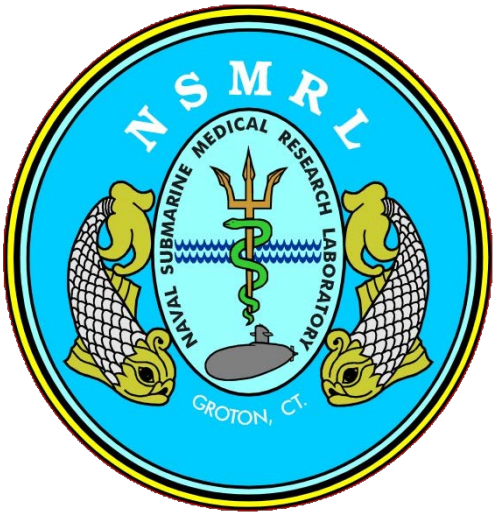
Naval Submarine Medical Research Laboratory

NSMRL/F1703/TR--2020-1340 July 28, 2020



Integration of a Computational Auditory Scene Analysis Software Architecture with the ARCADIA Cognitive Model: Software Design, Usage, Implementation, Demonstration

David Gever
Matthew Daley
Matthew Babina
Bohdan Snihurowych
Lia Bonacci
Jeffrey Bolkhovsky

Approved and Released by:
K. LEFEBVRE, CAPT, MSC, USN
Commanding Officer
NAVSUBMEDRESCHLAB

Approved for Public Release, Distribution is Unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
28-07-2020	Technical Report /Memorandum	17/4/2017 - 30/09/2019

4. TITLE AND SUBTITLE

Integration of a Computational Auditory Scene Analysis Software Architecture with the ARCADIA Cognitive Model: Software Design, Usage, Implementation, Demonstration

5a. CONTRACT NUMBER

N0001418WX00965

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Jeffrey Bolkhovsky, David Gever, Matthew Daley, Matthew Babina, Lia Bonacci, Bohdan Snihurowych

5d. PROJECT NUMBER

Z44G

5e. TASK NUMBER

5f. WORK UNIT NUMBER

F1703

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Submarine Medical Research Laboratory
Box 900
Groton, CT 06349-5900

8. PERFORMING ORGANIZATION REPORT NUMBER

NSMRL/F1703/TR--2020-1340

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
One Liberty Center
875 N. Randolph Street, Suite 1425
Arlington, VA 22203-1995

10. SPONSOR/MONITOR'S ACRONYM(S)

ONR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRIBUTION A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Naval Submarine Medical Research Laboratory (NSMRL) has developed an auditory model, which is a software application that attempts to emulate human hearing. It can feed its results to cognitive models that attempt to mimic the behavior of humans and are able to take advantage of auditory information as if it came from a human's ears. The software application addresses the "cocktail party problem," which occurs when humans are able to follow a particular conversation from within a diverse background of noise. The auditory model accomplishes this by implementing a series of signal processing steps that are inspired by human sensory processing, using Matlab® and Clojure programming languages.

15. SUBJECT TERMS

Computational Auditory Scene Analysis, Stream Segregation, Clustering Cochleagram, Gaussian Mixture Model, Gammatone Filter Bank, Matlab, Clojure

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			NAVSUBMEDRSCHLAB Commanding Officer
U	U	U	SAR	70	19b. TELEPHONE NUMBER *(Include area code)* 860-694-3263

Reset

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

[THIS PAGE INTENTIONALLY LEFT BLANK]

David Gever
Matthew Daley
Matthew Babina
Bohdan Snihurowych
Lia Bonacci
Jeffrey Bolkhovsky

Naval Submarine Medical Research Laboratory

Approved and Released by:

LEFEBVRE.KIM.LOREEN.1012602665 Digitally signed by LEFEBVRE.KIM.LOREEN.1012602665 Date: 2020.07.28 11:46:50 -04'00'

K. LEFEBVRE, CAPT, MSC, USN

Commanding Officer
Naval Submarine Medical Research Laboratory
Submarine Base New London Box 900
Groton, CT 06349-5900

Administrative Information:
The views expressed in this report are those of the author and do not necessarily reflect the official policy or position of the Department of the Navy, Department of Defense, nor the U.S. Government. This work was supported by and/or funded by work unit number F1703. I am a military service member (or employee of the U.S. Government). This work was prepared as part of my official duties. Title 17 U.S.C. §105 provides that 'Copyright protection under this title is not available for any work of the United States Government.

Approved for Public Release, Distribution is Unlimited.

[THIS PAGE INTENTIONALLY LEFT BLANK]

Abstract

The Naval Submarine Medical Research Laboratory (NSMRL) has developed an auditory model, which is a software application that attempts to emulate human hearing. Its results can be fed into cognitive models, which attempt to mimic human behavior by taking advantage of auditory information that human hearing obtains in the real world.

The software application addresses the "*cocktail party problem,*" as exemplified when humans are able to follow a particular conversation from within a diverse background of noise. The auditory model accomplishes this by implementing a series of signal processing algorithms that are inspired by the human physiology and psychoacoustics. The algorithms are based on literature from the field of **C**omputational **A**uditory **S**cene **A**nalysis (CASA). The CASA software model accepts stereo sound files and extracts several auditory features such as the azimuth from which the sounds came, specific bandwidths of frequencies, fundamental frequencies, as well as onsets and offsets of the signal arrivals. Features that appear to be correlated are then clustered, using a Gaussian Mixture Model (GMM), to yield separate auditory *"streams*". These are the ingredients that can be fed to a cognitive model that leverages auditory information in its processing. Although the CASA software model is intended to be agnostic to the cognitive architecture to which it is linked, the CASA software application is designed specifically to work with the cognitive architecture called ARCADIA ("**A**daptive **R**eflective **C**ognition in an **A**ttention **D**riven **I**ntegrated **A**rchitecture"), which has been developed by the Naval Research Laboratory (NRL).

This report describes our approach to developing and verifying a computerized auditory model that emulates how humans separate sounds into their individual components. The report also includes the results from the verification phase from a qualitative and quantitative stand point.

Executive Summary

Under the sponsorship of the Office of Naval Research (ONR), the Naval Submarine Medical Research Laboratory (NSMRL) has developed an auditory model, which is a software application that attempts to emulate human hearing. Its results can be fed into cognitive models, which attempt to mimic human behavior by taking advantage of auditory information that human hearing obtains in the real world.

The software application addresses the "*cocktail party problem,*" as exemplified when humans are able to follow a particular conversation from within a diverse background of noise. The auditory model accomplishes this by implementing a series of signal processing algorithms that are inspired by the human physiology and psychoacoustics. The algorithms are based on literature from the field of **C**omputational **A**uditory **S**cene **A**nalysis (CASA). The CASA software model accepts stereo sound files and generates a cochleagram by applying a Gammatone Filter Bank and a "hair cell model" to the entering audio signals. Several auditory features are then extracted from the cochleagram. Features such as the azimuth from which the sounds came, specific bandwidths of frequencies, fundamental frequencies, as well as onsets and offsets of the signal arrivals, are examples of what the software modules extract. Features that appear to be correlated are then clustered, using a Gaussian Mixture Model (GMM), to yield separate auditory *"streams*". These are the ingredients that can be fed to a cognitive model that leverages auditory information in its processing. Although the CASA software model is intended to be agnostic to the cognitive architecture to which it is linked, the CASA software application is designed specifically to work with the cognitive model called ARCADIA ("**A**daptive **R**eflective **C**ognition in an **A**ttention **D**riven **I**ntegrated **A**rchitecture"), which has been developed by the Naval Research Laboratory (NRL). The CASA software application currently ends with the stream segregation step, but in the future the code will classify the streams to yield "*objects*", which equate to the category of each segregated sound source. Additionally, in the future, the CASA front-end processing will accept "*top-down*" and "*bottom-up*" feedback from the cognitive architecture in order to tune its feature extraction logic to provide more relevant auditory features to the stream segregation operation.

Development of human inspired models requires a comprehensive understanding of how humans react to various stimuli, as well as an accurate implementation of that understanding. In order to accomplish this, a verification and validation (V&V) process needs to be completed. The first of these two steps ensures that the computer version of the human, acts as the software has been designed. The second step compares the computer results with human results. This report describes our approach to developing and verifying a computerized auditory model that emulates how humans separate sounds into their individual components, such that they could be fed to ARCADIA. The report also includes the results from the verification phase from a qualitative and quantitative stand point. Verification was accomplished by favorably comparing figures derived from the Clojure version of the software with those obtained from the MATLAB® version. Quantitative agreement through numeric comparisons of intermediate processing results were also used in the verification process.

[THIS PAGE INTENTIONALLY LEFT BLANK]

Table of Contents

Table of Figures

Acknowledgements

First, we would like to acknowledge that Dr. Paul Bello and Dr. Michael Qin initiated the project, through ONR, that resulted in this software application. Dr. Frank Ritter, reviewed and made valuable and significant additions and corrections to improve this document. Internal and external reviewers also enhanced the quality of this report. ONR (Contract Number: N0001418WX00965) provided funding for this project.

1.0 Introduction

The United States (U.S.) Department of Defense (DoD) has identified the modernization of key capabilities as a requirement to maintain and strengthen our advantage over our adversaries and competitors (DoD National Defense Strategy, 2018). As the military develops new technologies and functionalities that will be introduced into operational and combat environments, it is imperative to understand how humans interact with these new systems and how to design effective, human-centered interfaces. The Office of Naval Research (ONR), the Undersea Warfare Chief Technology Office, and the DoD all list research into, and improvement in, human-machine interfaces as primary objectives for their research and development strategies (ONR, UWCTO, 2016, DoD, 2019).

One way that researchers investigate how humans interact with machines is with human-based behavioral experiments. While effective, this approach can be time consuming and costly due to the large number of subjects and test scenarios required to gather enough data that is sufficient to perform meaningful statistical analyses (NIBIB/NIH Computational Modeling Fact Sheet). Thus, the field as a whole, and the military in particular, are increasingly using cognitive models to emulate human response to various stimuli and machine interfaces. Cognitive models are computerized representations of human problem-solving and mental processing, such that, if verified and validated to simulate human behavior, represent a significant savings in time and money. Furthermore, if accurate, models can be used to predict human behavior in response to stimuli in both the research community as well as in commercial and military settings.

One cognitive process of interest to the military involves the human auditory system. The military has identified a need to better understand how humans identify and respond to auditory stimuli, and how the auditory system works with the visual system to improve perception (ONR, Division 342). One of the most remarkable features of human hearing is the ability to focus on a specific noise or sound source from a multi-talker background. This phenomenon solves, what is often referred to as the "cocktail party problem" (Cherry, 1953). It is complex because it is primarily concerned with parsing out and focusing on conversations of interest in a complex auditory scene, and relies on the ability of humans to segregate multiple, simultaneously received sounds into individual "*streams.*" Humans can perform this task with ease, even while the underlying mechanisms for how they accomplish it are not well understood. Within the military setting, recognizing a commander's instructions within the context of a noisy operational environment, or isolating and identifying a specific alarm/sound during a casualty to determine a course of action, are examples of how this ability is important for mission success.

An important feature of the auditory system is its link to the visual system in that it can provide spatial information and situational awareness that complements that which is available from the visual system alone. Conversely, the visual system can augment the auditory system with its own cues (Bregman, 1990). For example, from the auditory standpoint, an individual might hear a noise and turn in the direction of the sound to identify it or gain more information visually. This is an example of using "*bottom-up*" auditory saliency to cue the visual attention in the direction of the sound. Alternatively, a "*top-down*" approach might be invoked, whereby a human may be goal oriented to listen for a specific kind of sound in order to direct his/her vision toward it when heard (Bregman, 1990). These two

approaches and the ability to separate and identify particular sounds and their sources, and react quickly and accurately, can be critical to military performance in operational settings that impact mission success and safety of service members (ONR: Division 342).

In recognition of the need to better understand how humans identify and respond to auditory stimuli and how this ability improves audio-visual perception, ONR has funded the Naval Submarine Medical Research Laboratory (NSMRL) to develop an auditory model that simulates a human's ability to interpret sounds (localization, binaural separation, speech recognition), and to integrate this auditory model with other sensory models (such as those based on the human visual system) to mimic human behavior. NSMRL has developed such an auditory model, based on **C**omputational **A**uditory **S**cene **A**nalysis (CASA) methodology that emulates how humans react to auditory stimuli. This document describes the CASA-based software that was built to provide auditory inputs to a cognitive model called ARCADIA ("**A**daptive **R**eflective **C**ognition in an **A**ttention **D**riven **I**ntegrated **A**rchitecture"), (Bridewell, Bello, 2016).

ARCADIA is a general software framework for modeling attention's role in cognition and action. It is based on the premise that human behavior requires a person to direct their attention to a stimulus before a response will occur. To demonstrate this, with the introduction of audio cues, such as directionality, amplitude variations, and/or frequency content, the ARCADIA software can emulate the way a person's eyes might react to auditory stimuli, once the person's attention has been focused on the stimulus. As noted above, this is an example of a "bottom-up" reaction. Because the visual system is closely linked with the auditory system, the presentation of a sound may stimulate the visual system (eyes) to turn in the direction of the sound (Braga et al, 2016). NSMRL's CASA-based auditory model enhances ARCADIA by providing the auditory information to which ARCADIA can respond. In this way, the CASA-based software application acts as a "*front-end*" to the ARCADIA cognitive model by feeding auditory "streams" or "objects," which then cause ARCADIA's visual attention to shift in response to these auditory cues.

Because human testing is expensive and time consuming, there is value in integrating audio and visual information together, into a single model, based on human test results. Such models may prove beneficial in our understanding of how humans' ears, eyes, and brain function together, which may lead to performance improvement in many environments, including military situations. , at a much lower cost than required with human testing. This is because multiple instances of a model can be executed very quickly, and models can be modified easily in order to more closely reflect human test results. The intention of this effort was for this software application to act as a "front-end" and feed auditory inputs to the ARCADIA cognitive model, thus extending the ARCADIA architecture to become more comprehensive in its modeling of human behavior, which in turn might contribute to a better understanding and better performance of human responses to audio and visual stimuli.

1.1 Human Hearing & Cognitive Models

Human ears process sound by passing entering sound waves from the outer ear, to the middle ear, past the ear drum to the inner ear, to the cochlea, and then to the brain via the cochlear nerve and central auditory system, terminating at the auditory cortex. The anatomical view of this is depicted in **Figure 1**. Traditionally, cognitive models act on predetermined and simulated auditory and visual objects as their

environmental stimuli and provide simulated human behavior in response to those stimuli (Bregman, 1990). For example, a model auditory environment may consist of a set of tones that have a certain onset, duration, periodicity, and, if stereo, a direction and amplitude difference between the two arriving sounds. As an output, these models can predict how and to what stimuli a human will respond, providing insight into the types of scenarios to avoid or ways to enhance or mitigate certain behaviors. However, humans are imperfect listeners, and can make mistakes in their interpretation of sounds, such as the transmit direction, number of sources, and/or type of sound source (Yost, 2000). To account for errors in human perception, traditional models' auditory signals are perturbed in some manner. This is typically achieved by adding *noise* to the signals' onset times, periodicity, and direction. The cognitive model then responds in a stochastic manner to these "noisy" signals in order to simulate the errors made by humans. This approach, however, is inconsistent with how humans actually hear and interpret sounds because humans do not simply add noise to the sounds that arrive at their ears, thus limiting the applicability of these models.

