Automated Ensemble Extraction and Analysis of Acoustic Data Streams

Eric P. Kasten and Philip K. McKinley
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824
{kasten,mckinley}@cse.msu.edu

Stuart H. Gage
Department of Entomology
Michigan State University
East Lansing, Michigan 48824
gages@msu.edu

Abstract

This paper addresses the design and use of distributed pipelines for automated processing of sensor data streams. In particular, we focus on the detection and extraction of meaningful sequences, called ensembles, from acoustic data streamed from natural areas. Our goal is automated detection and identification of various species of birds. Although this target application is relatively specific, the process employed is general and can be extended to other problem domains such as security systems and military reconnaissance.

1 Introduction

Advances in technology have enabled new approaches for sensing the environment and collecting data about the world; an important application domain is ecosystem monitoring [1–3]. Small, powerful sensors can collect data and extend our perception beyond that afforded by our natural biological senses. Moreover, wireless networks enable data to be acquired simultaneously from multiple geographically remote and diverse locations. Once collected, sensor readings can be assembled into data streams and transmitted over computer networks to observatories, such as the National Ecological Observatory Network (NEON) [4, 5], which provide computing resources for the storage, analysis and dissemination of environmental and ecological data. Such information is important to improving our understanding of environmental and ecological processes. For instance, early detection and tracking of invasive species may enable the establishment of policies for their control [6]. When data is continuously collected, automated and adaptive processing facilitates the organization and searching of the resulting data repositories. Without timely processing, the sheer volume of the data might preclude the extraction of information of interest.

The main contribution of this paper is to introduce a process that enables detection and extraction of meaningful sequences, called *ensembles*, from acoustic data streams. Here we apply this method to support automated detection and classification of bird species using a perceptual memory system that supports online, incremental learning [7]. Results of our classification experiments are promising and suggest that acoustics can enable automated monitoring of natural environments. Moreover, the extraction of ensembles from acoustic clips reduced the amount of data to be processed by approximately 80%. To support the study, we designed and implemented a dynamic distributed pipeline prototype, called Dynamic River, which enables sets of operators to be dynamically relocated to more suitable hosts to improve quality-of-service.

The remainder of this paper is organized as follows. Section 2 describes background on data collection and processing methods. Section 3 describes in detail the approach for ensemble extraction, and Section 4 presents the results of our experiments using ensemble extraction for classification of bird species. Section 5 describes related work, and Section 6 concludes the paper. Due to space limitations, many details of this study are omitted here, but may be found in an accompanying technical report [8].

2 Background

Data Collection. This study addresses the automated classification of bird species using acoustic data streams collected in natural environments. Acoustic data is collected from in field sensor stations located at the Kellogg Biological Research Station (KBS) and other locations in Michigan. Figure 1(a) shows an acoustic sensor station used in this study. Each station comprises a pole-mounted sensor unit and a solar panel coupled with a deep cycle battery for

providing power over extended periods. Figure 1(b) shows the internal components of the sensor unit. Each sensor unit contains a Crossbow Stargate processing platform equipped with a microphone and an 802.11b wireless interface card. The Stargate platform has a 400MHz Intel PXA225 processor and 64MB of RAM. The operating system used is TinyOS. Acoustic clips are collected by the sensor units and transmitted over a wireless network for later relay to the CEVL. Currently, clips are approximately 30 seconds long, comprising approximately 1.26MB of data, and are collected every 30 minutes.

(a) Sensor station (b) Sensor unit

Figure 1. Acoustic sensor station and unit.

Bird vocalizations vary considerably even within a particular bird species. Young birds learn their songs with reference to adult vocalizations during sensitive periods [9]. At maturity, the song of a specific bird will crystallize into a species-specific stereotypical form. However, even stereotypical songs vary between individual birds of the same species. Moreover, many vocalizations are not stereotypical but are instead plastic, and may change when sung or due to seasonal change, while some species can learn new songs throughout their lives. Extraction of candidate bird vocalizations from acoustic streams enables accurate recognition of a species, where misidentification one species as another should be avoided.

