method is rather complicated and indirect: First, a "correlation kernel" is extracted from the sonogram, then, the image is divided into sections which are piecewise constant, and finally, 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 in order to carry out this method, and it is not clear how best to set them, other than by using a brute force search through the parameter space. This would require a huge amount of labeled training data. In (Marcarini et al. 2008), 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 slightly different context, these are called *uniform scaling* and *time warping*, respectively (Fu et al. 2008)). There is significant evidence that virtually all real biological signals have such distortions (Desutter-Grandcolas 1998), and 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*.

A Review of the CK Measure

The CK distance measure is a recently introduced measure of texture similarity (Campana and Keogh 2010). 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. (Bianconi and Fernandez 2007). 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…*" (Bianconi and Fernandez 2007).

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 (Li et al. 2003; Keogh et al. 2007). The theoretical implications of this idea have been heavily explored over the last 8 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 (Campana and Keogh 2010). In essence, MPEG attempts to compress a short video clip by taking the first frame as a template, and encoding only the *differences* amongst 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.

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 effective on images as diverse as moths, nematodes, wood grains, tire tracks, etc. (Campana and Keogh 2010). However, this is the first work to consider its utility on sonograms.

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 (Mellinger and Clark 2000; Marcarini et al. 2008), we are interested in the *sound sequence* representation in the *visual space*, which is called the *sonogram*.

Definition 2: A *sonogram* 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 sonograms is beyond the scope of this paper, so we refer the reader to (Bardeli 2009) 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 (e.g. 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 sonogram can be extracted with a *sliding window*:

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;
```

Definition 3: A *sliding window* (*W*) contains the latest *w* data points ($S_{t\text{-}w+1}$, $S_{t\text{-}w+2}$, …, S_t) in the sound sequence **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* (**s**) of length m of a *sound sequence* **s** = (s_1, s_2, …, s_t) is a time series $s_{i,m}$ = (s_i, s_{i+1}, …, s_{i+m-1}), where $1 \leq i \leq t\text{-}m+1$.

Since our algorithm attempts to find the prototype of a sound sequence **S**, ultimately, a local sound subsequence **s** should be located with a distance comparison between **S** and **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* **s** and a longer *sound sequence* **S** is the minimum distance between **s** and all possible subsequences in **S** that are also length **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 **D** is $E(\mathbf{D}) = -p(X)\log(p(X))-p(Y)\log(p(Y))$, where *X* and *Y* are two classes in **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 **D** is:

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

where *E*(**D**) and *E'*(**D**) are the entropy before and after partitioning **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 Fig. 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 **D**, there exists *at most* (Note that there can be duplicate values in the ordering) |**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 Fig. 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 ***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.

We can concretely illustrate the definition of a sound fingerprint using the example shown in Fig. 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 Fig. 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 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.

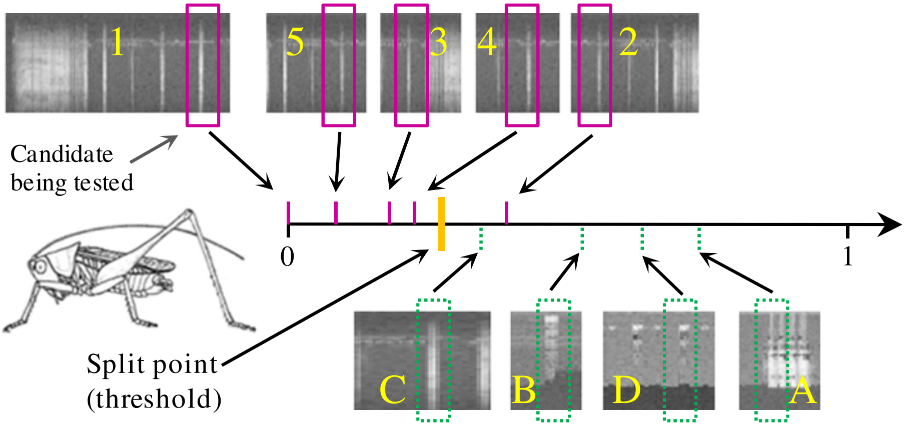

Fig. 2 A candidate sound fingerprint, the boxed region in sonogram 1, is evaluated by finding its nearest neighbor subsequence within both *P* and the four representatives of *U* and then sorting all objects on a number line

However, this measure would be 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.

Even though these two fingerprints have the same information gain of 0.590 as the example shown in Fig. 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 results in 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.

Sound Fingerprints

As the dendrogram we later show in Fig. 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.

The Intuition of Sound Fingerprints

We begin by expanding on the intuition behind sound fingerprints. For ease of exposition we will give examples using discrete text strings, but the reader will

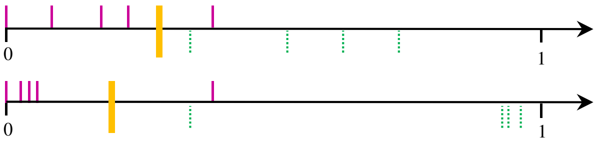

Fig. 3 Two order lines that have the same information gain. Our tie-breaking policy reflects the intuition that the *top* line achieves less separation than the *bottom* line

appreciate that we are really interested in streams of real-valued sounds. Assume we are given a set of three observations that correspond to a particular species, let us say *Maua affinis* (a cicada from South West Asia):

$$Ma = \{\texttt{rrbbcxcfbb},\ \texttt{rrbbfcxc},\ \texttt{rrbbrrbbcxcbcxcf}\}$$

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

$$\neg Ma = \{\mathrm{rfcbc},\ \mathrm{crrbbrcb},\ \mathrm{rcbbxc},\ \mathrm{rbcxrf}, .., \mathrm{rcc}\}$$

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 ¬*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 ¬*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 ¬*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 than 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 44,100 Hz.