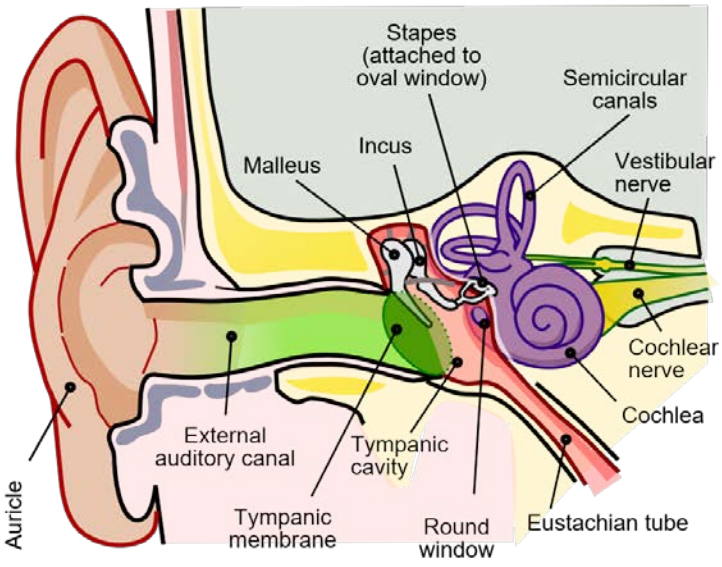


Figure 1. Path taken by sound in a human ear. Sound passes through the ear canal, past the ear drum, into the cochlea, which stimulates hair cells that consequently activate the auditory nerve which sends information to the auditory cortex in the brain. The cochlea acts as a spectrum analyzer, and the brain acts as the interpreter. (Unlicensed content: Perception Space—The Final Frontier, A PLoS Biology Vol. 3, No. 4, e137 doi:10.1371/journal.pbio.0030137, Fig. 1A)

NSMRL took a different approach to generating the auditory stimuli which will feed a cognitive model. Our approach is based on implementing computer code that adheres to physiologic and psychoacoustic principles of the human auditory system, as found in the literature (Wang, Brown, 2006), as opposed to simply adding noise, that has a Gaussian distribution for example, to the input data, to achieve the statistical uncertainty associated with correctly interpreting the sound. The human ear differentiates multiple incoming sounds via pitch and frequency-specific hair cells in the cochlea (Yost, 2000) (this is described in more detail in **Section 2.1**). For our CASA model, we implemented a set of computer algorithms that can separate a mixture of sounds into its individual components, referred to as *"streams"* in the field of Computational Auditory Scene Analysis (CASA). These "*streams*" are composed

of acoustic properties, or "*features*", that can be grouped together because they share common acoustic properties. This approach more closely emulates the way that scientists believe a human interprets sounds (Wang, Brown 2006). Our model does this by invoking a series of digital signal processing steps on the incoming auditory signals, which are recorded from two microphones and fed into the model in the form of stereo WAV files. Eventually these "*streams*" are *classified*, and considered "*objects*" or "sources" of the sounds (Bregman, 1990). However, our processing currently ends when the CASA based software model has separated a mixture of sounds into what is perceived to be the individual "*streams*". **Figure 2** illustrates the difference between the traditional "additive noise" and NSMRL's CASA approach.

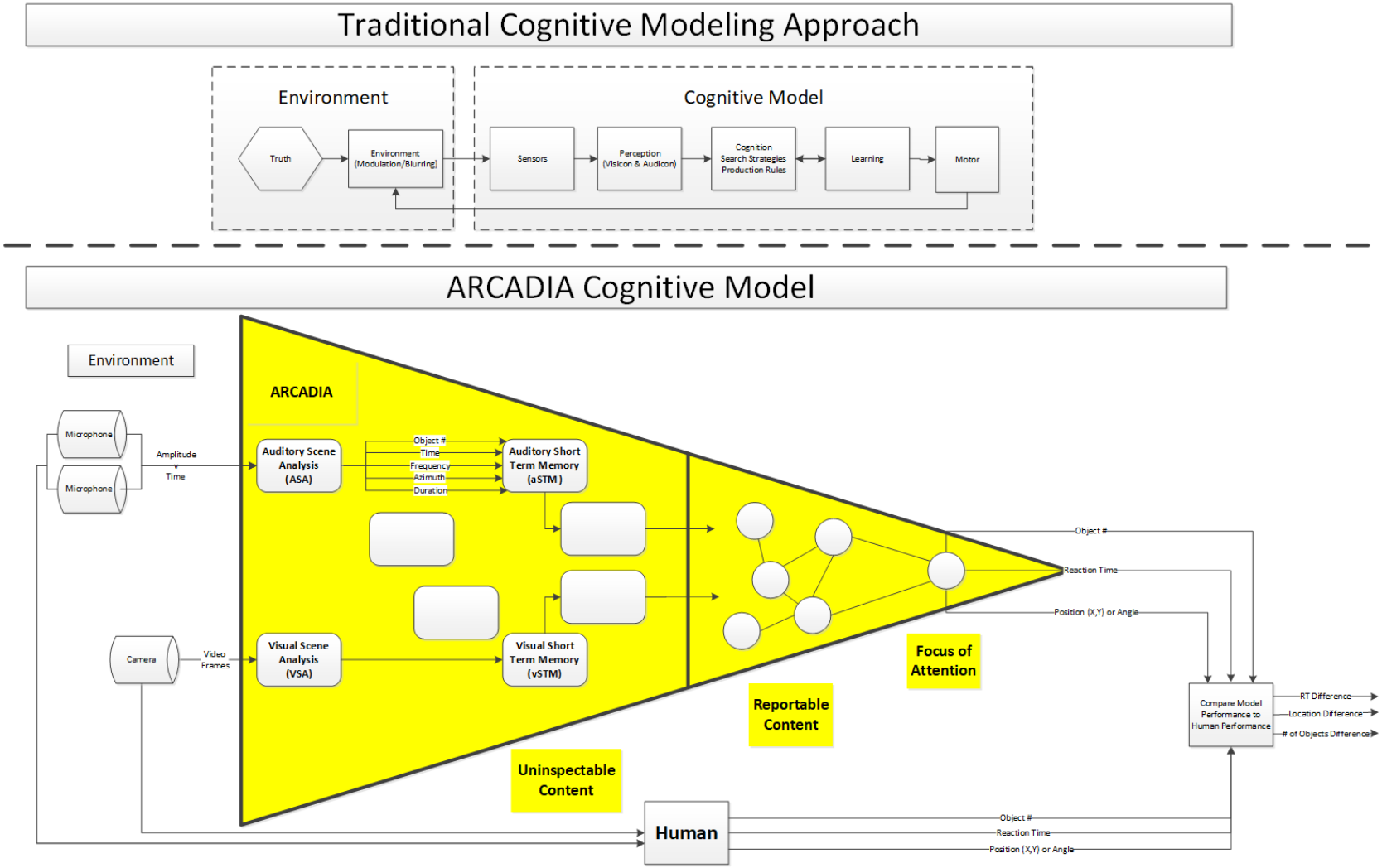


*Figure 2. Traditional vs CASA approach to providing auditory stimuli to a cognitive architecture, in this case "*ARCADIA*". Traditionally, noise is added to the input sound signals to introduce uncertainty prior to entering a cognitive model. Errors in human interpretation of those "noisy" sounds is then statistically distributed. The alternative CASA approach ingests the stereo sounds and extracts acoustic "features". "Features" that have common attributes are grouped together or "segregated" into "streams" that are fed to the ARCADIA cognitive model's "aSTM". The rectangular boxes represent signal processing "components" that act on the content to yield ARCADIA "objects". The circles depict "objects" as they become elevated to the "Focus of Attention".*

Integration of the audio data into the ARCADIA cognitive model is accomplished by passing the segregated "*streams*" or classified "*objects*" into the model. If an audio object is salient enough to garner the attention of ARCADIA's signal processing attention components, and the object has been attended to (perceived and features recognized), the model may direct the response of its sensors, eyes in this case, in the direction from which the sound came, based on the ability to discern direction from differential arrivals of acoustic signals. This may also cause its motor functions to react to the auditory stimulus. For example, imagine a person hears the sound of a ball being struck by a baseball bat, turns toward home plate to see where the ball is travelling, and potentially raises his/her hands to protect him/herself from being hit by the ball. In this way, the model should reflect the way in which humans respond to important auditory stimuli. Again, this activity is depicted in the yellow portions of **Figure 2** and in the function boxes at the right-most portion of the figure. Together with our application, acting as a "front-end" to a separate cognitive architecture, is an attempt to model the processing of the auditory stimulus, necessary to provide the visual and subsequent psychomotor response that follows.

The algorithms implemented within the CASA auditory model are not specific to any singular cognitive model. The CASA auditory model architecture may also be applicable as a *front-end* to the processing that may be running inside Unmanned Underwater Vehicles (UUVs), and/or robots, since some of them will perform their tasks based on visual inputs and cues provided by audio sensor data. However, this document describes the software that was specifically coded to integrate with the ARCADIA cognitive architecture, which was built at the Naval Research Laboratory (NRL), and its implementation in the Clojure programming language[1]. Therefore, our CASA model was built upon the Clojure computer language as well. In order to verify that the Clojure code worked as designed, the results were compared to a MATLAB® version of the algorithms in a qualitative and quantitative manner.

The document includes background information that should provide a better understanding of how our software fits into the field of CASA. With this background, one should be better poised to follow the software design rationale as described in **Section 2. Sections 3** and **4** contain the Results and Conclusions, respectively, of the Clojure based application. The **Appendices** of this document are mainly concerned with the implementation of the application, the usage of the software, and intermediate results that verify its functionality.

The current CASA-based software stops at the point of "*stream segregation*" because we had released a verified version of the CASA based software to NRL, so that they could learn how to integrate the auditory information produced up to this point. However, in the near future, this software will be extended to *classify* the type of acoustic sources into "*objects*" or categories of sound (e.g., tone, music, event, speech), likely through the use of a Support Vector Machine (SVM). We also plan to use both "*top-down*", or goal-based, and "*bottom-up*", or saliency-based, information provided by the attention-based ARCADIA model, as feedback to the feature extraction logic of the CASA front-end processor. In this way, the CASA algorithms will be able to provide more relevant data to the feature space upon

[1] *Clojure is a member of the Lisp family of programming languages. Clojure is a recent dialect of Lisp that is designed to be a pragmatic general-purpose language that places a strong emphasis on immutability and provides access to Java frameworks and libraries.*

which the cognitive model has directed its attention. In so doing, visual performance is expected to be improved by enhancing relevant auditory features while attenuating alternate features to be ignored, resulting in more focused auditory cues being fed to the cognitive architecture.

As an initial example, however, this particular CASA model is designed to provide information to the ARCADIA cognitive architecture in order to cue its attention-based logic so that it focuses its “eyes” to look in the direction of a particular sound. In this way, the combined model will more closely reflect the manner in which humans use both auditory and visual systems to interact with everyday settings. **Figure 3** depicts a high-level view of the software design that should accomplish this.

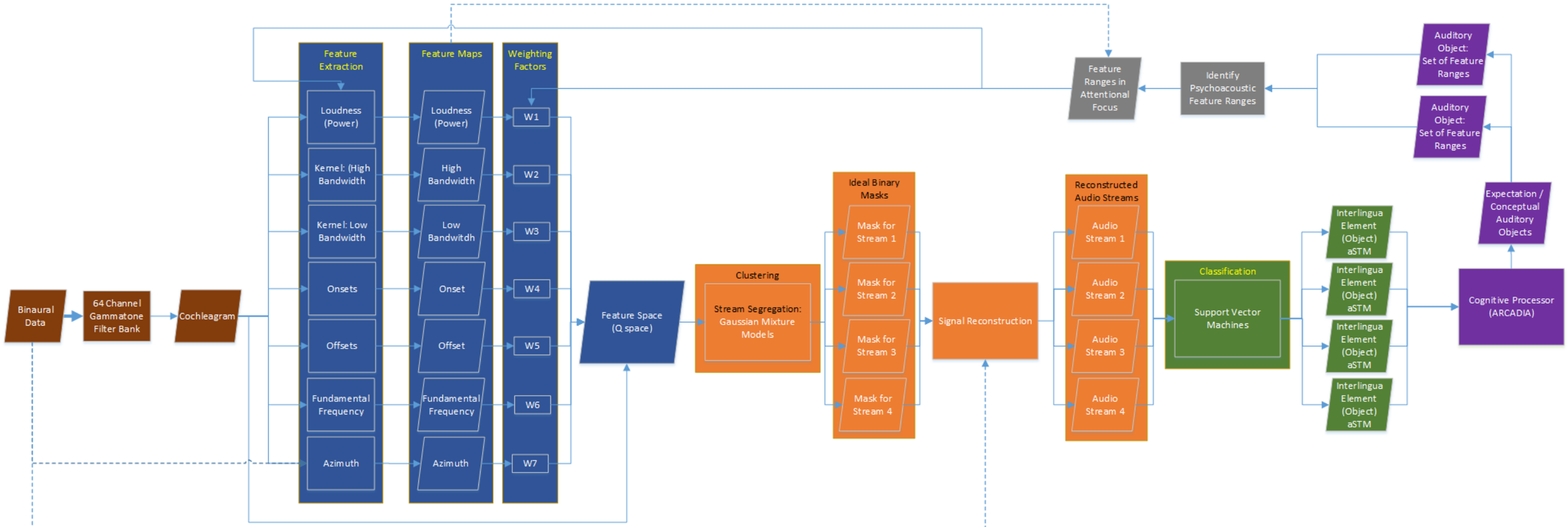


Figure 3. High-level auditory "front-end" CASA software design and its integration with the ARCADIA cognitive architecture. Spectral Decomposition (brown), Feature Extraction (blue), Stream Segregation (orange), Classification (green), Cognitive Model (purple), feedback functions (gray). Starting with a stereo audio sound file, features are extracted and fed to a clustering algorithm, which separates the sounds into streams. These streams will then be classified and fed to an attention-based cognitive architecture. Information from the cognitive architecture (gray) can be fed back to the feature extraction logic in a "top-down" and/or "bottom-up manner.

2.0 Software Design (Processing Steps)

The goal of the CASA software application is to identify the source of all sounds that arrive at both ears, so that one of the sound sources can be *attended-to* and *acted-upon* by a cognitive model. This involves isolating the sound to which attention will be drawn. This software application is designed to provide auditory-based cues to the "eyes" of the attention-based ARCADIA's cognitive architecture, such that it looks in the direction of a sound. To accomplish this, the CASA model extracts auditory *features* from a cochleagram and raw time series, as a precursor to grouping the features into auditory *streams*[2]. In other words, as a "*front-end*" to the ARCADIA cognitive model, the software provides *features* that are segregated into "*streams*" that contain common spectral and temporal attributes. The current CASA application stops with the stream segregation step. However, in the future, these streams will be classified to yield "*objects*", which are synonymous with the type of source that produced the sound, and these will then be fed to the ARCADIA architecture.

Our CASA model emulates the human auditory system by breaking up the signal processing into four main steps. This modularization of the software was considered to be appropriate because each of the four processing operations is loosely coupled. These steps are illustrated by the coloration of the shapes in **Figure 3** and in the list below. The first three of these steps were inspired by the work done by Elhilali & Shamma in 2008 concerning our approach to spectral decomposition/cochlear response, feature extraction, and stream segregation. These steps will be described below in detail. The fourth step (object classification) is in progress, and will require more time to complete. Because we have released a verified version of the CASA based software to NRL, so that they can start to integrate the auditory information produced up to this point, we decided to publish our interim results now.

1. **Spectral Decomposition and Cochlear Response:** (Gammatone Filter Bank & Hair Cell Model): frequency separation and power estimation as a function of time (brown shapes)
2. **Acoustic Feature Extraction**: amplitude, directionality, signal onsets and offsets, duration, periodicity/rhythm, fundamental frequency, bandwidths, pitch and harmonics (blue shapes)
3. **Stream Segregation:** correlation among the multiple feature dimensions. For example, smooth changes in amplitude and direction as a function of time, smooth changes in amplitude (AM) and frequency (FM) as a function of time, or abrupt changes in amplitude (onset/offset) as a function of time (orange shapes).
4. **Object Classification:** tone(s), music, event(s), human speech (green shapes) (*Not yet completed)*

This section describes the steps designed to act as the "front-end" auditory signal processing. Specifically, **Section 2.1** addresses the cochleagram generation step, i.e. visual representation of cochlear response. **Section 2.2** discusses general categories of auditory features that could be extracted. **Table 1** captures the general feature categories, examples and/or attributes, how they may be measured, and whether our software implemented the algorithms to do so. **Section 2.3** describes the auditory feature extraction operations that were actually invoked by the software.