Time series processing. Figure 2 depicts two common methods for visualizing an acoustic clip. The top graph shows a plot of the signal's amplitude, or oscillogram, normalized by subtracting the mean and scaling by the maximum amplitude. The bottom graph shows the same clip plotted as an acoustic spectrogram. A spectrogram depicts frequency on the vertical axis and time on the horizontal axis. Shading indicates the intensity of the signal at a particular frequency and time. In this study, spectrogram segments are distilled into signatures that can be used to identify the bird species that produced a particular vocalization.

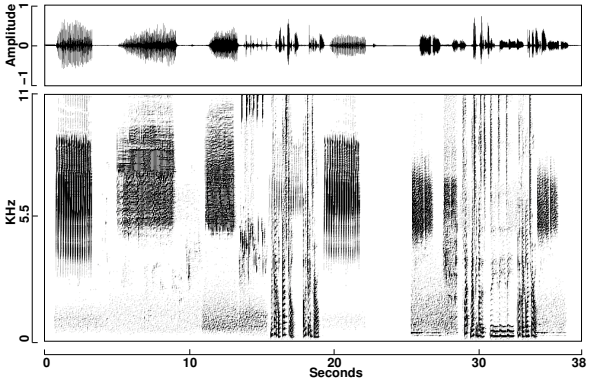

Figure 2. An oscillogram (top) and spectogram (bottom) of an acoustic signal.

Piecewise aggregate approximation (PAA) was introduced by Keogh et al. [10], and independently by Yi and Faloutsos [11], as a means to reduce the dimensionality of time series. An original time series sequence, Q, of length n is converted to PAA representation, $\overline{Q}$. First, Q is Z-normalized as follows:

$$\forall i \quad q_i = \frac{q_i - \mu}{\sigma},$$

where μ is the vector mean of original signal, σ is corresponding standard deviation and q_i is the i^{th} element of Q. Second, Q is segmented into $w \leq n$ equal sized subsequences, and the mean of each subsequence computed. $\overline{Q}$ comprises the mean values for all subsequences of Q.

PAA smoothes intra-signal variation and reduces pattern dimensionality, while Z-normalization helps equalize similar acoustic patterns that differ in signal strength. Figure 3 depicts the spectrogram shown in Figure 2 after conversion to PAA representation. This spectrogram was constructed by applying PAA to the frequency data comprising each column of the original spectrogram. Despite smoothing and reduction using PAA, these spectrograms are similar in appearance, demonstrating the potential utility of using PAA representation.

Extending the benefits of PAA is a representation introduced by Lin et al. [10] called *Symbolic Aggregate approXimation* (SAX). The purpose of SAX is to enable accurate comparison of time series using a symbolic representation. As shown in Figure 4, SAX converts a sequence from PAA representation to symbolic representation, where each symbol (we use integers here) appears with equal probability based on the assumption that the distribution of time series subsequences is Gaussian [10].

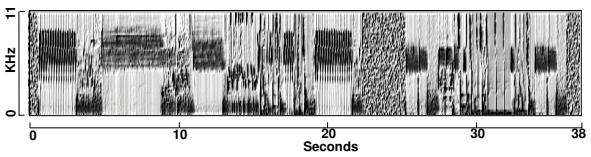

Figure 3. Spectrogram of the acoustic signal (see Figure 2) after conversion to PAA representation (stretched vertically for clarity).

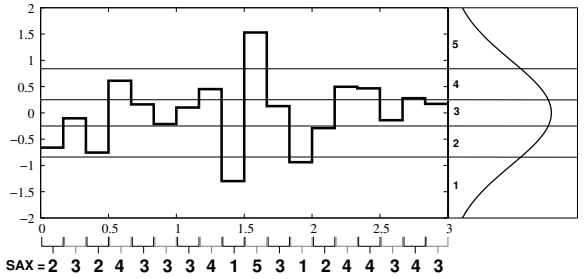

Figure 4. Conversion of the example PAA processed signal converted to SAX (adapted from [10]).

Kumar et al. [12] proposed *time series bitmaps* for visualization and anomaly detection in time series. SAX bitmaps are constructed by counting occurrences of symbolic subsequences of length n (e.g., 1, 2 or 3 symbols). Each bitmap can be represented using an n-dimensional matrix, where each cell represents a specific subsequence. Each cell contains the frequency with which the corresponding subsequence occurs. Frequencies are computed by dividing the subsequence count by the total number of subsequences. An anomaly score can be computed by comparing two concatenated bitmap matrices using Euclidean distance. As further discussed in Section 3, we use SAX bitmap matrices to compute an anomaly score for acoustic signals, enabling the extraction of bird vocalizations and other acoustic events.