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 if this is true, 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 12 snippets is shown in Fig. 4.

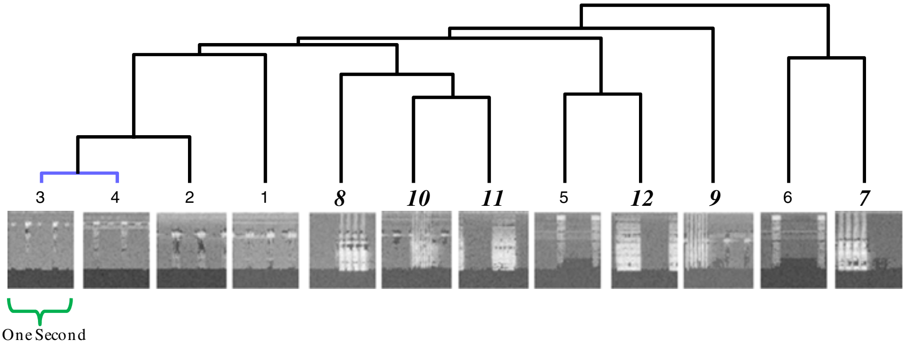

Fig. 4 A clustering of six pairs of one-second recordings of various katydids and crickets using the CK texture measure. Only one species pair {3,4} is correctly grouped. Ideally the pairs {1,2}, {5,6}, {7,8}, {9,10} and {11,12} should also be grouped together

The results are 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 sonograms.

For the moment, let us do this by hand. As the resulting images may be of different lengths, we have to redefine the distance measure slightly. 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.

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 is capable of doing this automatically.

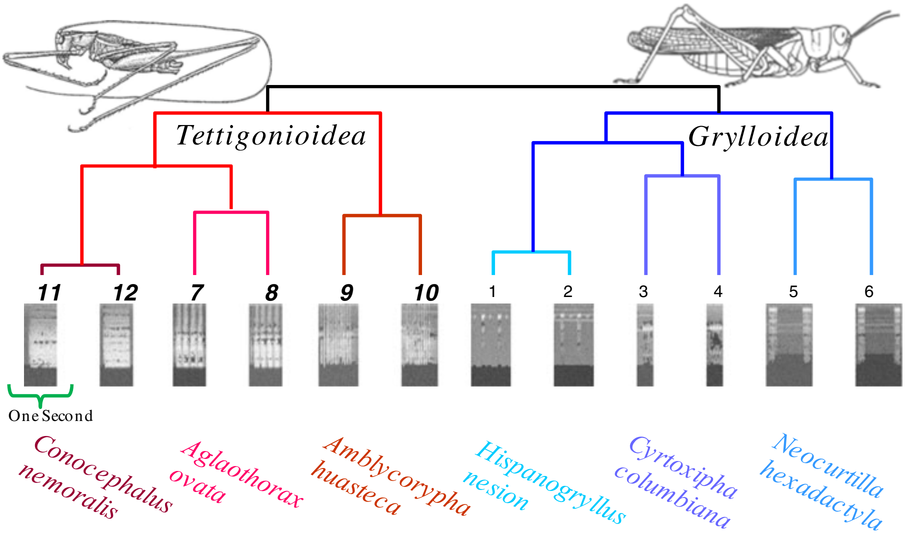

Fig. 5 A clustering of the same data used in Fig. 4, after trimming irrelevant prefix and suffix data. All pairs are correctly grouped, and at a higher level the dendrogram separates katydids and crickets

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 Fig. 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 (Campana and Keogh 2010). 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 (Campana and Keogh 2010) 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:

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

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. We have created an algorithm which can efficiently search such datasets. By "efficient", we mean that the time taken to search a dataset is significantly less than the time it typically takes to collect such data in the field. Our method exploits some state-of-the-art ideas from data mining, and may not be of direct interest to the reader interested only in practical applications of our ideas. We therefore have relegated a detailed discussion of our speedup techniques to Appendix A.

Disjunctive Sound Fingerprints

A single species of insect can produce a variety of sounds for different functions, such as attracting mates, claiming territory, repelling aggressors, etc. In addition, their sounds can be affected by external factors such as ambient temperature (Elliott and Hershberger 2007; Jang and Gerhardt 2007). Thus far, we have only considered using a *single* sound fingerprint to represent a class. However, it is easy to imagine situations where it may not be possible to use a *single* sound fingerprint to separate classes.

To illustrate this, we consider a toy example as shown in Fig. 6i. We can easily differentiate *Species A* from *Species B* with a *single* sound fingerprint A. In particular, every object within threshold T_A belongs to *Species A*, otherwise, it belongs to *Species B*. In contrast, as we show in Fig. 6.*center*, we may have a situation in which both the males and females of a species sing (In most Orthoptera only the males sing, but in some cases, such as the *Magicicada septendecim*, the females also sing (Elliott and Hershberger 2007)), as in *Species C* in our toy example. Moreover, as insects are often sexually dimorphic (Elliott and Hershberger 2007), these songs may be rather different.