[2] A "stream" in this context is analogous to a single voice in a crowded room.

Section 2.4 describes how the features are clustered into segregated auditory streams. **Section 2.5** describes the future capability of classifying the objects that were responsible for making the received sound.

Table 1: Auditory features based on the following categories; Examples and/or Attribute of the sound sources; Measurements used to extract the features; Whether the measurements have been Implemented in the CASA software application.

Feature Category	**Example or Attribute**	**Extraction Measurement**	Implemented in CASA Software?
Temporal	Sound Direction Sound Direction Rhythm	Interaural Time Difference (ITD) Interaural Level Difference (ILD) Periodicity	Yes No No
Spectral	Tone Musical Instrument Movement	Fundamental Frequency Bandwidths Doppler Shift	Yes Yes No
Loudness	Sound Level Distance	Received Power Received Power	Yes Yes
Event	Explosion Loud Clap	Onset Time Offset Time	Yes Yes
Complex	Human Speech Orchestra	Pitch, Formants Resonance, Timbre	No No

2.1 Cochlear Response (Spectral Decomposition)

The CASA model's first signal processing step mimics the behavior of the human cochlea in response to sound. The cochlea acts as a spectrum analyzer by separating the arriving sound into its frequency components which are converted into electrical signals and sent along the auditory nerve to the brain. It accomplishes this by vibrating hair cells, which are connected to the basilar membrane. The hair cells along the basilar membrane are in turn connected to neurons that send electrical signals (fire) as the hair cells oscillate at certain frequencies in response to the acoustic pressure waves. The portion of the basilar membrane that is closest to the ear drum, i.e., the first to receive the entering auditory signals, is most sensitive to the high frequencies, because the basilar membrane is stiffer along this portion of the cochlea. Conversely, the inner most part of the cochlear coil responds most actively (tuned) to the lower frequencies, because the basilar membrane is more flexible at this portion of the cochlea. The signal is thus separated into its frequency components (high to low), and the electrical signals, produced by the variable vibration rate, are then sent along the auditory nerve to the brain.

From a signal processing standpoint, there are several ways to separate a sound signal into its frequency components. Instead of using *a Discrete Fourier Transform (DFT)* on the time series to yield a *Power Spectral Density (PSD)* curve, we chose to apply a Gammatone Filter Bank to a set of overlapping frequency bands, called Equivalent Rectangular Bandwidth (ERB) channels, to more closely mimic the cochlear response to received sound energy (Lyon, 1983, Lyon, 2017). From a signal processing perspective, this can be accomplished by filtering the time series with multiple bandpass filters of

varying bandwidth that overlap one another. The suite of bandpass filters is known as a Gammatone Filter Bank. Each bandpass filter is applied to each ERB channel. The ERBs are centered at discrete frequencies, but are not equally spaced. Instead, their bandwidths increase as the center frequencies increase. The ERB channels are spaced in such a way that the center frequency and bandwidth correspond to the way scientists believe that the cochlea responds (Wang & Brown, 2006). The channels are spaced according to the following formulas (*Wang & Brown, 2006, Chapter 1, page 16*):

$$ERB(f_c) = 21.4 \log_{10}(0.00437 f_c + 1) \qquad \text{Eq. (1)}$$

$$bw(f_c) = 1.019\, ERB(f_c) \qquad \text{Eq. (2)}$$

Additionally, spectral decomposition via Gammatone Filtering was accomplished in the time domain, which is how scientists believe is the means that humans process sound energy that enters the cochlea, where the sound energy includes: amplitude, frequency, phase, and duration (Patterson et al, 1987). Once the filtering had occurred for each ERB channel, the resultant signal was half-wave rectified (Lyon, 1983). Each filtered ERB channel was then sliced in time at a period of 20 milliseconds. Finally, the received power within each ERB channel, and at each time slice was computed, yielding a cochleagram.

In our processing, 64 ERB channels were chosen, in order to better replicate the fidelity needed to more accurately hear sounds in difficult listening situations (Shannon et al, 2004). This number of ERB channels was also selected because it is a power of two, which has some signal processing advantages, and it provided enough frequency resolution to apply the processing algorithms in a time span that was short enough to complete multiple test runs per day. ERB center frequencies considered ranged from 40 Hz to 16000 Hz.

In order to demonstrate that the signal processing performed as we expected, we started the processing with a binaural time series that could be received by a human. This time series consisted of recordings from two microphones (stereo) that were separated by about 8 inches. The signals could be as simple as a continuous sine wave, or as complex as human speech or instrumental music. For our demonstration, however, we chose a 10-second WAV file that contained a non-overlapping constant tone, followed by some notes played by a saxophone. The WAV file was sampled at 44100 Hz.

2.2 Notional Feature Extraction

As noted above, a cochleagram depicts the power received at various frequencies that might be detected by a human ear as a function of time. It is the basis for most of the processing steps used to extract specific auditory features from all of the acoustic information encountered (Elhilali & Shamma, 2008). Below, we discuss the second major step in our processing, where our CASA software application extracts features. **Section 2.2** presents some of the general categories of auditory features that could be extracted (Wang & Brown, 2006), while **Section 2.3** describes the auditory features that our CASA software application actually extracts.

2.2.1 Temporal Based Features

In our implementation, the cochleagram is a 2D array that consists of about 10 seconds of filtered time samples belonging to 64 ERB channels, as opposed to a traditional spectrogram. However, in order to

process incoming sounds continuously, as human listeners do, the time series must be broken up into smaller segments. In our model, feature extraction is performed on the cochleagram by iteratively processing small (20 millisecond) windows of acoustic time series (aka snapshots or time slices), overlapped by 50%, and concatenating the results. In this way, as more sounds arrive and are sent through the processing stream, a more complete picture develops that captures the nature of the sounds over time. As the time history is built, temporal acoustic features such as the Interaural Time Difference (ITD) and Interaural Level Difference (ILD) can be extracted. These two features can be interpreted to represent sounds emanating from different azimuths. Features such as the "*onset*" of certain sounds can be discerned as well by finding those instants in time where the amplitude of the received sound is higher than the preceding time history. Once the onset of a sound is identified, it is possible to follow the higher amplitude, and consistent frequencies associated with the onset, until the sound stops, resulting in an "*offset*." From this, the signal's duration can be determined.

A regular pattern of onsets and offsets produces a periodic effect. If enough time history is available, the periodicity of a sound can be determined from a cochleagram. Likewise, if the periodicity is regular enough, a pattern (interpreted as rhythm) can also be discerned.

2.2.2 Frequency Based Features

A sound's fundamental frequency and its harmonics, which often correspond to the percept of pitch, can also be extracted as a feature from cochleagrams (Elhilali et al, 2008). That is, if enough time samples are available, modulation of the fundamental frequency (**FM**), and its harmonics, can be attributed to a particular sound source.

2.2.3 Amplitude Based Features

Modulation in amplitude (**AM**) can also be used to match a feature to a particular sound source (Elhilali et al, 2008). The amplitudes within a cochleagram can often be used to identify regions of frequency and time that contain energy that rise above the background ambient noise levels, thus attributing loudness to a particular source.

2.2.4 Events

Events can cross all three previously mentioned domains. They are often very loud, contain contrasting frequency content from what has been historically heard, and are short in duration. Onset and offset are important features that are often used to classify these kinds of sound signals (Elhilali et al, 2008).

2.2.5 Complex (Human Speech)

The patterns associated with human speech are generally more complex than those associated with either tones, music, or events, and therefore are more difficult to segregate and classify. The variability in frequency as a function of time can include harmonics that change quickly with time, as well as variable onsets, offsets (duration), and amplitude (Elhilali et al, 2008).

2.3 Implemented Feature Extraction Functionality

The CASA software application attempts to model the cochlea in its ability to separate sound waves that enter the ears as amplitude versus time, into a 3D surface of frequency versus time versus amplitude

known as a cochleagram. The feature extraction functions of our CASA model operate by searching for coherence in the following subset of notional cochleagram attributes listed above:

1. Loudness
2. Direction
3. Onset and Offset times (Periodicity/Rhythm)
4. Spectral Shape

2.3.1 Amplitude Coherence (Loudness)

Variability in levels of amplitude across frequency bands and time slices are typically relegated to different sources, whereas similar levels of amplitude often indicate a single sound source (Wang & Brown, 2006). Levels of loudness correspond to the amplitude of the received sound levels that are a product of the Gammatone filtering process (Patterson et al, 1987). Because the basic ingredient of a cochleagram is the output of the Gammatone Filter Bank, the filtered time series at each ERB channel was used to derive perturbations in the loudness of the sound at each ERB channel as a function of time, in order to discriminate one stream from another.

2.3.2 Azimuthal Coherence (Directionality)

Location can be used to collapse features into streams (Lyon, 1983). Sounds from a single source generally come from a single direction (reverberation notwithstanding). A capability that is inherent in human hearing, and that we attempted to emulate in the model, was the ability to discern the azimuth from which sound comes. It is thought that the arrival time difference (Interaural Time Difference or ITD) and the amplitude difference (Interaural Level Difference or ILD) of a sound that arrives at the two ears contribute to our ability to localize a sound source (Wang & Brown, 2006). For example, a sound arriving from the right would arrive at the right ear prior to arriving at the left ear, because of the extra distance needed for the sound to arrive at the left ear. Likewise, the level of the sound should be slightly louder at the right ear than the left because sound loses energy as it travels away from its source. The success in localizing (azimuth and elevation) of a sound depends on the size and shape of a person's head, and the size and shape of a person's pinnae (the external part of the ear). To improve azimuthal resolution, and from a signal processing perspective, a Head-Related Transfer Function (**HRTF**) can be designed and used to emulate a human's ability to discern directionality of incoming sounds (Pillow, 2017).[3] The current software employed a more simplistic approach to determining the azimuth (localization) of a sound source. It accomplished this by cross-correlating the stereo time series received by each ear. As part of this localization in azimuth processing of modeling human hearing, the Gammatone-filtered time series from each ERB channel was broken up into 20 millisecond chunks (windows or snapshots), that were sampled at 44100 Hz. Each window from the overall time series was cross-correlated between the two channels. The time lag associated with the peak in the correlation was captured and represented the ITD for each time window. These time lags were then concatenated across ERB channels and time. The result of such

[3] *The current CASA software application does not include the HRTF capability.*

processing was a frequency versus time map, where the Z-axis contained the time lag associated with the cross-correlation peak of each time window.

Although the software stopped at the point of recording the time lags, one could convert a time lag to an azimuth, assuming one knew the distance between the two ears (aperture or bitragon width) and the speed of sound of the medium in which the sound traveled (Woodworth et al, 1954). The most simplistic formula assumes a transparent head, such that:

$$\text{Azimuth} = \text{ArcCos}((\text{ITD} * \text{Sound Speed}) / \text{Aperture}) \quad \text{Eq. (3)}[4]$$

However, sounds that arrive from very close to either directly in front or behind a person's head will be nearly coincident in time and loudness. Therefore, the azimuth that will result from the calculation above is ambiguous and therefore likely to be confusing. Similar to this "front-back confusion in humans, the model too will yield ambiguous directionality. However, the CASA "front-end" processing only produces the time lag that is then fed to the down-stream cognitive architecture for conversion to and interpretation of an azimuth.

Additionally, as sound travels, it loses energy. Therefore, one should be able to detect the direction from which a sound came if the same sound is received from two separate sensors. Because human ears are separated by about 8 inches, the ear closer to the sound source should be louder. This difference in amplitude between the two ears is the Interaural Level Difference (ILD). The ILD can be computed similarly to the ITD calculation described above by comparing the average amplitude between the two channels. Whichever time series yields the higher amplitude is assumed to be the one nearer to the hemisphere in which the source of the sound originated. If both amplitudes are nearly equal, then this indicates that the sound source could have emanated from an ambiguous direction, that is, from either directly in front or in back of the person[5].

2.3.3 Temporal Coherence (Onsets and Offsets)

The temporal features cited here generally need to conform to a regular periodic pattern or vary smoothly in time in order to qualify as a stream (Bregman, 1990; Wang & Brown, 2006). Sounds that start and stop at the same time generally belong to the same stream (Bregman, 1990; Wang & Brown, 2006). Experiments have shown that a sequence of tones that occur regularly can be interpreted as either coming from a single source or separate sources, depending on factors such as the coincidence of sound inception and termination, the difference between the frequency of the tones, and the rate at which they arrive (Bregman, 1990; Wang & Brown, 2006). In either case, there is a consistent relationship between the temporal nature of the tones and their frequencies and/or the period in which they are heard (Bregman, 1990; Wang & Brown, 2006). The

[4] *The software does not include this computation to capture this feature as an angle. So, for efficiency reasons, time is saved by only calculating the time lag (ITD) between the two audio channels.*

[5] *This feature has not yet been included in the current version of the feature extraction software, but may be evaluated in the future as an additional feature to be fed to the stream separation function.*

CASA software identified onsets and offsets from quick increases and decreases, respectively, in cochleagram power values within each ERB channel.

2.3.4 Spectral Coherence (Spectral Shape)

If the frequency of the sound changes smoothly over time, or not at all, one may interpret that the sound comes from a single source (Bregman et al, 1973). However, if there is an abrupt change in frequency, then it is likely that there are two separate sources rather than a frequency fluctuation within the same source (Bregman et al, 1973). Spectral coherence, represented by measurements of spectral shape, are also thought to quantify difficult to articulate qualities of sound (Chi et al, 2005; Elhilali & Shamma, 2008; Krishnan et al, 2014). To expose these qualities, the CASA software created "scale filters", at two separate bandwidths; one narrow and one wide. These were then convolved with the frequency dimension of the cochleagram at each time slice. An amplitude threshold was then applied to both resulting frequency v time surfaces to expose a narrow and wide view of the cochlear response to the incoming sound.

2.4 Stream Segregation (Cluster Analysis)

After feature extraction, the goal of CASA is to segregate and then identify the source that is responsible for the sound (Bregman, 1990; Wang & Brown, 2006). As indicated above, the processing sequence used to determine this, first involved a spectral decomposition and cochleagram generation. The second step extracted auditory features *(such as: azimuth, onsets & offsets, power, bandwidths)* from the entering sound signals and cochleagram. The third step is to find those features that exhibit common attributes which can be used to group them into *streams or clusters. (The two terms, streams and clusters are used interchangeably).* The last step includes classifying these streams into a given category, also known as "*objects*". Example objects are: speech, horn, drum, siren, etc. (*The current software version does not yet perform this step*).

Because it is thought that humans group individual acoustic attributes that share some commonality, into more general entities, the next step in processing identified cohesive clusters ("streams") within the feature space. A Gaussian Mixture Model (GMM) was chosen as the approach for identifying these cohesive clusters. (Grira et al, 2005). The GMM was selected because it provided a means for assigning individual time and frequency samples for each feature, as taken from the cochleagram, based on thresholding the probability that they belonged to a particular cluster, as opposed to a cluster assignment that was based on an either/or approach, as used by a "K-Means" algorithm, for example. Additionally, GMMs were chosen because the clusters themselves are believed to be composed of features that exhibit Gaussian distributions (Reynolds et al, 1995).