MESO. For classification and detection experiments we use MESO [7], a perceptual memory system designed to support online, incremental learning and decision making in autonomic systems. MESO is based on the well-known leader-follower algorithm, an online, incremental technique for clustering a data set. A novel feature of MESO is its use of small agglomerative clusters, called *sensitivity spheres*, that aggregate similar training patterns. Once MESO has been trained, the system can be queried using an unlabeled pattern. MESO tests the new pattern and returns the label associated with the most similar training pattern or a sensitivity sphere containing a set of similar training patterns and their labels. When evaluated on standard data sets, MESO accuracy compares very favorably with other classifiers, while requiring less training and testing time in most cases [7].

Dynamic River. We have developed a prototype system, Dynamic River [8], that enables the construction of a distributed stream processing pipeline. A Dynamic River *pipeline* is defined as a sequential set of operations composed between a data source and it's final sink (destination). The network operators enable record processing to be distributed across the processor and memory resources of many hosts. Pipeline *segments* are created by composing sequences of operators that produce a partial result important to the overall pipeline application. Segments can receive and emit records using the `streamin` and `streamout` operators, respectively, enabling instantiation of segments and the construction of a pipeline across networked hosts. Moreover, pipelines can be recomposed dynamically by moving segments among hosts.

Preserving the integrity of data streams in the presence of a dynamic environment is a challenging problem. Dynamic River records can be grouped using `record_subtype`, `scope` and `scope_type` header fields. We define a *data stream scope* as a sequence of records that share some contextual meaning, such as having been produced from the same acoustic clip. Within the data stream, each scope begins with an `OpenScope` record and ends with a `CloseScope` record. Optionally, `CloseScope` records can be replaced with `BadCloseScope` records to enable scope closure while indicating that the scope has not reached its intended point of closure. For instance, if an upstream segment terminates unexpectedly and leaves one or more scopes open, the `streamin` operator will generate `BadCloseScope` records to close all open scopes,

Scopes can be nested. The `scope` field indicates the current scope nesting depth, larger values indicate greater nesting while scope depth 0 indicates the outermost scope. The `scope_type` field enables the specification of an application specific scope type. For instance, a scope can be identified as comprising an acoustic clip or an ensemble. Optionally, `OpenScope` records may contain context information, such as the sampling rate of an acoustic clip. Scoping can also be used to support graceful shutdown and fault tolerance in streaming applications [8].

3 Ensemble Extraction and Processing

A sensor data stream is a time series comprising continuous or periodic sensor readings. Typically, readings taken from a specific sensor can be identified and each reading appears in the time series in the order acquired. Online clustering or detection of interesting sequences benefits from time-

efficient, distributed processing that extracts finite candidate sequences from the original time series.

We define *ensembles* as time series sequences that recur, though perhaps rarely. This definition is similar to other time series terms. For instance, a *motif* [13] is defined as a sequence that occurs frequently and a *discord* [14] is defined as the sequence that is least similar to all other sequences. A notable limitation for finding a discord in a time series is that the time series must be finite. Our use of ensembles addresses this limitation by using a finite window for computing an anomaly score and thereby detecting a distinct change in time series behavior. An anomaly score greater than a specified threshold is considered as indicating the start of an ensemble that continues until the anomaly score falls below the threshold.

Figure 5 depicts a typical approach to data acquisition and analysis using a Dynamic River pipeline that targets ecosystem monitoring using acoustics. First, audio clips are acquired by a sensor platform and transmitted to a `readout` operator that writes the clips to `record` for storage. Although additional record processing is possible prior to storage, it is often desirable to retain a copy of the raw data for later study. During analysis, a data `feed` is invoked to read clips from storage and write them to `wav2rec` to encapsulate acoustic data (WAV format in this case) in pipeline records. The remaining operators comprise the process for extracting ensembles and processing them for classification or detection using MESO.