In that case, *Species C* contains two well-defined distributions, if we attempt to use only one sound fingerprint to separate *Species C* from *Species D*, almost half of the instances could be misclassified into *Species D* as shown in Fig. 6.*center*. Note that if we increase the threshold (i.e. expand the radius of the gray circle), we will capture more true positives only at the expense of capturing more false positives (labeling *Species D* as into *Species C*).

How can we solve this non-linear separation problem? Inspired by the similar nearest centroid algorithm (MacQueen 2009), we propose to use multiple fingerprints to represent each species (when necessary). As shown in Fig. 6III, we can use *multiple* fingerprints (C_1 and C_2) to represent the *single* concept of *Species C*.

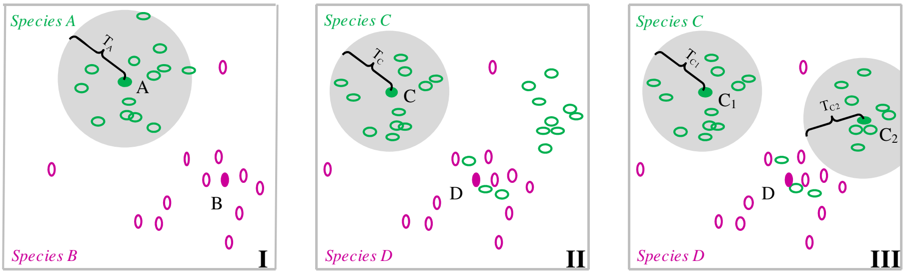

Fig. 6 I) A toy problem showing instances of crickets (*Species A*) and katydids (*Species B*). In this case a single sound fingerprint A can separate the two classes. II) In contrast, *Species C* is an example where a single sound fingerprint cannot separate the classes, perhaps because the insects are sexually dimorphic with the males and females singing differently. III) Two sound fingerprints C_1 and C_2 connected by an *OR* operation *can* better separate the classes

The intuition behind this is that we can augment the representational power of sound fingerprints with the logical **OR** operator. Intuitively, we say that a candidate sound is of class X, if it is similar to *sound fingerprint*$_1$ **OR** *sound fingerprint*$_2$. **OR** *sound fingerprint*$_k$. We call such a representation *Disjunctive Sound Fingerprints*. Each sound fingerprint may correspond to a different behavior for the animal, e.g., mating calls, aggressive utterances, mother-pup interactions, etc. In addition, for insects, the multiple sound fingerprints may correspond to different external conditions which influence their sound production apparatus. For instance, for some insects, especially in *Orthoptera*, it is well known that the external temperature greatly affects the sounds the insects produce (Elliott and Hershberger 2007). Thus, in this case, three or four sound fingerprints could represent the sound a particular insect produces at, say 10 to 20 degrees, 21 to 30 degrees, etc. While we have some such domain knowledge, it is important to note that we do not propose to "hand-craft" the disjunctive sound fingerprints. We wish to preserve the black-box nature of our algorithms.

During the search process, we wish to avoid finding *redundant* fingerprints, that is to say, minor-variations of previously discovered sound fingerprints. To achieve this, objects that are already "explained" (i.e., correctly classified) by a fingerprint are removed before rerunning the search on a now smaller set of data objects. This continues until either we cannot find any sound fingerprint to separate ***P*** from ***U*** or there are less than two objects left in ***P***.

For instance, in Fig. 6.*center*, our algorithm learned C_1 as the first sound fingerprint, however, almost half of the objects are still not classified correctly. Thus a second sound fingerprint C_2 is learned after removing objects which can be described by C_1, as shown in Fig. 6III. However, there are still three remaining objects, which are not encompassed by the gray circles as shown in Fig. 6.*right*. We can keep searching for another sound fingerprint, but this opens the possibility of overfitting, given there is so little data remaining. If these three objects cannot be described by another sound fingerprint that separates them from *Species D*, they are unclassified and the search for sound fingerprints of *Species C* is terminated. Thus, our algorithm finds between zero to K sound fingerprints for a given class. Finding zero fingerprints signals the user that the dataset cannot be represented by a single sound fingerprint and has no exploitable structure.

The formal algorithm to learn multiple fingerprints is shown in Table 2.

In *DisjuntFPDiscovery*(), we call the *SoundFP_Discovery*() procedure (see appendix A) with entropy-based pruning and Euclidean distance ordering heuristic speed up techniques to find the first sound fingerprint (line 2). When the first sound fingerprint is found, we check to see if ***P*** has at least two objects to the right of the distance threshold, because that means using a single sound fingerprint cannot represent ***P*** sufficiently to separate it from ***U***.

Given the CK distance ordering *currentDist* and the *threshold*, if there is more than *one* object of ***P*** with distances larger than the threshold, another sound fingerprint is required to learn to represent these objects. After finding the new sound fingerprint we test the information gain to see if it can separate the two classes further. If the separation improves, we look for a new $\boldsymbol{P_j}$ and continue as before. Once there are less than two objects left in ***P*** or we cannot achieve better separation, we terminate *SoundFP_Discovery*() routine and return sound fingerprint(s).