As discussed above, the features can be grouped into four main categories that have common attributes in: amplitude, time, frequency, and azimuth. However, the main factors that are used to determine whether a feature, or set of features, belongs to a stream are time and frequency (Bregman, 1990; Wang & Brown, 2006). If enough time has been analyzed, then temporally correlated features are more likely to be identified, and if stable enough in amplitude and/or frequency, over time, they can also be classified as *streams*. The CASA model included 7 auditory "features" in the GMM stream segregation processing (clustering): power, azimuth, 2 bandwidths, onset, offset, and fundamental frequency.

In the current CASA model, 3 streams were created, where each stream was represented by a 2D matrix that was the same size as the cochleagram. Each cell from each 2D time/frequency "cluster" matrix contained the probability that all 7 features belonged to that particular "cluster" (or stream). The Mahalanobis Distance (*Appendix C.4.3*), between the probability of the features at each time and frequency observation, and a cluster's mean, was compared to a threshold to decide to which cluster (stream) a particular feature was assigned. Once the streams had been computed, they were used to build a mask of 1's and 0's, that had the same dimensions as the original cochleagram, to yield new separate frequency/time regions that could be associated with the individual sources of each sound.

This process is depicted in **Figure 4,** where there are 3 clusters that are shown that fall within the 3 feature axes: time, pitch, and location. The probability that the features belong to a particular cluster is used to generate a 2D Ideal Binary Mask (IBM) of 1's and 0's, the size of the cochleagram. That mask was applied (multiplied, cell by cell) to the original cochleagram. Applying this mask to the cochleagram that contained information from multiple sound sources, is the CASA model's way of isolating sound sources from the mixture of received sounds. One way to depict the segregation is to display the cochleagram that results after the masking operation has occurred on each cluster. The amount of separation becomes evident when the spectral and temporal attributes that belong to each cluster, that is associated with an individual source, are noticeably different from one another. By virtue of this approach, the CASA model attempts to solve the "cocktail party problem". **Figures 5, 6,** and **7** in the **Results** section demonstrate this masking operation.

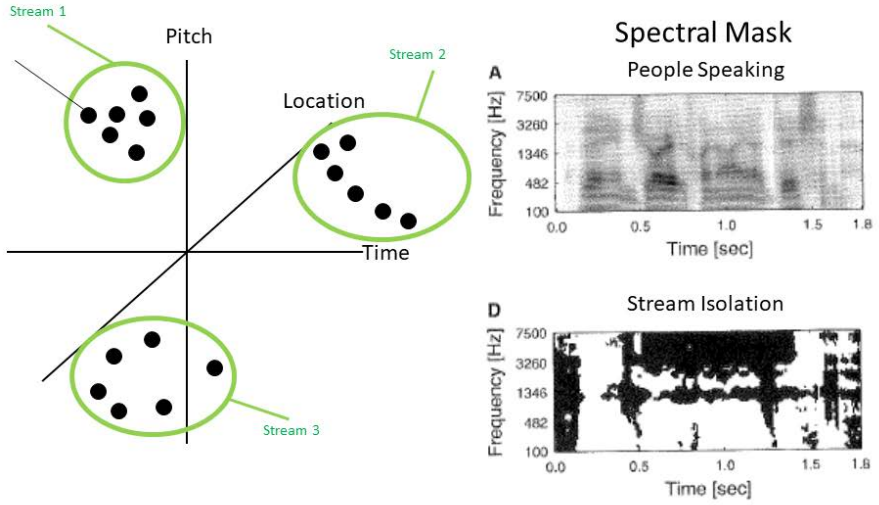


Figure 4. Clustering (stream separation) based on time, pitch, and location of each stream, and using the results to apply an "Ideal Binary Mask" (IBM) to the original cochleagram. Each stream's mask (bottom right figure), when applied to the

cochleagram (top right figure), yields the contribution of the corresponding source to the overall received sound. The "black" regions have a value of "0," while the "white" regions have a value of "1." Thus, when the mask is applied (element by element multiplication) to the cochleagram, only the areas of the cochleagram, under the "white" regions, will be preserved, representing the separated "streams." This procedure is completed for each stream (or cluster), yielding a new cochleagram for each stream (or cluster).

2.5 Object/Source Classification

The last goal of CASA is to categorically *classify* the *segregated* "*streams*". Support Vector Machines (SVMs), Neural Networks, Markov Models, and Principal Component Analysis (PCA) are some of the techniques that can be used to associate (classify) a stream with a source category (Bishop, 2006). For our purposes of integration with ARCADIA, classification would yield four general categories of sources—in order of complexity—include: tones, events, music, and human speech.

This portion of the software front-end has not yet been implemented. However, we plan to build a Support Vector Machine (SVM) module that will ingest the streams that the GMM has produced, and classify them into an "*object*" or type of sound, such as a siren (multiple tones), musical instrument, human speech, or specific short duration event of some sort.

3.0 Results and Discussion

This software effort resulted in the development of a human inspired (CASA-based) computer application that has completed three of the four steps necessary to provide audio cues to a cognitive architecture. The fourth step (classification) is in progress. However, the software program, as it stands, is a fully functioning program that decomposes sounds by frequency and time, extracts essential auditory features, and then groups related features into streams. Once classification of the streams to form objects, or sound sources, has been completed, the CASA-based software application will be able to feed an external cognitive model with the *type* of source that produced the sound, and its attributes, such as: *time, azimuth, predominant frequency* and *amplitude*.

To verify that this software performs as expected, we fed two sounds to the program: a tone and a saxophone signal, accompanied with background white noise. Results from the signal processing steps are described in detail in the **Appendices**. A quantitative comparison of intermediate processing results between the Clojure and MATLAB® implementations was completed as one method of software verification. Qualitative comparisons were also done as illustrated by the figures that are included in this section. The Clojure derived figures were compared to analogous figures generated from MATLAB® code by visually inspecting each pair. In short, from a quantitative and qualitative stand point, the software appears to have successfully separated the sounds into three clusters, based on features axes of amplitude, frequency, azimuth, and time (similar to those in **Figure 4):** one cluster corresponding to the tone, the second cluster corresponding to the saxophone, and the third and last cluster corresponding to the noise. Each of these clusters was used to create a new cochleagram where the *masked* frequency/time regions differed based on the source of the sound, as illustrated in **Figures 5**, **6**, and **7**. Since the separate regions correspond to separate sound sources, the software appears to be able to solve a simple case of the "cocktail party problem".

When interpreting **Figure 5** and **Figure 6,** one can see how the tone and saxophone signals were separated reasonably well into their respective clusters. For example, **Figure 5** illustrates the segregation processing that was associated with cluster #1. The amplitude of the tone in this figure appears to be much higher than the loudest saxophone tone. Conversely, segregation associated with cluster #2 is depicted in **Figure 6**. This figure illustrates how the tone does not fit well within cluster #2, but that the saxophone is greatly amplified with respect to the tone. Additionally, **Figure 7**, which illustrates the segregation processing that is associated with cluster #3, shows that both the tone and the saxophone notes do not fit very well into the 3rd cluster. This is because the GMM clustering was tuned to the background noise rather than to either the tone or the saxophone. The manifestation of this is that both signals' amplitudes appear to have nearly an equal response to the GMM clustering operation.

The Clojure code was verified by comparing its results to those generated by similar MATLAB® functions. From the analysis of these comparisons, we found that the figures generated by the Clojure code qualitatively agreed well with those produced by the MATLAB® code, except for the onset and offset feature maps, where there was a difference with the Clojure and MATLAB® algorithms that were implemented. Additionally, intermediate processing results were seen to quantitatively agree between the Clojure and MATLAB® codes. As a result, we determined that the Clojure implementation worked as designed, since the MATLAB® code was considered to be the "benchmark" for our comparison. However, we also found that the executable code was very slow (about two orders of magnitude slower) and required excessive memory (greater than 16 GB), limiting its practicality as a true front-end to a cognitive model that runs in real time. However, it does represent a proof of concept that spans the CASA processing steps to include cochlear response, feature extraction, and stream segregation. Although the software is written in Clojure, the same design could be followed, and the code ported to another language, in order to act as an auditory front-end to a different cognitive architecture, other than ARCADIA. The next phase of development will attempt to extend the processing to include classification of the segregated streams and feature extraction that is tailored to feedback from the cognitive model.

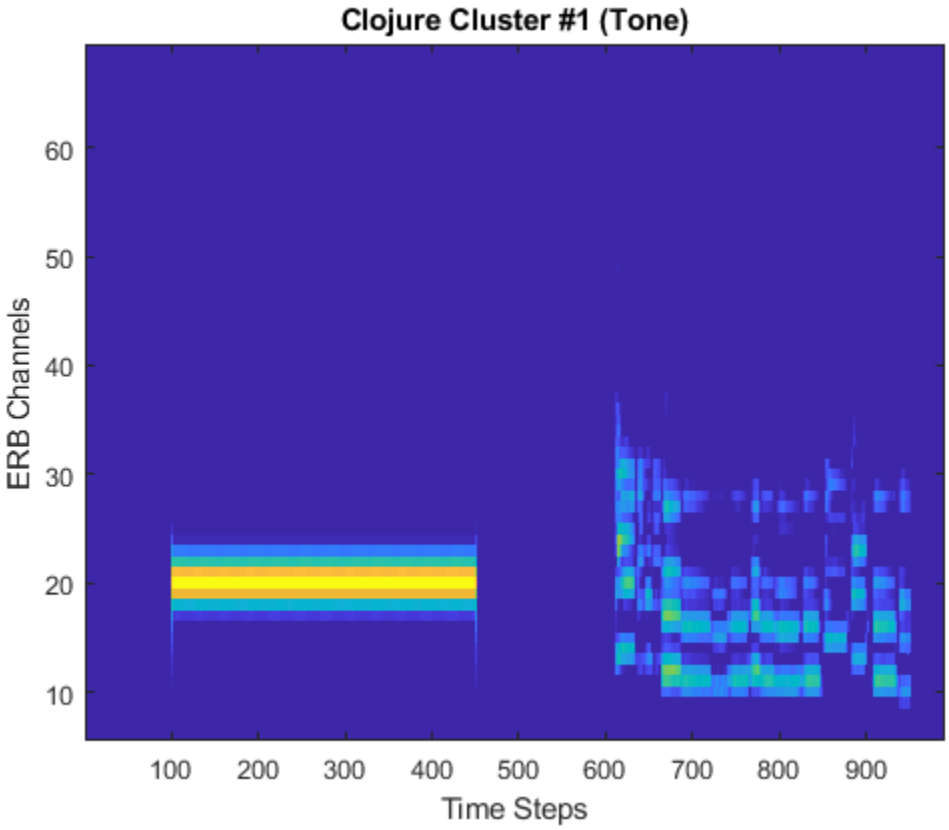


Figure 5. Cochleagram results after the Mahalanobis distance (see Appendix C.4.3) between each observation and the cluster #1's mean has been used as a threshold to mask the original cochleagram's power values. The color represents the power of the signals. Note that the tone between time segments 100 and 450 has been accentuated with respect to the saxophone notes that appear between time segments 600 and 950. This is due to the fact that the original means for Cluster #1 were chosen within the tone (signal) region of the original cochleagram. This illustrates the ability of the Gaussian Mixture Model (GMM) to separate disparate streams when there are an equal number of clusters and streams.

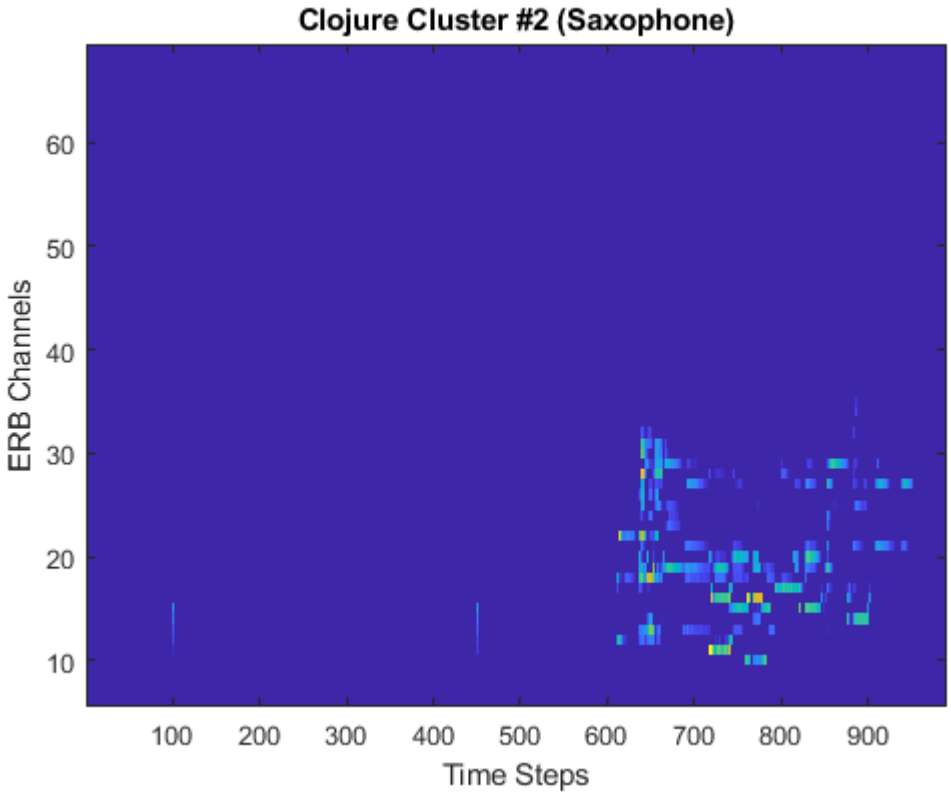


Figure 6. Cochleagram results after the Mahalanobis distance (see Appendix C.4.3) between each observation and the cluster #2's mean has been used as a threshold to mask the original cochleagram's power values. The color represents the power of the signals. Note that the tone between time segments 100 and 450 has been attenuated with respect to the saxophone notes that appear between time segments 600 and 950. This is due to the fact that the original means for Cluster #2 were chosen within the saxophone notes (signal) region of the original cochleagram. This illustrates the ability of the Gaussian Mixture Model (GMM) to separate disparate streams when there are an equal number of clusters and streams.

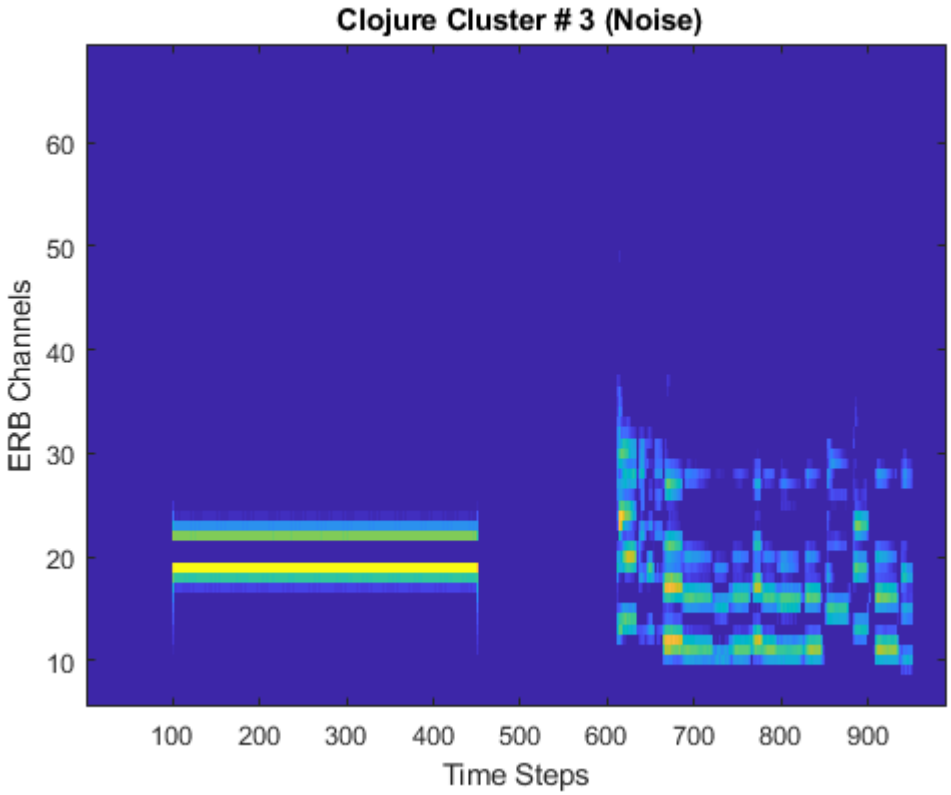


Figure 7. Cochleagram results after the Mahalanobis distance (see Appendix C.4.3) between each observation and the cluster #3's mean has been used as a threshold to mask the original cochleagram's power values. The color represents the power of the signals. Note that the tone between time segments 100 and 450 and the saxophone notes that appear between time segments 600 and 950 have not been affected by the masking or have been affected by the masking about equally. This is due to the fact that the original means for Cluster #3 were chosen within a noise region of the original cochleagram. This illustrates the ability of the Gaussian Mixture Model (GMM) to separate disparate streams when there are more clusters than there are actual streams.