The operators `saxanomally`, `trigger`, and `cutter` compose a pipeline segment that transforms records comprising acoustic data into ensembles. The incoming record stream is scoped, with each clip delimited by an `OpenScope`/`CloseScope` pair. The outgoing record stream comprises ensembles that are also delimited by an `OpenScope`/`CloseScope` pair. The clip and ensemble scopes are typed, using the `scope_type` record header field, as `scope_clip` or `scope_ensemble` respectively.

The moving average of the SAX anomaly score, as described in Section 2, is output by `saxanomaly` in addition to the original acoustic data. Parameters, such as the SAX anomaly window size, SAX alphabet size and a moving average window size, can be set to meet the needs of a particular application or data set. The SAX anomaly window size specifies the number of samples to use for constructing each concatenated matrix used for computing the SAX anomaly score, for a given SAX alphabet. The moving average window size specifies the number of anomaly scores to use for computing a mean anomaly score that is output by `saxanomaly`. Using a moving average smoothes anomaly score "spikes" over a longer period that can be used as a window of anomalous behavior by the `cutter` operator. In our experiments with environmental acoustics, we

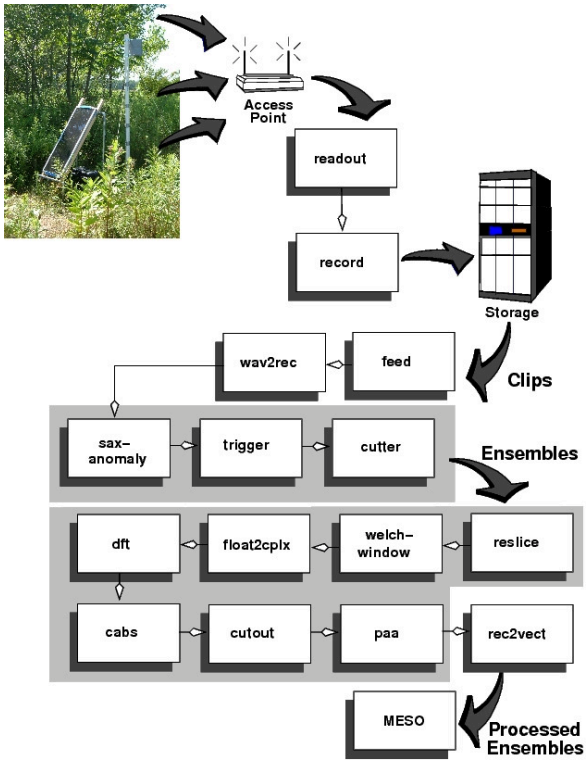

Figure 5. Block diagram of pipeline operators for converting acoustic clips into ensembles for detection of bird species.

set the moving average window to 2250 samples, the SAX anomaly window to 100 samples and the SAX alphabet size to 8.

Figure 6 depicts the trigger signal output by the `trigger` operator (top) and the corresponding ensembles extracted from the original acoustic signal depicted in Figure 2 by the `cutter` operator (bottom). The `trigger` operator transforms the anomaly score output by `saxanomaly` into a trigger signal that has the discrete values of either 0 or 1. The `trigger` operator is adaptive in that it incrementally computes an estimate of the mean anomaly score, μ_0, for values when the trigger value is 0. `Trigger` emits a value of 1 when the anomaly score is more than 5 standard deviations from μ_0 and a 0 otherwise. The number of standard deviations is specific to the particular data set or application.

The `cutter` operator reads both the records containing the original acoustic signal and the records emitted by `trigger`. When the trigger signal transitions from 0 to 1, `cutter` emits an `OpenScope` record, designating the start of an ensemble, and begins composing an ensemble. Each ensemble comprises values from the original acoustic

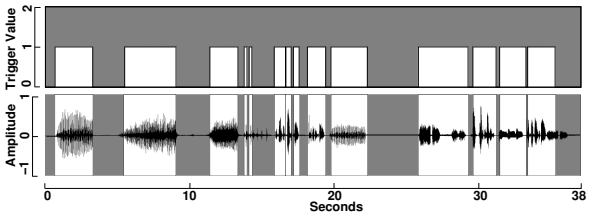

Figure 6. Trigger signal and ensembles extracted from the acoustic signal shown in Figure 2.

signal that correspond to when the trigger value is 1. When the trigger value transitions from 1 to 0, `cutter` emits a `CloseScope` record, and resumes consuming acoustic values until the trigger value again transitions to 1. The record stream, as emitted from `cutter`, comprises clips that contain one or more ensembles.