Table 2 Disjunctive fingerprint discovery

```
DisjuntFPDiscovery (D, $L_{min}$, $L_{max}$)
Require: A dataset D (P and U) of sound sequence's sonogram, user defined minimum length and maximum length of sound fingerprint
Ensure: Return a set of sound fingerprints(can be an empty set)

For index = 1 to |P|+|U|-1
  soundFP, currentDist, threshold, bsf_infoGain = SoundFP_Discovery(D, $L_{min}$, $L_{max}$) with speedup techniques
  For j = 1 to |currentDist|
    If currentDist(j) > threshold
      newP = $P_j$
    EndIf
    If |newP|>1 and |$P_j$|-|newP|>1 and bsf_infoGain will not increase {number of objects in newP, number of objects whose distance to the current fingerprint less than threshold should larger than 1}
      Recursive call SoundFP_Discovery(newP&U, $L_{min}$, $L_{max}$)
      Increment index
      Append new sound fingerprint to soundFP
    Else
      break
    EndIf
  EndFor
EndFor
Return soundFP
```

Experimental Evaluation

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 Fig. 7. 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*" (Roach 2006). Thus, even when handling a living specimen, most entomologists cannot tell them apart without resorting to DNA analysis.

We suspected that we might be able to tell them apart by sound. 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 (Wells and Henry 1998). While we do not have enough data to do forceful and statistically significant experiments, we can do two tentative tests. As shown in Fig. 7, 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.

The results suggest that these congeneric (Species belonging to the same genus are *congeneric*) species are almost linearly separable in two-dimensional space (they *are* linearly separable in three-dimensional space).

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

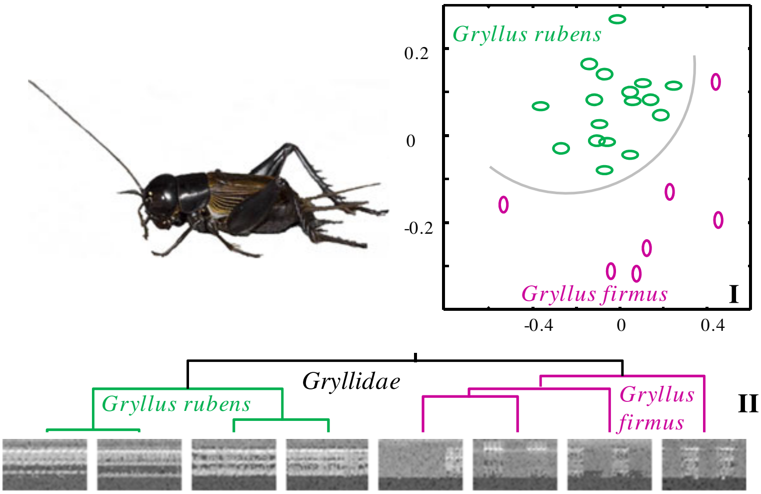

Fig. 7 I) Projecting one-second snippets of songs from both insects into 2-D space suggests they are almost linearly separable, a possibility reflected by their clusterability (II)

unwilling to share their data. To redress this, we created and placed into the public domain a large classification dataset (Hao 2011). The data consists of twenty species of insects, eight of which are *Gryllidae* (crickets) and twelve of which are *Tettigoniidae* (katydids) (A full description of the data is at (Hao 2011)). 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 *profoundly* 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. Likewise, we have multiple representatives from both the *Belocephalus* and *Atlanticus* genera.

We learned 20 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 3.

Table 3 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 (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). 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 good results, we defer such discussions to future work.

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 Fig. 8, 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. As we can see in Fig. 8, 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.

Similar to the experiment above, we also learned the sound fingerprints for three frog species, *Bufo alvarius*, *Bufo canorus*, and *Pseudacris crucifer*. We learned sound fingerprints from the training sets consisting of ten frogs for each species.

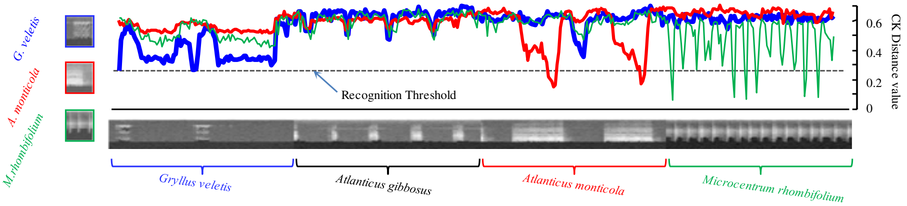

Fig. 8 (Image best viewed in color) Three insect sound fingerprints are used to monitor an eight- second audio clip on the left. In each case, the fingerprint distance to the sliding window of audio dips below the threshold as the correct species sings, but not when a different species is singing

To allow visual appreciation of our method, as shown in Fig. 9 we produced a fourteen-second sequence of audio by concatenating snippets of seven different species, including holdout (i.e. *not* seen in the training set) examples from our three species of interest. Again, while each fingerprint has a slightly different threshold, for simplicity and visual clarity we show just the averaged threshold. As we can see in Fig. 9, this method also achieves three true positives, and more significantly, no false positives.

Robustness to Mislabeled Data

No matter how carefully the animal audio dataset is obtained, we must resign ourselves to the possibility of mislabeled data. We claim our algorithm is robust to mislabeled data so long as the *majority* of data in P is correctly labeled. To test our algorithm's robustness to mislabeled training data, we will give a simple example using one katydid data (*Atlanticus dorsalis*).