4.0 Conclusion

As seen with humans, the auditory system can discern individual sources of sound within a complex acoustic environment. Human audition also complements the visual system, which can result in enhanced performance in visual search tasks. Although the Clojure code was built specifically to integrate with the ARCADIA cognitive architecture, it demonstrates a physiologically inspired approach to addressing the "*cocktail party problem,*" through CASA techniques, in that it can segregate a mixture of sounds into "streams" based on the commonalities of auditory "features" emergent from a single auditory source. This capability is valuable by providing audio cues to cognitive architectures and/or systems, which in turn may provide improved performance in terms of reduced reaction time and increased accuracy in response to audio and visual stimuli. Incorporating the CASA techniques, demonstrated by this effort, could benefit commercial and military Audio-Visual-Integrated (AVI) systems, such as autonomous devices like robots, weapons, and vehicles by augmenting their visual capabilities with auditory cuing.

4.1 Future Tasks

Everyday environments often contain many competing sounds, yet the human brain has mechanisms for selecting a single source (Fritz et al, 2007). This selection may occur based on the relative salience of sounds in the mixture, with the most important sound capturing a listener's attention (de Counsel et al, 2010). To accommodate this phenomenon, a future aspect of this project will be to choose the most salient "*bottom-up*" stream among those that have been segregated from the mixture such that the model will "attend" to it (de Counsel et al, 2010, Bregman, 1990)). Humans may also base their attention on achieving "*top-down*" goals (Yarbus, 1967, Bregman, 1990). We plan to address both of these capabilities within a combined model, as provided by feedback from the cognitive architecture.

We released the software at a stable but intermediate state to NRL so that they could witness our progress and learn how to best integrate our efforts into their architecture. Some of the tasks identified below are incomplete. Therefore, in order to fully integrate the CASA software into ARCADIA, the following items need to be addressed:

1. In order to evaluate the robustness of our approach to CASA, the model should be tested with more dynamic and uncontrolled auditory scenes to evaluate the robustness of the stream segregation component. For example, the tone and the saxophone notes could be overlapped in time and in azimuth, and the processing repeated to further evaluate the code's veracity.
2. The clustered results need to be sent to a classifier as part of the ARCADIA integration effort. The choice of this function has yet to be determined, however current exploration efforts are favoring Support Vector Machine (SVM) algorithms because the boundary that separates the data points into their respective groups need not be straight lines as is the case with some simpler classification techniques. In so doing, one can capture more complex relationships among the observations (Bishop, 2006).
3. Integration with ARCADIA's "*Interlingua*" infrastructure needs to be accomplished.
4. ARCADIA "*components*" need to be built that will address the attention-based factors that drive the behavior of the audio and visual portions of the architecture to produce a comprehensive audio-visual cognitive model.
5. Other features may need to be investigated and evaluated for their efficacy in the segregating streams. Options that are currently being considered are beyond the scope of this report.
6. The current Clojure coding should be made more efficient, or ported to another language, since this implementation will not support the signal processing that is required to run in near real-time.
7. The MATLAB® code, upon which the Clojure code is based, as well as any future code, needs to be rewritten so that it runs in near real-time by processing smaller "chunks" of the WAV file.
8. If MATLAB® will be used from now on to provide the "*streams*" that are fed to ARCADIA, then a communication interface needs to be implemented that will transfer the auditory information to and from ARCADIA. A "*socket*" based interface has been built and may become an attractive communications mechanism.

4.2 Model Limitations/Suggestions

1. The Clojure model is currently very inefficient. (Using “idiomatic” Clojure, porting to another language, faster processor, and parallel processing are potential solution options)
2. The Clojure (or MATLAB®) code needs to send its results to a “down-stream” application via some communications mechanism. (A network socket method is one potential option).
3. The MATLAB® code needs to operate on much smaller chunks of time and send the processed results in such a way that the duty cycle runs in near real-time. (Strategic buffering may be required).

References

1. Bishop, Christopher M., (2006), "Pattern Recognition and Machine Learning", Springer Science+Business Media, LLC., ISBN-10: 0-387-31073-8, ISBN-13: 978-0387-31073-2.
2. Braga, Rodrigo, M., Fu, Richard Z., Seemungal, Barry M., Wise, Richard J.S., Leech, Robert, (2016), "Eye Movements during Auditory Attention Predict Individual Differences in Dorsal Attention Network Activity", Frontiers in Human Neuroscience, 09 May 2016. Available from: https://doi.org/10.3380/fnhum.2016.00164.
3. Bregman, A.S., Dannenbring, G.L., (1973), "The effect of continuity on auditory stream segregation", Perception & Psychophysics, Vol. 13, No. 2, pp 308-312.
4. Bregman, A.S. (1990), "Auditory Scene Analysis: The Perceptual Organization of Sound", MIT Press, Cambridge, MA.
5. Bridewell, Will, Bello, Paul, (2016). "A theory of Attention for Cognitive Systems", Advances in Cognitive Systems 4, 1 - 16.
6. Cherry, E. Colin, (1953), "Some Experiments on the Recognition of Speech with One and Two Ears," The Journal of the Acoustical Society of America. 25(5):975-79.
7. Chi, T., Ru, P., & Shamma, S. A., (2005), "Multiresolution spectrotemporal analysis of complex sounds", The Journal of the Acoustical Society of America, 118(2), 887-906.
8. De Coensel, B., & Botteldooren, D. (2010), "A model of saliency-based auditory attention to environmental sound", In 20th International Congress on Acoustics (ICA-2010) (pp. 1-8).
9. DoD: Department of Defense. (2019), "Department of Defense Strategic Medical Research Plan: response to Section 736 of the John S. McCain National Defense Authorization Act for Fiscal Year 2019", (Public Law 115-232).
10. Elhilali, M., & Shamma, S. A. (2008), "A cocktail party with a cortical twist: how cortical mechanisms contribute to sound segregation", The Journal of the Acoustical Society of America, 124(6), 3751-3771.
11. Fritz, J.B., Elhilali, S.V. David, and Shamma, S.A., (2007), "Auditory attention – focusing the searchlight on sound", Curr. Opin. Neurobiol., vol 17, no, 4, pp. 437-455.
12. Grira, Nizar, Michel Crucianu, and Nozha Boujemaa, (2004), "Unsupervised and semi-supervised clustering: a brief survey", *A review of machine learning techniques for processing multimedia content* 1 (2004): 9-16.
13. Hasan, Bashar Awwad Shiekh and Gan, John Q, (2009), "Sequential EM for Unsupervised Adaptive Gaussian Mixture Model Based Classifier", in Perner P. (eds) Machine Learning and Data Mining in Patter Recognition. MLDM 2009. Lecture Notes in Computer Science, vol. 5632, Springer, Berlin, Heidelberg.
14. Krishnan, L., Elhilali, M., & Shamma, S. (2014), "Segregating complex sound sources through temporal coherence", PLoS computational biology, 10(12), e1003985.
15. Lyon, Richard F. (2017), "Human and Machine Hearing – Extracting Meaning from Sound", Chapter 9, Cambridge University Press.
16. Lyon, Richard F. (1983), "A Computational model of binaural localization and separation", In Proceedings of the International CONFERENCE ON Acoustics, Speech and Signal Processing, pp 1148-1151.
17. National Institute of Biomedical Imaging and Bioengineering (National Institutes of Health), "Computational Modeling Fact Sheet", https://www.nibib.nih.gov/sites/default/files/Computational Modeling Fact Sheet.pdf.
18. Office of Naval Research: ONR. (n.d.). Naval Research and Development: A Framework for Accelerating to the Navy & Marine Corps After Next). Available from: https://www.onr.navy.mil/en/our-research/naval-research-framework.

19. Office of Naval Research, Division 342: Auditory Neuroscience & Performance. Available from: https://www.onr.navy.mil/en/Science-Technology/Departments/Code-34/All-Programs/warfighter-protection-applications-342/auditory-neuroscience
20. Patterson, R., Nimmo-Smith, I., Holdsworth, J., Rice, P., (1987), "An efficient auditory filterbank based on the gammatone function". Paper Presented at a Meeting of the IOC Speech Group on Auditory Modelling at RSRE.
21. Pillow, Jonathan, (2017), "Auditory System & Hearing Chapter 10", "Sensation & Perception, Lecture 18" from PSY 345 / NEU 325. Available from: http://pillowlab.princeton.edu/teaching/sp2017/slides/Lec18_Hearing_Chap10.pdf
22. Reynolds, Douglas A., Rose, Richard C., (1995), "Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models", IEEE Transactions on Speech and Audio Processing, Vol. 3, No. 1, January.
23. Romans, N., Wang, D.L., Brown, G.J., (2003), "Speech segregation based on sound localization", The Journal of the Acoustical Society of America, Volume 114, Issue 4.
24. Shannon, Robert V., Fu, Qian-Jie, Galvin III, John, (2004), "The Number of Spectral Channels Required for Speech Recognition Depends on the Difficulty of the Listening Situation", Acta Otolaryngol, Supl 552: 50-54.
25. Undersea Warfare: Undersea Warfare Chief Technology Office, (2016). "Undersea Warfare Science & Technology Objectives". Available from: https://www.navsea.navy.mil/LinkClick.aspx?fileticket=Z0Z0mzYhhhw%3d&portalid=103
26. Wang, DeLiang, Brown, Guy J., (2006), "Computational Auditory Scene Analysis: Principles, Algorithms, and Applications", 1st Edition, John Wiley & Sons.
27. Woodworth R. S., and Schlosberg H. (1954), "Experimental Psychology", Holt, Rinehard, and Winston, New York: pp. 349–361.
28. Yarbus, A. L. (1967), "Eye movements and vision" (B. Haigh, Trans.). New York: Plenum Press.
29. Yost, William A., (2000), "Fundamentals of Hearing", 4th Edition, Academic Press.

Abbreviations and Acronyms

AM – Amplitude Modulation
ARCADIA – Adaptive Reflective Cognition in an Attention Driven Integrated Architecture
ASTM – Auditory Short Term memory
CASA – Computational Auditory Scene Analysis
DFT – Discrete Fourier Transform
DoD – Department of Defense
ERB – Equivalent Rectangular Beamwidth
FFT – Fast Fourier Transform
FM – Frequency Modulation
GMM – Gaussian Mixture Model
HRTF – Head Related Transfer Function
IBM – Ideal Binary Mask
ILD – Interaural Level Difference
ITD – Interaural Time Difference
NRL – Naval Research Laboratory
NSMRL – Naval Submarine Medical Research Laboratory
ONR – Office of Naval Research
PCA – Principal Component Analysis
PSD – Power Spectral Density
REPL – Read Execute Print Loop
SVM – Support Vector Machine
U.S. – United States
UUV - Unmanned Underwater Vehicle
VSTM – Visual Short Term Memory

Appendix A: Software Development Approach

This document is mainly concerned with how the front-end software, written in Clojure, is integrated with the back-end, ARCADIA cognitive architecture, also written in Clojure. In our design, acoustic object or source information will enter into the ARCADIA cognitive model as an "*Interlingua*" message. Such a message could be comprised of a Clojure "*map.*" The plan is for the CASA software to provide *Interlingua* messages to ARCADIA, and for these messages to be deposited into a buffer called the "*Auditory Short Term Memory*" (ASTM). This part of the ARCADIA architecture is equivalent to the already existing "*Visual Short Term Memory*" (VSTM). In the case of an auditory stimulus, each message sent to a cognitive model defines an object (sound source) accompanied by its attributes. It is intended that the attention-based ARCADIA model will be able to use this information to cue its eyes to look in a direction of the auditory object that the CASA model has identified. Attributes that might be included are: time stamp, ID (object type: tone, music, event, human speech), onset and offset times, frequency content (fundamental frequency, harmonics), azimuth, and level of confidence.

At this point in the software development cycle, the Clojure code separates a mixture of sounds into its correlated components (streams). It does not yet identify what source has made those individual sounds. All along, however, the software development approach has been to try to adhere to physiologically inspired algorithms, implement them first in MATLAB®, and then convert the essence of those functions into Clojure. The next sections concentrate on the software engineering specifics of the development process.

A.1 Interaction with MATLAB®

We first developed the auditory front-end software prototype in MATLAB®, and then coded equivalent functionality in Clojure, comparing the processing results from the two implementations. Several functions that are intrinsic to MATLAB® had to be coded separately in Clojure because they could not be found from publicly available sources. Additionally, little regard was given to optimizing the Clojure code with respect to either run-time or memory usage. The Clojure code is highly *non-idiomatic* in its current state. However, it is able to process about 10 seconds of recorded stereo audio data, in the form of WAV files, but it requires large amounts of time (1 day) and large amounts of computer memory (> 16 gigabytes). As seen in this document, the Clojure software is able to extract seven "features" from an input sound file, and yield three streams from a Gaussian Mixture Model (GMM) clustering algorithm. The intent is that these streams will be fed to a classifier (SVM module) for use by future ARCADIA cognitive model's "*attention-aware*" components. The following section describes the functions within the current CASA processing chain, and how they are hierarchically arranged.

A.2 Development Platform

The Clojure software was built and tested on an MS Windows computer running the Windows 7 operating system, and a Java Virtual Machine (JVM). The Clojure code was tested against the Windows version of MATLAB® v 13b. The Clojure code was run and tested using the *leiningen* infrastructure, which included its "*repl*" (Read Execute Print Loop) capability. Debugging was done with "println" statements embedded in the code.

The Clojure implementation requires one external signal processing library. It is used to read the stereo WAV file(s) and to perform forward and inverse Fast Fourier Transforms (FFTs). This library "*mikera*" came from a repository in "*GitHub*" by Mike Anderson.

Appendix B: Software Usage

This section describes how to use the software application as a demonstration and proof of concept.

Separate flowchart diagrams are available in **Appendix D** that illustrate the functions along with their inputs and outputs that appear as labels to the arrows that link the various functions. Consulting that flow-charts might assist one in following the description below.

Currently, the software is configured to process 9.9 seconds of a 10 second WAV file called: "*resources/tones_and_sax_10_sec.wav*".

(The current software configuration is hardwired in the code to write files to: *"src\casa"). This is the same directory that houses the source code.*

If the appropriate processing switches (flags) are set to "1," then the code will create seven feature maps and cluster them into three streams. The results are written to text files so that they can be processed separately and viewed independently with MATLAB®.

B.1 Executing the Program

First, one should install the ***"leiningen***" release such that one can run its "REPL". (REPL = Read Execute Print Loop)

Once that has been done, enter the following commands. Start a Windows "*command window*"

"cd" to the directory where the Clojure code exists

(The current configuration is: *"d:\Clojure\dynne\dynne-test\src\casa")*

"lein –o deps"

"lein repl'

Open a second Micrsoft Windows "*command window*"

Open the source file "*casa_main_program.clj*" in a text editor

Copy and paste the entire contents of the source file "*casa_main_program.clj*" into the first command window that is running the "*repl"*

(Whatever was copied to the REPL will execute, without interruption).