The operators, `reslice`, `welchwindow`, `float2cplx`, `dft` and `cabs` compose a pipeline segment that transforms the amplitude data of each ensemble into a frequency domain (power spectrum) representation. First, for each pair of ensemble records, the `reslice` operator constructs a new record comprising the last half of the first record and the second half of the second original record. This new record is then inserted into the record stream between the two original records. The remainder of the pipeline segment computes a floating point representation of each ensemble's spectrogram, where each ensemble comprises one or more records of spectral data. This computation proceeds as follows: the `welchwindow` operator applies a Welch window to each "resliced" record, helping minimize edge effects between records; then `float2cplx` converts each value to complex number format required by the `dft` operator for computing the discrete Fourier transform; finally, `cabs` computes the complex absolute value of each complex value, emitted by `dft`, as a floating point value.

Next, each record of each ensemble is passed to the `cutout` operator. The `cutout` operator selects specific frequency ranges from each record and emits records comprising only these ranges. Data outside of the selected range is discarded. For our classification experiments, the frequency range ≈[1.2kHz,9.6kHz] was `cutout`. Frequencies above and below this range typically have little data useful for classification or detection of bird species. Moreover, data below this range typically comprises low frequency noise, including the sound of wind and sounds produced by human activity.

The optional `paa` operator reduces each record to a PAA representation as discussed in Section 2. For our experiments, we used records that were either reduced by a factor of 10 using PAA or that were not reduced. The effectiveness of using PAA representation for smoothing acoustic spectral data is demonstrated in Section 4. Finally, the `rec2vect` operator converts pipeline records to vectors of floating point values (patterns), suitable for use in our classification and detection experiments with MESO.

4 Assessment

Listed in Table 1 are the four-letter species codes and the common names for the 10 bird species whose vocalizations we use in our experiments. Also listed are the number of individual patterns and ensembles extracted from the recorded vocalizations and included in our experimental data sets. For testing classification accuracy, we used four data sets produced from a set of audio clips; each extracted ensemble contains the vocalization from one of the 10 bird species. Although each ensemble contains the vocalization for only a single species, the clips typically contain other sounds such as those produced by wind and human activity. Additionally results can be found in the technical report [8].

Table 1. Bird species codes, names and counts.

Code	Common name	Patterns	Ensembles
AMGO	American goldfinch	229	42
BCCH	Black capped chickadee	672	68
BLJA	Blue Jay	318	51
DOWO	Downy woodpecker	272	50
HOFI	House finch	223	26
MODO	Mourning dove	338	24
NOCA	Northern cardinal	395	42
RWBL	Red winged blackbird	211	27
TUTI	Tufted titmouse	339	59
WBNU	White breasted nuthatch	676	84

Ensemble data sets. Two ensemble data sets, comprising 473 ensembles, were produced using the method described in Section 3. The data sets differ in that one was processed with PAA while the other was not. The ensembles produced by the `cutter` operator were validated by a human listener as being a bird vocalization. The validated ensembles were then fed to the `dft` operator for further processing (refer to Figure 5). Each ensemble comprises one or more patterns. Each pattern was constructed by merging 3 frequency domain records. A single pattern represents 0.125 seconds of acoustic data in the range ≈[1.2kHz,9.6kHz] and comprises either 1050 features or, when processed with PAA, 105 features. A voting approach is used for testing each ensemble, specifically each pattern belonging to a given ensemble is tested independently and represents a "vote" for the species indicated by the test. The species with the most votes is returned as the recognized species.

Pattern data sets. Each of the two pattern data sets comprises 3,673 patterns extracted from the 473 ensembles in the ensemble data sets. Like the ensemble data sets, each

pattern has either 1050 or 105 features and represents 0.125 seconds of acoustic data. Ensemble grouping is not retained and, as such, recognition is based on testing with a single pattern.