There are ten objects in ***P*** and ten objects in ***U***; we ran our algorithm and found the sound fingerprints shown on the left of Fig. 10I. To simulate mislabeled training data, we randomly swapped two objects in ***P*** with two objects in ***U***, creating 20 % mislabeled training data. We relearned sound fingerprints from this mislabeled data and discovered the sound fingerprint as shown on the left of Fig. 10II. As in Section 5.3, we concatenated the testing data in ***P*** and ***U*** in a round-robin fashion to obtain a forty-second long audio clip to test the monitoring accuracy of the two learned sound fingerprints.

As shown in Fig. 10, the fingerprint learned from the correctly labeled data achieves nine (out of ten) true positives and one false negative. The fingerprint learned from the mislabeled data also achieves nine (out of ten) true positives and a single false negative, although the location of false negative is different.

These results suggest that our algorithm is largely invariant to small amounts of mislabeled data.

Insect Classification with Disjunctive Sound Fingerprints

To this point, we have evaluated *single* sound fingerprint representation. We will now test the expressive power of *disjunctive* sound fingerprints. Recall the toy example in Fig. 6. We showed that our dataset may have multiple "clusters" of sound within a single species, and further proposed a technique to learn multiple sound fingerprints

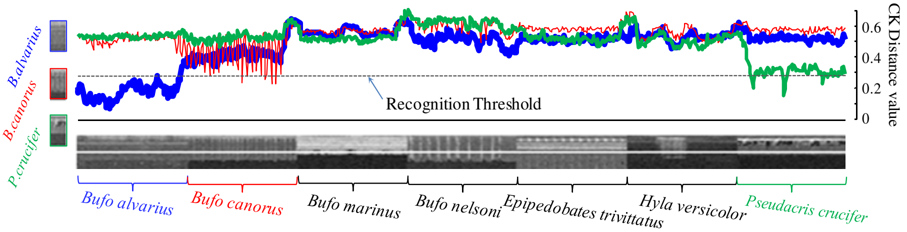

Fig. 9 (Image best viewed in color) Three frog sound fingerprints are again used to monitor a fourteen-second audio clip on the left. In each case, the fingerprint distance to the sliding window of audio dips below the threshold as the correct species sings, but not when a different species is singing

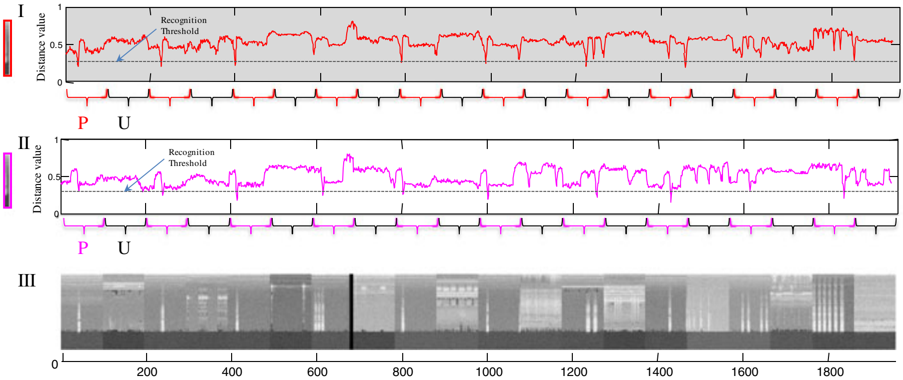

Fig. 10 I, II) Sound fingerprint of *Atlanticus dorsalis* is used to monitor a twenty-second testing audio clip without (with) mislabeled data in *P*. The fingerprint distance to the sliding window of audio dips below the threshold as the correct species sings, but not when a different species is singing. III) Forty second sequence of audio by concatenating snippets of testing audio clips from *P* and *U* alternatively

to represent the class concept. We will consider two examples to demonstrate the utility of the disjunctive sound fingerprint discovery algorithm for improving classification accuracy.

Insect Dataset I

The first dataset consists of two classes, which are Crickets (*Anaxipha litarena* and *Anurogryllus celerinictus*) and Katydids (*Amblycorypha carinata* and *Neoconocephalus bivocatus*). Note that there are intrinsically two subclasses in each class, although our algorithm is not "aware" of this.

There are twenty training and twenty testing objects for each class, all of which are two-seconds long. We ran the disjunctive sound fingerprints discovery algorithm on this problem. We find that two sound fingerprints described the class Crickets. Gratifyingly, the two sound fingerprints correspond to the two subclasses of crickets, the first came from *A.celerinictus* and the second from *A.litarena*. However, we only find one sound fingerprint to describe Katydids, presumably this is because the two subclasses of Katydids were more similar to each other.

We predicted the testing data class labels by sliding each fingerprint across each object and recording the fingerprint that produced the minimum value as the exemplar's nearest neighbor. Disjunctive sound fingerprints reduced the classification error rate from 0.325 (with a single sound fingerprint, as in Section 5.2.) to 0.175.

Insect Dataset II

The second dataset also has two classes, which we again labeled as Crickets (*A.litarena*, *A. celerinictus*, *Gryllus fultoni*, and *Velarifictorus micado*) and Katydids (*A.carinata*, *Atlanticus dorsalis*, *Belocephalus sabalis*, and *Neoconocephalus retusus*). There are intrinsically four subclasses in each class, but as before our algorithm does not have access to this information.