B.2 The Main Program

The CASA main program is interpreted, line by line, and executes as it finishes interpreting the code. Depending on the processing switches, within the main program, certain functions are executed and others will be skipped. The processing switches (flags) and their effect(s) are listed in **Table 2** below:

Table 21. Software switches (flags) that control the processing flow from the main program.

def DO-GAMMATONE-FILTER-BANK 1	Invoke the Gammatone Filtering
def DO-COCHLEAGRAM 1	Generate a Cochleagram
def DO-ITD-ILD 1	Find the azimuth via an ITD approach (and ILD)
def DO-RATE-FILTER 0	Apply a "rate filter" to the cochleagram (probably don't want to turn this function on. It takes too long, with questionable results)
def DO-SCALE-FILTER 1	Apply a "scale filter" to the cochleagram
def DO-FUND-FREQ-DETECT 1	Find a primary and secondary fundamental frequency from the cochleagram
def DO-AUTO-CORRELATION-FLAG 1	Flag used to (1) compute the auto-correlations or (0) simply read them from a pre-existing text file. (Done inside "casa-fund-freq")
def DO-PEAK-PICK 1	Flag to pick peaks from the fundamental frequency surface (1 = pick 2 peaks, 0 = don't pick any peaks)
def DO-ONSETS-OFFSETS 1	Find onsets and offsets within a cochleagram
def DO-GMM 1	Cluster the "observations" within "features" using a Gaussian Mixture Model

B.3 Results Visualization (Requires MATLAB®)

MATLAB® was used to plot the results of the various processing steps (see **Figures 5 – 17**).

The following are the MATLAB® commands used to display the clustered streams, once one has opened one of the files in a text editor (such as "*Notepad*"), and copied its contents with "*cntrl-c*":

a = *cntrl-v*; (paste the previously copied contents into the variable "a")

ar= reshape (a, [64 990]); *(In the case of matrices that have the dimensions of the cochleagram)*

ar = -ar; *(Needed only for plots of: Fundamental Frequency, Onset, and all 3 of the Separated Streams)*

figure

imagesc (ar)

B.4 Intermediate and Diagnostic Files

The program writes out several intermediate and diagnostic files as processing proceeds. The files are written to the same directory in which the source code resides. These files are ASCII text files that have been output from the Clojure "*spit*" function. Therefore the first character is "*[*" and the last character is "*]*". This makes it convenient to copy and paste these files into MATLAB® for plotting. Because "spit" takes a vector as an argument, even 2D arrays are written as concatenated vectors that yield a single long vector when saved to the disk.

(The current configuration is hardwired, in the code, to write files to: *"src\casa")*

The files include:

Audio_channel_1_filtered_data.txt – Gammatone filtered time series for audio channel #1

Audio_channel_2_filtered_data.txt - Gammatone filtered time series for audio channel #2

proto_cochleagram.txt - 64 ERB channels x 990 time segments of pre-cochlear amplitudes

cochleagram.txt – 64 ERB channels x 990 time segments of cochlear amplitudes (in dB)

itd.txt – 64 ERB channels of "raw" time lagged amplitudes for each cross-correlation window (signed floating point values)

itd_avg.txt – 64 ERB channels of averaged (downsampled) time lagged amplitudes for each cross-correlation window (signed floating point values)

ild.txt – 64 ERB channels of "raw" magnitudes for each cross-correlation window (positive floating point values). (This file is used for diagnostics only)

scale_filtered_cochleagram_half.txt - 64 ERB channels x 990 time segments of scale filtered cochlear amplitudes (in dB), as a result when the *"SCALE_VALUE*" was set to "0.5".

scale_filtered_cochleagram_one.txt - 64 ERB channels x 990 time segments of scale filtered cochlear amplitudes (in dB), as a result when the *"SCALE_VALUE"* was set to "1.0".

summed_erb_auto_correlation_magnitudes.txt – an intermediate file of auto-correlation values from the summation of the ERB channels, used in finding the fundamental frequencies from the cochleagram. Frequency samples within each time segment is very dense.

primary_secondary_fund_freq.txt – 64 ERB channels x 990 time segments of ERB channel index at which the frequency spectrum generated the largest and second largest magnitude.

fund_freq_maximum.txt - 64 ERB channels x 990 time segments of ERB channel index at which the largest auto-correlation peak magnitude occurred within each time segment.

fund_freq_averaged.txt - 64 ERB channels x 990 time segments of ERB channel index that contains the average of the auto-correlation magnitudes that occurred within each time segment.

onsets.txt - 64 ERB channels x 990 time segments that contains the time segment index where the sound energy rose above some amplitude threshold.

offsets.txt - 64 ERB channels x 990 time segments that contains the time segment index where the sound energy dropped below some amplitude threshold.

masked_cochleagram.txt – cochleagram after a binary mask was applied (used as input to the GMM).

masked_averaged_itd – ITD surface after a binary mask was applied (used as input to the GMM).

masked_scale_filter_1.txt – scale filtered cochleagram #1 after a binary mask was applied (used as input to the GMM).

masked_scale_filter_2.txt – scale filtered cochleagram #2 after a binary mask was applied (used as input to the GMM).

masked_fund_freq.txt – fundamental frequency surface after a binary mask was applied (used as input to the GMM).

masked_onsets.txt – onset surface after a binary mask was applied (used as input to the GMM).

masked_offsets.txt – offset surface after a binary mask was applied (used as input to the GMM).

gmm_mu_output.txt – output from the GMM function that holds the clusters' "means" for each feature.

gmm_cov_output.txt – output from the GMM function that holds the clusters' "covariance matrix" for each feature.

casa_binary_masked_cochleagram_cluser_1.txt – cochleagram after applying a mask, based on the Mahalanobis distance that each feature observation was from cluster #1's means.

casa_binary_masked_cochleagram_cluser_2.txt – cochleagram after applying a mask, based on the Mahalanobis distance that each feature observation was from cluster #2's means.

casa_binary_masked_cochleagram_cluser_3.txt – cochleagram after applying a mask, based on the Mahalanobis distance that each feature observation was from cluster #3's means.

casa_probability_masked_cochleagram_cluser_1.txt – cochleagram after applying a mask, based on the probability that each feature observation belonged to cluster #1.

casa_probability_masked_cochleagram_cluser_2.txt – cochleagram after applying a mask, based on the probability that each feature observation belonged to cluster #2.

casa_probability_masked_cochleagram_cluser_3.txt – cochleagram after applying a mask, based on the probability that each feature observation belonged to cluster #3.

After the GMM function has been invoked, two diagnostic files are written. The first holds the means of each clustered feature, called *"gmm_mu_output.txt*" (3 clusters x 7 features). The other diagnostic file holds the final covariance matrix associated with each clustered set of features. It is called "*gmm_cov_output.txt*" (3 covariance matrices, each 7 x 7 elements in size).

Appendix C: Software Implementation, Function Description, Demonstration

This section contains some of the Clojure specific coding conventions and short-comings as well as a description of the Clojure modules that comprise the application. They are listed and described in the order in which they are intended to be executed, starting with the main "driver" program.

A separate flow-chart diagram is available that illustrates the functions along with their inputs and outputs that appear as labels to the arrows that link the various functions. Consulting that flow-chart might assist one in following the description below. It shows the function and data flow that is required to execute three of the four of the major functions that comprise the "front-end" to ARCADIA:

1. Main Program
2. Feature Extraction
3. Stream Segregation
4. Object Classification (not included)

C.1 Coding Conventions, Idiosyncrasies, Lessons Learned

"*casa-*" precedes the module name (that NSMRL wrote) for ease in identifying similar functions

The string "*;; *" precedes comments within the source code that describe the functionality.

Revision history is included within each source file.

Constants appear as UPPER CASE text.

Variables appear as lower case text.

Array names are prefaced with the string: "*a2d-*" or "*a3d-*".

Vector names are prefaced with the string: "*v-*".

Function names contain "-" (hyphens) in their names. This is an important convention.

File names contain "_" (under-bars) in their names. This is an important convention.

Indentation is important for readability.

C.1.1 Clojure Coding Limitations

Separate functions should be responsible for building and applying the individual feature masks.

There is inconsistent and inadequate use of "*let*" in order to accommodate repetitious use of variable names.

The main program is too monolithic. It needs to be modularized.

The entire NSMRL-generated code base is non-idiomatic and needs re-factoring to speed it up and reduce its memory footprint.

C.2 The Main Program

casa-main-program.clj – This is the main driving program that performs the Gammatone filtering, followed by the feature extraction, followed by the Gaussian Mixture Model clustering. One of the first steps in the processing chain is to generate the ERB channel's center frequencies. This is done by the function "*casa-make-erb-frequencies*". The subsequent feature extraction functions require the vector that results from this operation.

C.3 Feature Extraction

1. Cochleagram Generation
2. Azimuth (ITD) Determination
3. Scale Filter the Cochleagram to find consistent bandwidth tones (2 of them: narrow and wide)
4. Fundamental Frequency Determination
5. Onset and Offset Determination
6. Rate Filter the Cochleagram (*Not Used in Stream Segregation*)

C.3.1 Gammatone Filtering

In order to begin emulating a human's hearing, the software filters the audio stereo time series with a Gammatone Filter Bank. This filter is built for each of the ERB channels. This is accomplished by the function "*casa-gammatone-filter-bank.*" This function reads a text file that contains filter coefficients used in generating the Gammatone Filter Bank. This file is called *"CASA_Filter_Coefficients_lookup_table.txt."* This file was generated ahead of time by a MATLAB® script. This file contains the coefficients for band passing, interpolating, and down sampling the lower ERB channels (below 400 Hz).

These coefficients are used to compute the group delays that will be applied later when the time series of each ERB channel are aligned with one another. It will then, call the function "*casa-input-time-series*" that uses the function "*list_samples_dhg*" inside the file "*sampled_sound.clj*" to read the stereo WAV file, as defined in "*casa_main_program.clj*" as: "STEREO-WAV-FILENAME" ("*resources/tones_and_sax_10_sec.wav*"), and apply the Gammatone Filter Bank, in the time domain, to the 2 audio time series. This is done by the function "*casa-apply-filters.*" Two text files will be written that contain 10 seconds of 64 ERB channels of filtered data. The files are called *"audio_channel_1_filtered_data.txt"* and called *"audio_channel_2_filtered_data.txt."*

The function "*casa-prepare-cochleagram*" accepts the 2 filtered time series for each ERB channel, averages them together, and time aligns the result with the other ERB channels. It returns a 2D array, and also saves the resulting array, as a vector, in a file called "*proto-cochleagram.txt,*" which is used as input to the next step.

C.3.2 Cochleagram Generation

In order to transition from the "proto-cochleagram" to a "cochleagram," the software applies a "hair cell model" to the "proto-cochleagram." The function *"casa-apply-hair-cell-model"* does this by "half-wave" rectifying the time series samples for the ERB channels and breaking up the time series into multiple 10 milli-second second time segments, and averaging the contents of each, resulting in a 2D matrix of amplitude values for each time-frequency point. It writes a file called "*cochleagram.txt*" as a vector that can be used by other functions. **Figure 8** shows the result of the cochleagram generation processing. Amplitudes of this surface become one of the features that is fed to the stream segregation logic later in the processing. This array is also the basis for many of the feature extraction functions that follow.

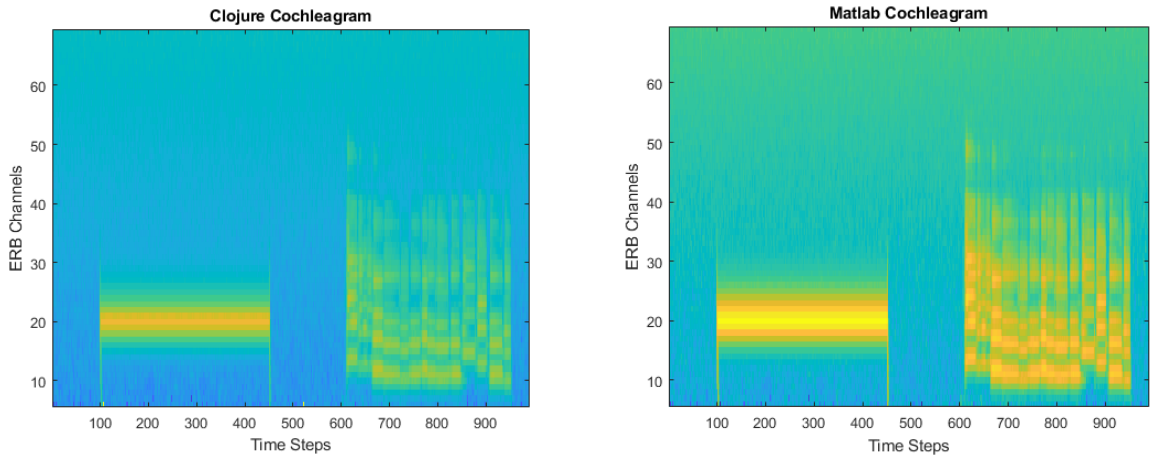


Figure 8. Cochleagram. ERB Frequency Channels versus Time Segments (0.01 seconds each). The color represents the power of the signals. The figure shows a tone that occurs for 3.5 seconds (time segments 100-450), and saxophone notes for 3.5 seconds (time segments 600-950), separated by 1.5 seconds (time segments 450-600). The left-hand surface is a product of the Clojure code, the right-hand surface is from the MATLAB® code. The difference in the appearance of the two surfaces is due to the different dynamic ranges of the power levels.

C.3.3 Azimuth

One of the features that is fed to the stream segregation function is azimuth. The function "*casa-find-itd*" performs the operations that produces a surrogate for azimuth. The inputs are the 2 Gammatone-filtered stereo time series. Each time series is broken up into many smaller time segments, segments from one time series are cross-correlated with segments from the other time series. This is done in the frequency domain (Fast Fourier Transform) by the function "*casa-cross-correlate-in freq-domain,*" which involves a forward and inverse FFT (*casa-half-fft*" and "*casa-half-ifft*"). Their time lags for each ERB channel and time segment are stored, as a vector, in a text file called "*itd.txt.*" This vector is densely sampled as a result of the FFT. Therefore, in order to reduce its resolution, each time segment for each ERB channel is averaged. The result of this down sampling is saved in a text file called "*itd_avg.txt.*" The function "*casa-downsample-itd-results*" completes this task. Time lags can take on negative and positive values as well as 0. The number of lag samples is related to the arrival time difference, since the sample rate is known (44100 Hz). As mentioned above, these time lags could be converted to azimuth, by knowing the distance that sound in audio channel #1 had to travel with respect to audio channel #2, and

the sound speed. But the software stops at simply producing the signed time lags. **Figure 9** shows the result of the azimuthal processing.