Experimental method and assessment. Extraction of ensembles from acoustic clips reduced the amount of data that required further processing by 80.6%. As such, automated ensemble extraction helps address the need for timely processing of large volumes of data found when continuously collecting sensor readings. To verify the viability of using ensembles for birdsong recognition, we tested classification accuracy using cross-validation experiments as described by Murthy et al. [15] using a leave-one-out approach [16]. The leave-out-out approach was used due to the high variability found in bird vocalizations and the relatively small size of the data sets. Each experiment is conducted as follows:

1. Randomize the data set. For the ensemble data set, randomize the order of the ensembles. For the pattern data set, randomize the order of the patterns.
2. In turn select each ensemble/pattern as a test pattern, train MESO using all remaining data. Test MESO using the single selected ensemble/pattern.
3. Calculate the classification accuracy by dividing the sum of all correct classifications by the total number of ensemble/patterns.
4. Repeat the preceding steps n times, and calculate the mean and standard deviation for the n iterations.

In our leave-one-out tests, we set n equal to 20. Thus, for each mean and standard deviation calculated, MESO is trained and tested 9,460 times in the case of the ensemble data set and 73,500 times in the case of the pattern data set.

We also executed a resubstitution test, where MESO was both trained and tested using the entire data set. Although lacking statistical independence between training and testing data, resubstitution affords an estimate of the maximum classification accuracy expected for particular data set. Each experiment is conducted as follows:

1. Randomize the data set. For the ensemble data set, randomize the order of the ensembles. For the pattern data set, randomize the order of the patterns.
2. Train and test MESO using all ensembles/patterns.
3. Calculate the classification accuracy by dividing the sum of all correct classifications by the total number of ensemble/patterns.
4. Repeat the preceding steps n times, and calculate the mean and standard deviation for the n iterations.

In our resubstitution tests, we set n equal to 100. Thus, for each mean and standard deviation calculated, MESO is trained and tested 100 times for both the pattern and ensemble data sets.

Table 2 summarizes the accuracies and timing results for the four birdsong data sets. Resubstitution is greater than 92% accurate for all data sets while leave-one-out results are somewhat less accurate. Given that bird vocalizations are highly variable and that data set sizes are relatively small, we can consider these results promising.

Table 2. MESO classification results.

Data set	Accuracy%/Time(s)
Pattern	
Leave-one-out	71.5%±0.9%
Resubstitution	92.3%±3.1%
Training	57.7±1.1
Testing	57.7±1.9
Ensemble	
Leave-one-out	76.0%±1.1%
Resubstitution	96.3%±2.8%
Training	56.1±1.7
Testing	58.6±2.8
PAA Pattern	
Leave-one-out	80.4%±0.3%
Resubstitution	94.7%±0.8%
Training	57.7±1.1
Testing	57.7±1.9
PAA Esnemble	
Leave-one-out	82.2%±0.9%
Resubstitution	97.2%±1.2%
Training	56.1±1.7
Testing	58.6±2.8

Table 3 shows the confusion matrix for classification using PAA ensembles and the leave-one-out approach. Matrix columns are labeled with the species predicted by MESO, while rows are labeled with the species that actually produced the original vocalization. The main diagonal (in bold) indicates the percentage of patterns correctly classified. Other cells indicate the percentage of patterns confused with other species. For instance, the intersection of the row labeled AMGO with the column labeled BLJA indicates that 0.5% of blue jay patterns were confused with the American goldfinch. As shown, most ensembles are correctly classified, with the red winged blackbird most likely to be classified correctly.

5 Related Work

Automated processing of data streams is a large and active field; we focus here on several closely related contributions; additional related work is discussed in [7, 8]. Several research projects address selection of tuples from data streams [17–19]. Such works treat a data stream as a database and optimize query processing for better efficiency. Our work with automated extraction of ensembles and annotation of data stream content complements these

Table 3. Confusion matrix using ensembles.