We have forty training and forty testing two-second instances for each class. After running the disjunctive sound fingerprint discovery algorithm in Table 2, we find that four sound fingerprints represented the class Crickets. The first one is from *A.litarena*, the second one is from *A. celerinictus*, the third one is from *G. fultoni*, and the fourth one is from *V. micado*. However, we only find three sound fingerprints to represent Katydids. The first one is from *A.carinata*, the second one is from *N.retusus*, and the third one is from *B.sabalis*. As before we suspect that the reason is that two species *A.carinata* and *A. dorsalis* are similar to each other. Once again the disjunctive sound fingerprints reduced the classification error rate, this time from 0.45 (with a single sound fingerprint, as in Section 5.2.) to 0.325.

Insect Dataset III

To test the disjunctive sound fingerprints discovery algorithm to find an appropriate number of sound fingerprints for a given class, we will provide a simple example to show that our algorithm did not find redundant fingerprints. This dataset also has two classes, Crickets (*A.litarena*) and Kaytdids (*A. dorsalis*). We have ten training and ten testing objects for each class. By visual, manual inspection alone, the diversity among each class is not large so a single sound fingerprint should be able to represent one class. To test our prediction, we run the disjunctive sound fingerprint discovery algorithm in Table 2. We found only one sound fingerprint for each class and the classification error rate was 0.15. The result suggests that our disjunctive sound fingerprint algorithm can both avoid overfitting and help to reduce the classification error rate.

Beyond improving the classification accuracy, disjunctive fingerprints may also be useful for exploratory data mining, telling us something about the data that would otherwise be difficult to discover.

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 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.

Acknowledgments We thank the Cornell Lab of Ornithology for donating much of the data used in (Macaulay Library, www.macaulaylibrary.org/index.do), Jesin Zakaria for help projecting snippets of sounds into 2D space, and Dr. Agenor Mafra-Neto for entomological advice.

Appendix A: Speeding Up the Search for Sound Fingerprints

In the main text, we defined sound fingerprints concretely, but did not discuss how we find them. We have regaled that discussion to this appendix to enhance the flow of the paper.

Table 4 Brute-force sound fingerprint discovery

```
SoundFP_Discovery(D, $L_{min}$, $L_{max}$)
Require: A dataset D (P and U) of sound sequence's sonogram, 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 sonogram 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
```

As we noted in the main text, there may be huge number of candidate sound fingerprints to examine and score; thus, we must provide an efficient mechanism for searching for the best one for a given species. 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.

A Brute-Force Algorithm

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.

The brute force algorithm is described in Table 4. 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 16 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 Fig. 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 5.

Table 5 Check the utility of single candidate

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

In the subroutine *CheckCandidates*(), shown in Table 5, 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 6. In essence, this is the procedure illustrated in Fig. 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 6 Compute minimum subsequence CK distance

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

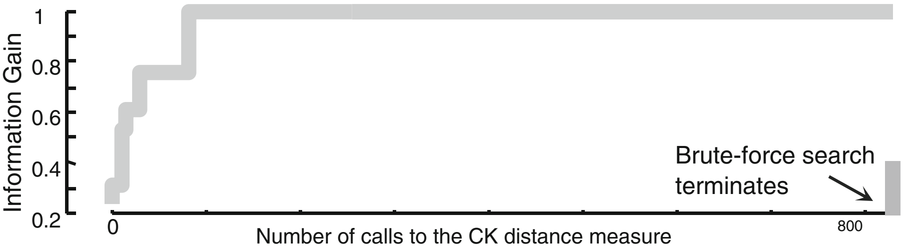

Fig. 11 A trace of value of the *bsf_Gain* variable during brute force search on the *Atlanticus dorsalis* dataset. Only sound fingerprints of length 16 are considered here for simplicity

We do this for every sonogram 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. Figs. 2 and 12). In *minCKdist*() (Table 6), we use the CK measure (6. Campana and Keogh 2010) as the distance measurement between a candidate fingerprint and a generally much longer sonogram.

In Fig. 11 we show a trace of the brute force algorithm on the *Atlanticus dorsalis* problem.

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 h. 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.

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 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

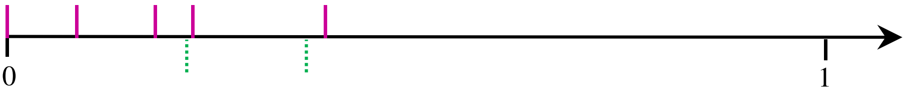

Fig. 12 The order line of all the objects in *P* and just the first two objects in *U*

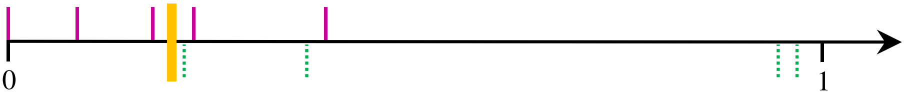

Fig. 13 The logically best possible order line based on the distances that have been calculated in Fig. 12. The best split point is shown by the yellow/heavy line

"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 4), 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 Fig. 2. Assume that the partially computed linear ordering is shown in Fig. 12. We have computed the distances to all five objects in ***P***, and to 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 Fig. 13.