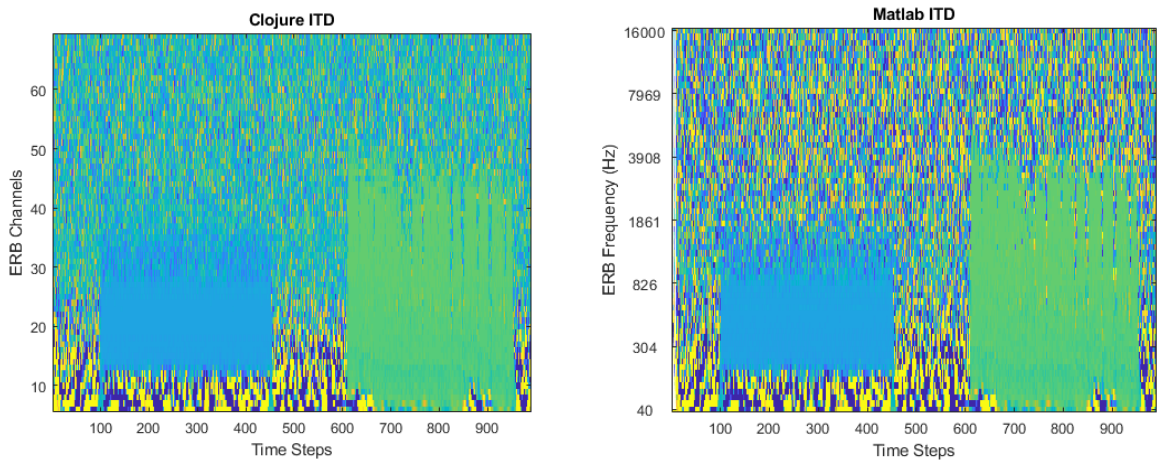


Figure 9. Azimuth (Interaural Time Difference (ITD).ERB Frequency Channels v Time Segments (0.01 seconds each). The figure shows a tone that occurs for 3.5 seconds (time segments 100-450), and saxophone notes for 3.5 seconds (time segments 600-950), separated by 1.5 seconds (time segments 450-600). The color coding indicates the ITD time lags between the rght and left stereo audio channels. The ITD time lags are negative (blue) for the tone, and the time lags are positive (yellow/green) for the saxophone. This indicates that the tone came from a different direction (signals came from the left of center) as opposed to the saxophone notes (signals came from the right of center). The left-hand surface is a product of the Clojure code, the right-hand surface is from the MATLAB® code.

C.3.4 Scale Filter

In order to find consistent bandwidth signals of the entering sound, the cochleagram is "scale filtered" by the function "*casa-scale-filter*." This is done at two different frequency bandwidths, producing two new, but filtered versions of the original cochleagram. The main program calls this function twice in order to generate two features that will be fed to the stream segregation function. Currently, the scale filter settings are 0.5 and 1.0 cycles/ERB. The scale filter function first builds a wavelet. The wavelet then undergoes a Hilbert Transform, which yields complex values. These complex values are then "scaled" by the entering "*scale-value*" bandwidth parameter. A vector of 64 ERB channel samples, at a particular time segment, is then convolved separately with the real and imaginary samples of the scaled Hilbert Transformed wavelet. The "real" portion of the convolution results are then returned to the main program where they are saved in a text file are called: *"scale_filtered_cochleagram_half.txt"* and *"scale_filtered_cochleagram_one.txt,"* respectively. Each file has the same dimensions as the original cochleagram. **Figures 10 and 11** show the result from scale filtering the original cochleagram at the two bandwidth settings.

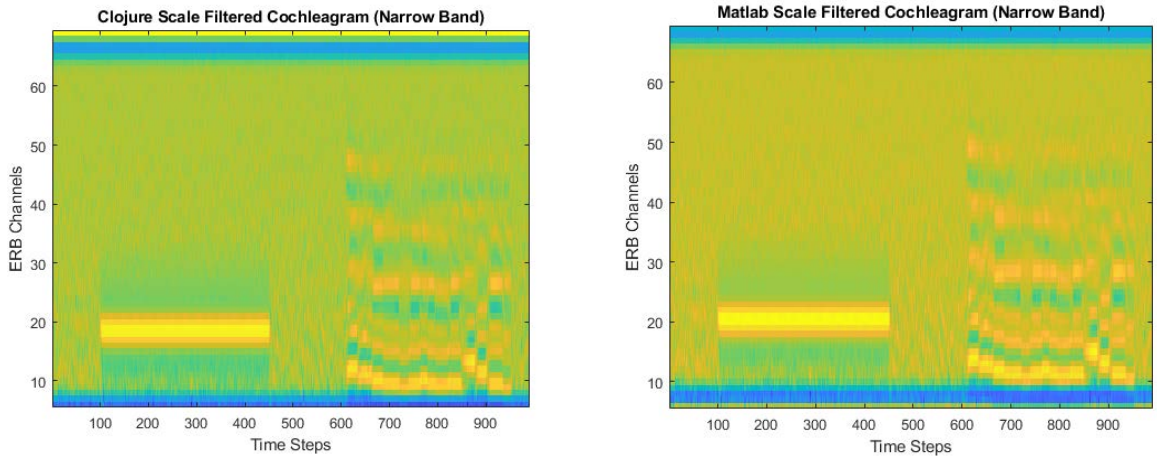


Figure 10. Scale Filtered Cochleagram #1, where the "scale-value" = 1.0 cycles/octave. ERB Channels v Time Segments (0.01 seconds each). The color represents the power of the signals. The figure shows a tone that occurs for 3.5 seconds (time segments 100-450), and saxophone notes for 3.5 seconds (time segments 600-950), separated by 1.5 seconds (time segments 450-600). The left-hand surface is a product of the Clojure code, the right-hand surface is from the MATLAB® code. The difference in the appearance of the two surfaces is due to the different dynamic ranges of the power levels.

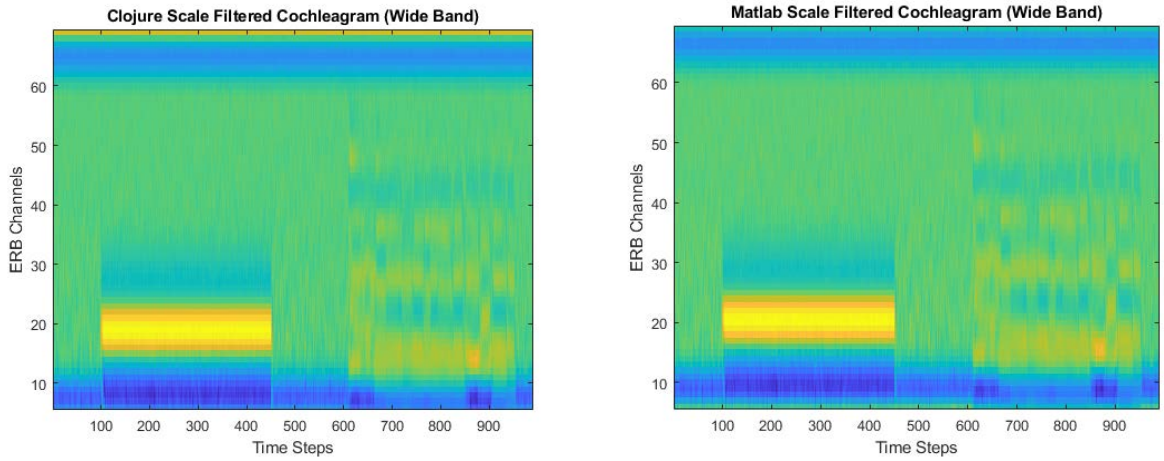


Figure 11. Scale Filtered Cochleagram #2, where the "scale-value" = 0.5 cycles/octave. ERB Channels v Time Segments (0.01 seconds each). The color represents the power of the signals. The figure shows a tone that occurs for 3.5 seconds (time segments 100-450), and saxophone notes for 3.5 seconds (time segments 600-950), separated by 1.5 seconds (time segments 450-600). The left-hand surface is a product of the Clojure code, the right-hand surface is from the MATLAB® code. The difference in the appearance of the two surfaces is due to the different dynamic ranges of the power levels.

C.3.5 Fundamental Frequency

Another feature that is used in the stream segregation step is fundamental frequency. The function "*casa-fund-freq-detect*" extracts the most prominent frequency that occurred in the cochleagram. It accomplishes this by performing an auto-correlation on time segments from each ERB channel of the original cochleagram. The auto-correlation is done in the frequency domain by first performing a forward FFT on both time series, multiplying them together, and then returning the results back into the time domain with an inverse FFT. Then each of the 64 short time segments of auto-correlated data are summed together. A spectrum analysis is then performed on the summed time series, again, using a forward FFT. The magnitudes are then computed. The vector that results is saved for this time segment,

and the process is repeated for all other time segments of the cochleagram. The function returns a 2D array to the main program that contains the magnitudes of the summed ERB channels' auto-correlated time series. It also saves this 2D matrix as a text file called *"summed_erb_auto_correlation_magnitudes.txt."* (This process requires a significant amount of time).

Because the density of the resulting spectral analysis is much greater than that of the original 64 ERB channels, the main program calls "*casa-downsample-fund-freq-results*" to down sample the densely populated magnitudes in the frequency dimension to 64 frequency samples, centered at the ERB channels center frequencies.

If the parameter "*DO-PEAK-PICK*" is set to 1, in the main program, then the function "*casa-find-peaks,*" which is inside "*casa-fund-freq-detect,*" is used to pick the index where the highest and next highest spectral magnitudes occur. These two values are saved in a 2D array, and also to a text file called: "*primary_secondary_fund_freq.txt.*"

NOTE: Because the time required to compute the auto-correlations is long, the flag *"DO-AUTO-CORRELATION-FLAG" can be* set in the main program, to avoid the compute intensive auto-correlation task. If this flag is set, then the file *"summed_erb_auto_correlation_magnitudes.txt"* is read prior to the primary and secondary peak picking logic, instead of having to wait for the auto-correlations to be computed. **Figure 12** shows the results of finding the fundamental frequency.

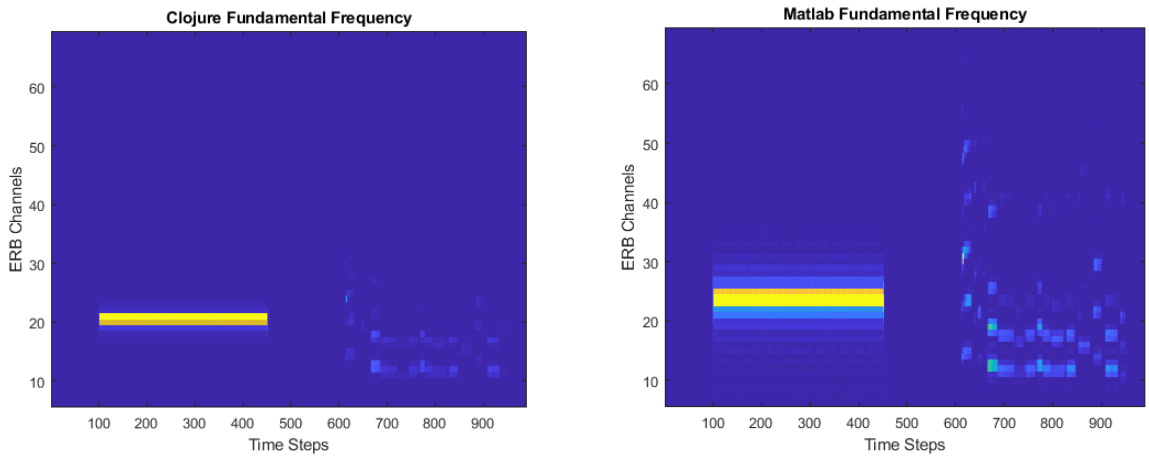


Figure 12. Fundamental Frequency. ERB Channels v Time Segments (0.01 seconds each). The color represents the power of the signal. The figure shows a tone that occurs for 3.5 seconds (time segments 100-450), and saxophone notes for 3.5 seconds (time segments 600-950), separated by 1.5 seconds (time segments 450-600). Note that the predominant frequency was centered around the constant tone and not the saxophone notes. The left-hand surface is a product of the Clojure code, the right-hand surface is from the MATLAB® code. The difference in the appearance of the two surfaces is due to the different dynamic ranges of the power levels.

C.3.6 Onsets and Offsets

Onsets and offsets are two features that can be fed to the stream segregation operation. To take advantage of these features, the function *"casa-onsets-offsets-separated" searches* for times, within the cochleagram, when sound first appears and then disappears, i.e. pairs of onsets and offsets. It does so by applying a Gaussian filter to each ERB channel's time segments, and computing the first positive and

negative derivative of the resulting amplitudes. The width of the Gaussian filter is set by the parameter inside this function ("*NUM-GAUSS-SAMPLES*" = 50, or 0.5 seconds). Positive derivative values correspond to when energy first arrives (onset), while the negative derivatives indicate when energy is lost (offset). If onsets precede offsets, a pair is identified and the time segment index of each is saved within its own file. One file is called *"onset.txt,"* while the other is called "*offset.txt.*" Each one is written as a vector that can be input to the clustering function. **Figures 13 and 14** show the result from finding the onset and offset indices, respectively, from the cochleagram. The algorithm employed in computing the onsets an offsets, differed slightly between the Clojure and MATLAB® codes, which likely degraded a more favorable stream segregation comparison.

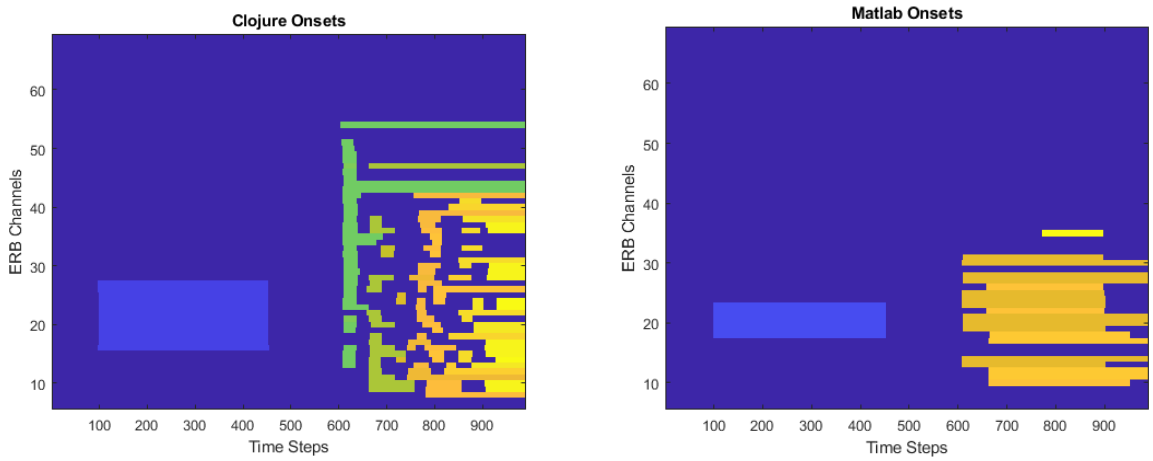


Figure 13. Onsets. ERB Channels v Time Segments (0.01 seconds each). The figure shows a tone that occurs for 3.5 seconds (time segments 100-450), and saxophone notes for 3.5 seconds (time segments 600-950), separated by 1.5 seconds (time segments 450-600). The onset time segment indices are color coded. Blue equates to early time segment indices while red colored values occur late in time. The left-hand surface is a product of the Clojure code, the right-hand surface is from the MATLAB® code.

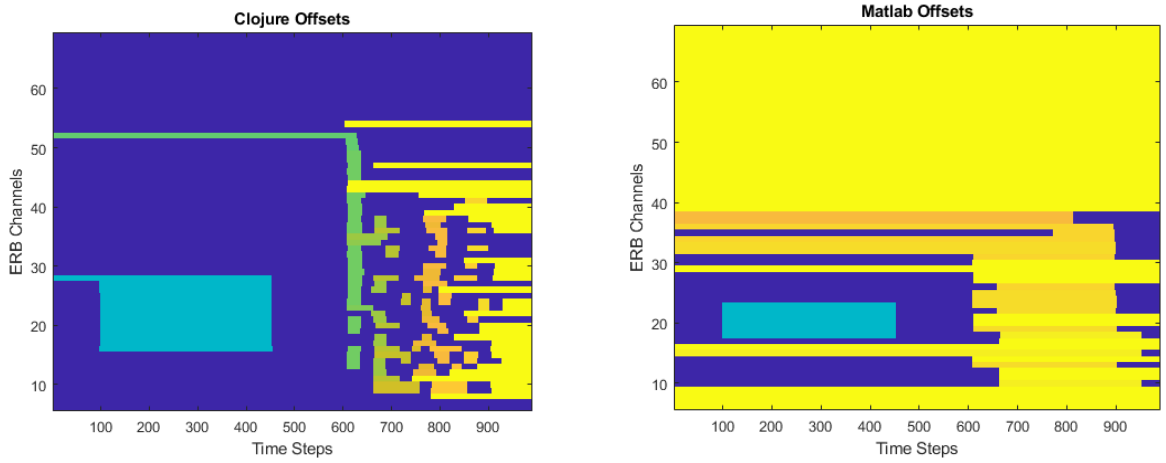


Figure 14. Offsets. ERB Channels v Time Segments (0.01 seconds each). The figure shows a tone that occurs for 3.5 seconds (time segments 100-450), and saxophone notes for 3.5 seconds (time segments 600-950), separated by 1.5 seconds (time segments 450-600). The offset time segment indices are color coded. Green/blue colors equate to early time segment indices while red/orange colored values occur later in time.