	Predicted									
	AMGO	BCCH	BLJA	DOWO	HOFI	MODO	NOCA	RWBL	TUTI	WBNU
AMGO	**70.3**	7.8	0.5	1.5	0.5	3.8	2.8	4.5	1.7	6.6
BCCH	5.2	**69.2**	4.3	2.5	4.4	0.1	2.6	3.7	2.9	5.2
BLJA	2.1	3.5	**86.0**	0.5		3.4	1.7	0.5	0.2	2.2
DOWO		5.5	0.5	**90.5**	1.1		0.1	0.1	2.2	
HOFI	2.9	1.2	2.3	3.9	**79.3**		6.6		3.7	0.2
MODO	7.6	1.6	1.8	3.7	4.1	**67.0**	6.4	3.1		4.7
NOCA	6.0	0.1	0.1		0.3	0.1	**90.8**	0.6		2.0
RWBL	0.9	0.5		2.8		0.5	0.5	**94.7**	0.2	
TUTI	2.2	2.6	0.7		2.1		1.1		**90.5**	0.9
WBNU	3.4	0.3	0.1	1.2		4.8	2.4	0.4	1.2	**86.1**

approaches. For example, annotations can be treated as tuples that describe the underlying data stream and can be used by selection schemes for routing data stream to address application specific requirements.

A number of research projects have addressed the construction of distributed stream processing engines (SPEs) that provide quality-of-service optimization or guarantees. Examples include Infopipes [20], Wavescope [21], Aurora [22], Medusa [23], Borealis [24] and CANS [25]. Although in general these systems provide functionality similar to that of Dynamic River, the latter's support for scoping while addressing graceful recomposition and fault resilience, can be considered its chief advantage. Pipelines composed for data acquisition and analysis of continuous sensor data streams must be able to resynchronize and enable the continuation of meaningful data stream processing in the face of pipeline recomposition and faults.

Recently, there has been increased interest on identifying motifs [13, 26] in time series. Motifs are defined as frequently occurring time series sequences. Identification of motifs requires analysis of a time series to determine which subsequences occur frequently. Motifs can be used for the construction of a model that represents the normal behavior of a time series. On the other hand, a discord [14] is defined as the sequence that is least similar to others. Our work with ensembles complements work on motifs and discords in that ensembles can be considered as candidate motifs or discords. However, rather than focus on the most or least frequent time series patterns, ensembles are locally anomalous patterns that may recur only rarely. Our focus is on the timely, automated processing of continuous streams of sensor data that likely comprise variable length events. As such, processor and memory efficient techniques for extracting and processing ensembles are needed. Our approach to ensemble extraction requires only a single scan of a time series and extracts variable length ensembles.

6 Conclusions

We have presented a technique for extracting ensembles from acoustic data streams with the goal of recognizing bird species in natural environments. Our Dynamic River prototype enables distributed data stream processing with support for resynchronization of data scope in the face of dynamic pipeline recomposition or pipeline segment failure. Results of our classification experiments show promise for automating species surveys using acoustics. Moreover, ensemble extraction and processing using distributed pipelines may enable timely annotation and clustering of sensor data streams. Currently, we have extracted ensembles from data streams comprising a single signal. Although acoustic data streams are data rich, extracting ensembles from multiple correlated data streams may enhance classification and detection of time series events. For instance, species identification may be more accurate when acoustic data is coupled with geographic, weather or other information about the environment. We plan to address this issue in a future study.

Further information. A number of related papers and technical reports of the Software Engineering and Network Systems Laboratory and the Computational Ecology and Visualization Laboratory can be found at the following URLs: `http://www.cse.msu.edu/sens` and `http://www.cevl.msu.edu`.

Acknowledgments. The authors would like to thank Ron Fox and Joo Wooyeong at Michigan State University for their contributions to this work. This work was supported in part by the U.S. Department of the Navy, Office of Naval Research under Grant No. N00014-01-1-0744, National Science Foundation grants EIA-0130724, and ITR-0313142 and a Quality Fund Concept grant from Michigan State University.

References

[1] J. Porter, P. Arzberger, H.-W. Braun, P. Bryant, S. Gage, T. Hansen, P. Hanson, C.-C. Lin, F.-P. Lin, T. Kratz, W. Michener, S. Shapiro, and T. Williams, "Wireless sensor networks for ecology," *Bioscience*, vol. 55, pp. 561–572, July 2005.

[2] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Proceedings of The Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, (Baltimore, Maryland, USA), November 2004.

[3] K. Martinez, J. K. Hart, and R. Ong, "Environmental sensor networks," *IEEE Computer*, vol. 37, pp. 50–56, August 2004.

[4] P. Arzberger, ed., *Sensors for Environmental Observatories*, (Seattle, Washington, USA), World Technology Evaluation

Center (WTEC) Inc., Baltimore, Maryland, December 2004. Report of the NSF sponsored workshop.