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 Fig. 13 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 Fig. 12 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 5. However, as we shall see, this simple idea can prune more than 95 % of the calculations for more realistic problems.

Table 7 Entropy upper bound pruning

```
EntropyUBPrune ($U_m$, currentDist, $S_{k,l}$, bsf_Gain)
Require: A sound sequence's sonogram $U_m$, current distance ordering,
sound fingerprint candidate $S_{k,l}$, best-so-far information gain
Ensure: Return fraction of distance computations in U
fraction = 0
counter = 0
rightmostDist = largest distance value in currentDist + 1
bestDist = Add rightmostDist for $U_m$ to currentDist
gain = CheckCandidates(bestDist, $S_{k,l}$)
 If gain >bsf_Gain
    Return False and increment counter
 Else
    Return True
 EndIf
Return fraction = counter/|U|, gain
```

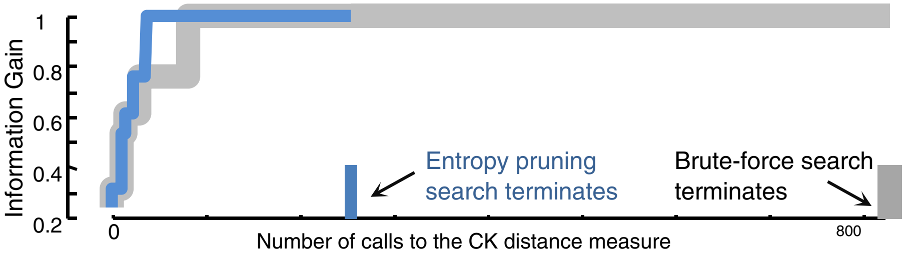

Fig. 14 A trace of the *bsf_Gain* variable during brute force and entropy pruning search on the *Atlanticus dorsalis* dataset

The formal algorithm of admissible entropy pruning is shown in Table 7. After the very first sound fingerprint candidate check, for all the remaining candidates, we can simply insert *EntropyUBPrune*() in line 4 of Table 5, 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.

We can get a hint as to the utility of this optimization by revisiting the *Atlanticus dorsalis* problem we considered above. Figure 14 shows the difference entropy pruning makes in this problem.

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 (Xi et al. 2008).

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,

Table 8 Euclidean distance measure order pruning

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

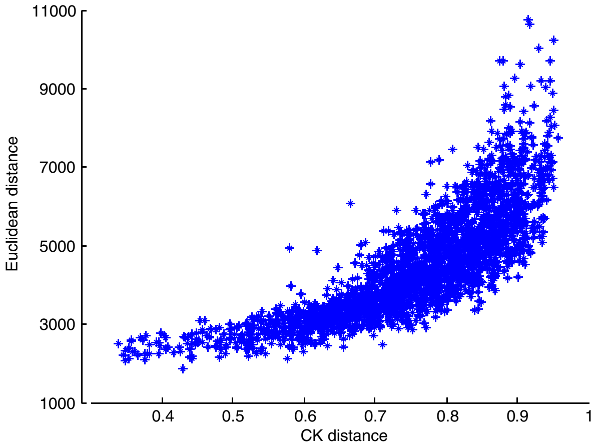

Fig. 15 The relationship between Euclidean and CK distance for 1,225 pairs of sonograms

we do not need to find the *optimal* ordering, we just need to encounter a relatively good candidate relatively early in the search. Table 8 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 4 to obtain a better ordering to check all the candidates.

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 2-D scatter plot, as shown in Fig. 15. The results suggest that Euclidean distance is a good surrogate for CK distance.

To measure the effect of this reordering heuristic we revisited our running example shown in Figs. 11 and 14.

The Euclidean distance reordering heuristic is shown in Fig. 16.

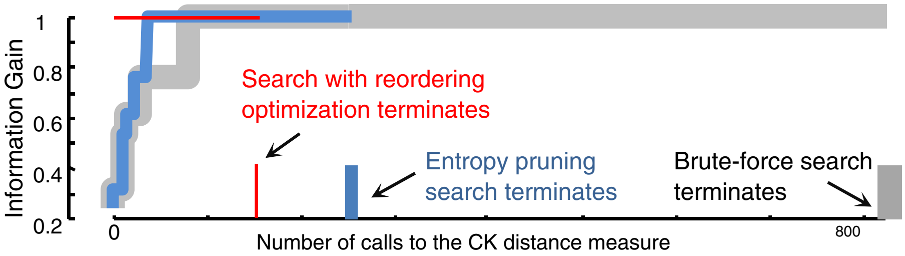

Fig. 16 A trace of value of the *bsf_Gain* variable during brute force, entropy pruning, and reordering optimized search on the *Atlanticus dorsalis* dataset

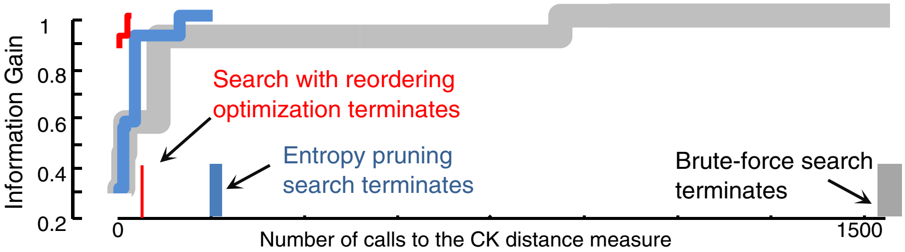

Fig. 17 A trace of value of the *bsf_Gain* variable during brute force, entropy pruning, and reordering optimized search on the *Atlanticus dorsalis* dataset with the 200-object universe

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 (Xi et al. 2008).

Scalability of Fingerprint Discovery

In the experiments shown above, 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 Fig. 17, the speedup achieved by our reordering optimization algorithm is a factor of 93 in this case.

In essence, we believe that our algorithm is fast enough for most practical applications. In particular the time taken for our algorithm will typically be dwarfed by the time taken to collect the data in the field or type up field notes etc.

References

Bardeli R (2009) Similarity search in animal sound databases. IEEE Trans Multimed 11(1):68–76

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

Bianconi F, Fernandez A (2007) Evaluation of the effects of Gabor filter parameters on texture classification. Pattern Recogn 40(12):3325–35

Blumstein DT (2001) Acoustic monitoring in terrestrial environments using microphone arrays: applications, technological considerations, and prospectus. J Appl Ecol 48:758–767

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

Campana BJL, Keogh EJ (2010) A compression-based distance measure for texture. Stat Anal Data Min 3 (6):381–398

Dang T, Bulusu N, Feng WC, Hu W (2010) RHA: A Robust Hybrid Architecture for Information Processing in Wireless Sensor Networks. In 6th ISSNIP

Dietrich C, Schwenker F, Palm G (2001) Classification of time series utilizing temporal and decision fusion. Proceedings of Multiple Classifier Systems (MCS), LNCS 2096, pp 378–387, Cambridge

Desutter-Grandcolas L (1998) First Analysis of a Disturbance Stridulation in Crickets, Brachytrupes tropicus (Orthoptera: Grylloidea: Gryllidae). J Insect Behav 11

Elliott L, Hershberger W (2007) The songs of insects. Houghton-Mifflin Company

Fu A, Keogh EJ, Lau L, Ratanamahatana CA, Wong RCW (2008) Scaling and time warping in time series querying. VLDB J 17(4):899–921

Han NC, Muniandy SV, Dayou J (2011) Acoustic classification of Australian anurans based on hybrid spectral-entropy approach. Appl Acoust 72(9):639–645

Hao Y (2011) Animal sound fingerprint Webpage. www.cs.ucr.edu/~yhao/animalsoundfingerprint.html

Holy TE, Guo Z (2005) Ultrasonic songs of male mice. PLoS Biol 3:e386

Jang Y, Gerhardt HC (2007) Temperature effects on the temporal properties of calling songs in the crickets Gryllus fultoni and G.vernalis: implications for reproductive isolation in sympatric populations. J Insect Behav 20(1)

Keogh EJ, Lonardi S, Ratanamahatana CA, Wei L, Lee S, Handley J (2007) Compression-based data mining of sequential data. DMKD 14(1):99–129

Kogan JA, Margoliash D (1998) 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

Li M, Chen X, Li X, Ma B, Vitanyi P (2003) The similarity metric. Proc'of the 14th Symposium on Discrete Algorithms, pp 863–72

MacQueen JB (2009) Some methods for classification and analysis of multivariate observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp 281–297. MR0214227. ZbI0214.46201

Mankin RW, Hagstrum DW, Smith MT, Roda AL, Kairo MTK (2011) Perspective and promise: a century of insect acoustic detection and monitoring. Amer Entomol 57:30–44

Marcarini M, Williamson GA, de Garcia LS (2008) Comparison of methods for automated recognition of avian nocturnal flight calls. ICASSP 2008:2029–32

Mellinger DK, Clark CW (2000) Recognizing transient low-frequency whale sounds by sonogram correlation. J Acoust Soc Am 107(6):3518–29

Mitrovic D, Zeppelzauer M, Breiteneder C (2006) Discrimination and retrieval of animal sounds. In Proc. of IEEE Multimedia Modelling Conference, Beijing, China, 339–343

Celis-Murillo A, Deppe JL, Allen MF (2009) Using soundscape recordings to estimate bird species abundance, richness, and composition. J Field Ornithol 80:64–78

Nowak DJ, Pasek JE, Sequeira RA, Crane DE, Mastro VC (2001) Potential effect of Anoplophora glabripennis on urban trees in the United States. J Entomol 94(1):116–122

Panksepp JB, Jochman KA, Kim JU, Koy JJ, Wilson ED, Chen Q, Wilson CR, Lahvis GP (2007) Affiliative behavior, ultrasonic communication and social reward are influenced by genetic variation in adolescent mice. PLoS One 4:e351

Riede K, Nischk F, Thiel C, Schwenker F (2006) Automated annotation of Orthoptera songs: first results from analysing the DORSA sound repository. J Orthoptera Res 15(1):105–113

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

Schmidt AKD, Riede K, Römer H (2011) High background noise shapes selective auditory filters in a tropical cricket. J Exp Biol 214:1754–1762

Wells MM, Henry CS (1998) 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

Xi X, Ueno K, Keogh EJ, Lee DJ (2008) Converting non-parametric distance-based classification to anytime algorithms. Pattern Anal Appl 11(3–4):321–36

Yu G, Slotine JJ (2009) Audio classification from time-frequency texture. IEEE ICASSP, pp 1677–80