[5] "National ecological observatory network (NEON)." `http://www.neoninc.org`, November 2006.

[6] S. A. Isard and S. H. Gage, *Flow of Life in the Atmosphere: An airscape approach to understanding invasive organisms*. East Lansing, Michigan, USA: Michigan State University Press, 2001.

[7] E. P. Kasten and P. K. McKinley, "MESO: Supporting online decision making in autonomic computing systems," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 19, no. 4, pp. 485–499, 2007.

[8] E. P. Kasten, P. K. McKinley, and S. H. Gage, "Automated ensemble extraction and analysis of acoustic data streams," Tech. Rep. MSU-CSE-06-40, Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan, USA, December 2006. Available at http://www.cse.msu.edu/∼mckinley/acoustics.pdf.

[9] W. H. Thorpe, "The learning of song patterns by birds, with especial reference to the song of the chaffinch, Fingilla coelebs," *Ibis: The international journal of avian science*, vol. 100, pp. 535–570, 1958.

[10] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, (San Diego, California, USA), June 2003.

[11] B.-K. Yi, , and C. Faloutsos, "Fast time sequence indexing for arbitrary Lp norms," in *Proceedings of the 26th International Conference on Very Large Databases*, (Cairo, Egypt), September 2000.

[12] N. Kumar, N. Lolla, E. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Time-series bitmaps: A practical visualization tool for working with large time series databases," in *Proceedings of SIAM International Conference on Data Mining (SDM'05)*, (Newport Beach, California, USA), pp. 531–535, April 2005.

[13] J. Lin, E. Keogh, S. Lonardi, and P. Patel, "Finding motifs in time series," in *Proceedings of the 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (Edmonton, Alberta, Canada), July 2002.

[14] E. Keogh, J. Lin, and A. Fu, "HOT SAX: Finding the most unusual time series subsequence," in *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005)*, (Houston, Texas, USA), pp. 226–233, November 2005.

[15] S. Murthy, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *Journal of Artificial Intelligence Research (JAIR)*, vol. 2, pp. 1–32, 1994.

[16] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Boston, Massachusetts, USA: Pearson Education, Incorporated, 2006.

[17] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the 21st ACM Symposium on Principles of Database Systems (PODS)*, (Madison, Wisconsin, USA), June 2002.

[18] R. Avnur and J. M. Hellerstein, "Eddies: Continuously adaptive query processing," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, (Dallas, Texas, USA), May 2000.

[19] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. A. Shah, "TelegraphCQ: Continuous dataflow processing for an uncertain world," in *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR)*, (Asilomar, California, USA), January 2003.

[20] A. P. Black, J. Huang, R. Koster, J. Walpole, and C. Pu, "Infopipes: An abstraction for multimedia streaming," *Multimedia Systems (special issue on Multimedia Middleware)*, vol. 8, no. 5, pp. 406–419, 2002.

[21] L. Girod, K. Jamieson, Y. Mei, R. Newton, S. Rost, A. Thiagarajan, H. Balakrishnan, and S. Madden, "The case for a signal-oriented data stream management system," in *Proceedings of the Third Biennial Conference on Innovative Data Systems Research (CIDR)*, (Pacific Grove, California, USA), January 2007.

[22] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, , N. Tatbul, and S. Zdonik, "Aurora: A new model and architecture for data stream management," *VLDB Journal*, vol. 12, pp. 120–139, August 2003.

[23] S. Zdonik, U. Çetintemel, M. Stonebraker, M. Balazinska, M. Cherniack, and H. Balakrishnan, "The Aurora and Medusa projects," *IEEE Data Engineering Bulletin*, vol. 26, March 2003.

[24] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. S. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik, "The design of the Borealis stream processing engine," in *Proceedings of the Second Biennial Conference on Innovative Data Systems Research (CIDR)*, (Pacific Grove, California, USA), January 2005.

[25] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti, "CANS: Composable, adaptive network services infrastructure," in *The 3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, (San Francisco, California, USA), pp. 135–146, March 2001.

[26] B.-K. Yi, H. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Proceedings of the IEEE International Conference on Data Engineering*, (Orlando, Florida, USA), pp. 201–208, February 1998.