C.3.7: Rate Filter

Note: "*casa-rate-filter*" is included in the software release, but it is **NOT** recommended to be used in the stream segregation. Therefore, it is recommended **NOT** to set the "DO-RATE-FILTER" flag to "1" in the main program. This is due to the fact that it requires too much time to run the rate filter and the results are not known to contribute value to the stream segregation process.

C.4 Stream Segregation (Clustering)

The stream segregation phase is the third major area of functionality, and it follows the feature extraction stage of the processing. It consists of using a Gaussian Mixture Model (GMM) to cluster the features found (above) into groups that share some commonality in their attributes. Prior to being fed to the GMM algorithm, a feature-specific mask is applied to each feature "map." This is done within the main program. (*A separate function should be built to do this in the future*). All observations from all features are placed into a single large 2D matrix, upon which the GMM algorithm will operate. The size of this 2D array is:

*number of observations = 64 ERB channels * 990 time steps*

number of observations * *number of features*, or ((64 * 990) * 7 = 443520) elements

C.4.1 Gaussian Mixture Model

The main program calls the Gaussian Mixture Model (GMM) function, "*casa-gmm-3-7*" followed by some post-processing functions that modify the cochleagram based on the cluster statistics that are returned from the GMM. This function invokes the GMM algorithm to separate the streams. It accepts up to 7 features (see **Figures 8 - 14** above), and is currently configured to produce 3 streams (clusters). The features include:

1. Cochleagram (power values)
2. ITD time lags (azimuth expressed in units of time samples)
3. Scale Filtered #1 Cochleagram (power values)
4. Scale Filtered #2 Cochleagram (power values)
5. Fundamental Frequency (ERB Center Frequency in Hz)
6. Onsets (time slice indices)
7. Offsets (time slice indices)

The main program first reads the features from intermediate files that have been saved earlier by the feature extraction functions. It then applies masks to the feature maps in order to reduce the background noise that would "confuse" the clustering operation. This in effect increases the Signal to Noise Ratio (SNR) of the data upon which the GMM operates. A binary mask is a 2D array that is comprised of 1s and 0s and has the same dimensions as the original cochleagram. A mask is applied by multiplying each element of the mask by its corresponding element of the feature map. The various types of masks and the features to which they are applied are described next.

A binary mask, derived from thresholding the cochleagram power values, is applied to the azimuth (ITD) feature map. The cochleagram is clipped in the main program such that its dynamic range is set to

"*COCHLEAR-CLIP-VALUE*" (= 30.0). Therefore, all values in the mask that exceed the clipping power level are set to 1.0, while those that fall below 0.0 are set to 0. The 2 scale-filtered cochleagrams are used to produce 2 other binary masks, based on 2 threshold values that correspond to the respective scale-filtered cochleagram. The threshold value used is called "*SCALE-FILTER-CLIP-VALUE*" and it is set to 60.0 and 50. 0 for each scale filtered cochleagram, respectively. The second threshold value is called "*SCALE-FILTER-THRESHOLD,*" and its value is set to 0.33 for both scale filtered cochleagrams. The resulting masks are combined into one and applied to the fundamental frequency, onset, and offset feature maps.

Once the features have been prepared, the main program builds a large 2D matrix of ALL "observations" that will be input to the GMM function, where an "observation" is defined to be a single time/frequency sample. Each feature's observation consists of a sample from one ERB channel and one time segment. In the demonstration, there are 64 ERB channels and 990 time segments). The parameter called "*NUM-FEATURES-REQUESTED,*" in the main program, is set to seven. Therefore, the input matrix contains 64 x 990 x 7 (or 443520) elements. The number of clusters reflects the number of streams that are expected to contribute to mixture of sounds. The software is currently configured to output 3 of them (see "*NUM-CLUSTERS*" in the main program). Once this large 2D matrix has been formed, it is fed to the GMM function.

GMMs are formed using the Expectation-Maximization (EM) algorithm, which iterates through "Expectation" and "Maximization" steps until the iterations converge on an optimal set of means and covariance matrices. The GMM algorithm starts with the "*Expectation*" step, which requires a set of mean values, taken from each feature map, for each cluster to be identified to start the processing. The selection of the initial "mean" values is arbitrary. Due to the inefficiencies of the GMM algorithm's implementation in Clojure, specific values (from each feature map 2D matrix) were chosen as the initial means, in order to minimize the amount of time needed to reach convergence. The initial means selected for demonstration were selected based on a priori knowledge of the structure of the cochleagram. That is, there were two sources of sound, i.e. a single tone followed by some saxophone notes. For this demonstration the number of clusters was manually set to 3; where the first two clusters corresponded to specific sound sources, and the third one corresponded to the background noise. The specific indices of these initial means were chosen from the cochleagram's MATLAB® figure, and hardwired into this function.

After the initial means have been defined, the GMM computes the covariance among the features. The function then iterates over clusters (outer loop) and observations (inner loop). Thus, for each observation from each feature, the corresponding cluster mean is removed, and along with the covariance for the cluster, a "Gaussian Distribution" is computed. The "PDF" value returned indicates the probability that a particular set of features at one observation belongs to a particular cluster. From relevant literature, this probability is also known as the "*responsibility*" (Hasan et al, 2009). Next, the initial probability of each cluster is defined. To start, each cluster is assigned an equal probability ("1/K"; where "K = 3" is the number of clusters). These values are then multiplied by each feature's observation to yield the initial weight that each sample is assigned to a particular cluster.

Once this has been completed for all clusters and observations, the GMM algorithm enters a loop that starts with the "*Maximization*" step. Here, the probabilities computed previously are used to compute a new mean and covariance values of the 7 features for the observations in each cluster. They are also used to calculate a new distribution of cluster probabilities, whose sum must always add up to 1. These new statistics are then fed to the "*Expectation*" step, where a new probability is found that determines whether a feature at a particular observation belongs to particular cluster.

The cycle continues until the difference in the maximum likelihood, between the new probabilities and the previous ones, is less than some predetermined threshold, or a pre-defined number of iterations. The number of iterations is limited by the following three parameters which are set in the main program:

- Minimum number of iterations ("*MIN-NUM-GMM-EM-ITERATIONS*") = 6
- Maximum number of iterations ("*MAX-NUM-GMM-EM-ITERATIONS*") = 10
- Current Threshold ("*GMM-THRESHOLD*") = 0.99

The function performs the "E-M" cycle until the minimum number of iterations is reached. At this time, the difference between the most recent maximum likelihood value is compared to the previous one. If the difference is less than the "Current Threshold" parameter, i.e. convergence is reached, then the function returns its results to the main program. If the difference continues to exceed the "Current Threshold," then the function iterates until it exceeds the maximum number of iterations before it returns its values to the main program. The returned values include the final cluster means, covariances, and probability that each observation lies within a cluster. These quantities are contained within a single vector as three aforementioned matrices. This vector is called "*gmm-out.*" The means and probabilities (responsibilities) are held in 2D arrays, while the covariances are saved in a 3D array

The main program then loops over the three clusters that the GMM function produced, performing post-processing operations to finally yield the separated streams. The next several sections describe the subsequent individual post clustering processes.

C.4.2 Chi Square Threshold

The function "*casa-evaluate-chi-square*" is used in building the masks, based on the clusters returned from the GMM stream segregation operation that was applied to the original cochleagram. Based on a confidence level and the number of degrees of freedom associated with the size of a sample population, a Chi Square probability value can be computed that an observation will lie inside or outside of some distance threshold from a population's mean. Here, this threshold value is used to determine whether a time/frequency sample from the original cochleagram will be included in a particular cluster or not by the next function ("*casa-mahalanobis-distance*"). The current confidence level is set at 0.05 (95%), and the number of degrees of freedom is set at "*number-of-input-features-check-1*" (or 7 – 1 = 6).

C.4.3 Mahalanobis Distance Threshold:

As stated above, the GMM produces 3 clusters, each with a mean for the 7 input features (μ) and a covariance matrix associated with an observation (time/frequency sample). Each observation's vector of

feature values (x) is combined into a single value, by virtue of the cluster means and covariance (C). The distance that this combination of feature values is displaced from the cluster's mean is determined by the *Mahalanobis* distance formula:

$$M = \sqrt{(x - \mu)^T * C^{-1} * (x - \mu)} \quad \text{Eq. (4)}$$

The Clojure function "*casa-mahalanobis-distance"* invokes this equation. This distance is returned for each feature's observation, and they are stored as a vector for each cluster.

C.4.4 Build Cluster Mask

The function "*casa-build-vmask*" uses the *Mahalanobis* distance to determine whether an observation in the cochleagram is preserved or not. It does this by building a mask, based on the Mahalanobis distance that was computed at each cochlear frequency and time segment index. If the Mahalanobis distance for an observation, at a particular frequency and time segment index, fell within the threshold distance, then the mask value is assigned a value of "1." If the Mahalanobis distance fell outside of the distance threshold, then the mask value, for a particular frequency and time segment index, is assigned a value of 0. This function returns a vector of these binary values. One of these vectors is returned for each cluster produced by the GMM function.

C.4.5 Apply Cluster Mask

The function "*casa-apply-vmask*" applies the previous function's output vector to the original cochleagram. The result of multiplying each element of the vector by the cochleagram's observation value, at the corresponding frequency and time segment index, can be seen in **Figures 15, 16, and, 17** as generated with MATLAB®. (These are duplicates of **Figures 5, 6,** and **7** in **Section 3**). The numerical values that went into these figures are held in three text files that contain the separated streams. The files are written to the same directory in which the Clojure source code resides. Even though the number of elements in each file is the same as in the cochleagram, the files are written as a vector. Their file names are:

1. "*casa_binary_masked_cochleagram_cluster_1.txt" (cluster/stream #1: tone)*
2. *"casa_binary_masked_cochleagram_cluster_2.txt" (cluster/stream #2: saxophone)*
3. *"casa_binary_masked_cochleagram_cluster_3.txt" (cluster/stream #3: background noise)*

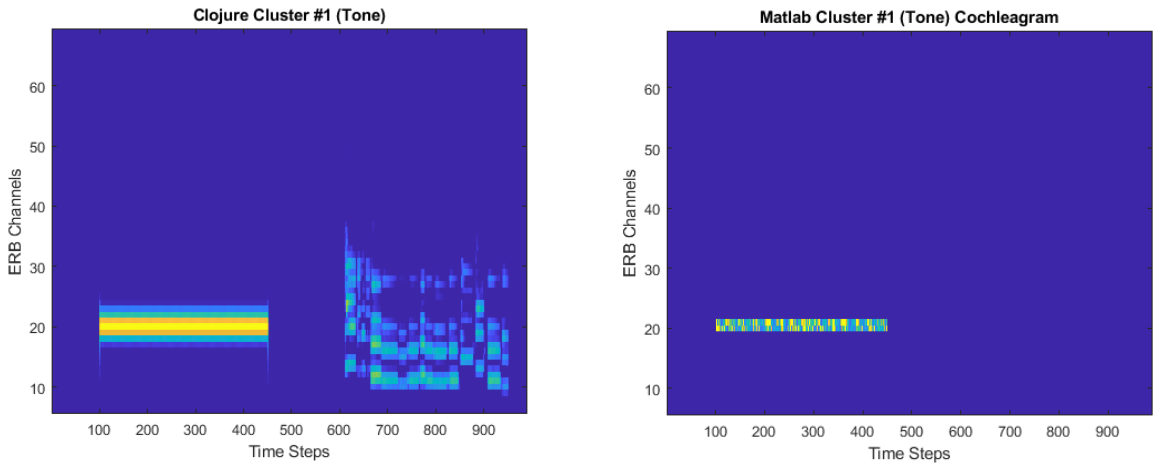


Figure 15. Cochleagram results after the Mahalanobis distance between each observation and the cluster #1's mean has been used as a threshold to mask the original cochleagram's power values. The color represents the power of the signals. Note that the tone between time segments 100 and 450 has been accentuated with respect to the saxophone notes that appear between time segments 600 and 950. This is due to the fact that the original means for Cluster #1 were chosen within the tone (signal) region of the original cochleagram. This illustrates the ability of the Gaussian Mixture Model (GMM) to separate disparate streams when there are an equal number of clusters and streams. The left-hand surface is a product of the Clojure code, the right-hand surface is from the MATLAB® code. The difference in the appearance of the two surfaces is due to the different dynamic ranges of the power levels of each cochleagram.

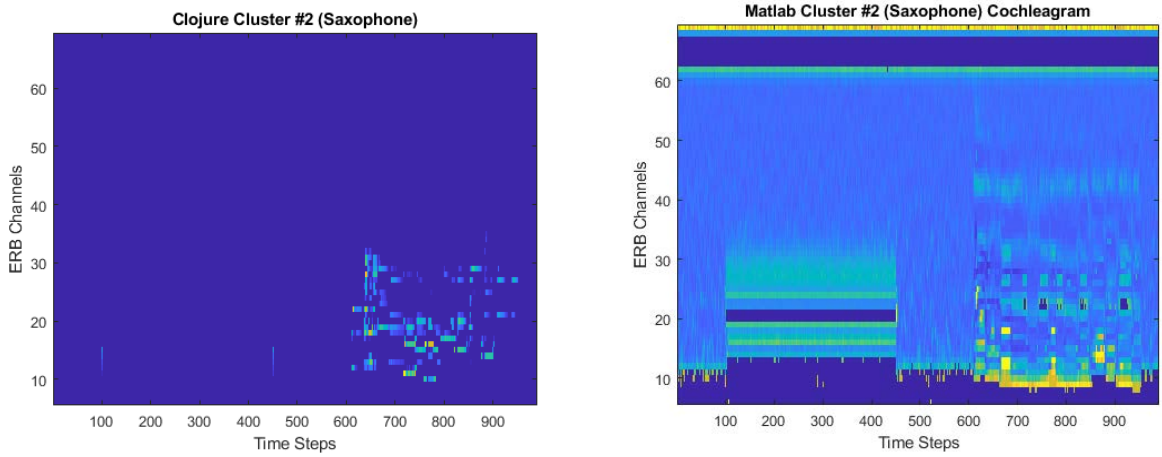


Figure 16. Cochleagram results after the Mahalanobis distance between each observation and the cluster #2's mean has been used as a threshold to mask the original cochleagram's power values. The color represents the power of the signals. Note that the tone between time segments 100 and 450 has been attenuated with respect to the saxophone notes that appear between time segments 600 and 950. This is due to the fact that the original means for Cluster #2 were chosen within the saxophone notes (signal) region of the original cochleagram. This illustrates the ability of the Gaussian Mixture Model (GMM) to separate disparate streams when there are an equal number of clusters and streams. The left-hand surface is a product of the Clojure code, the right-hand surface is from the MATLAB® code. The difference in the appearance of the two surfaces is due to the different dynamic ranges of the power levels of each cochleagram.

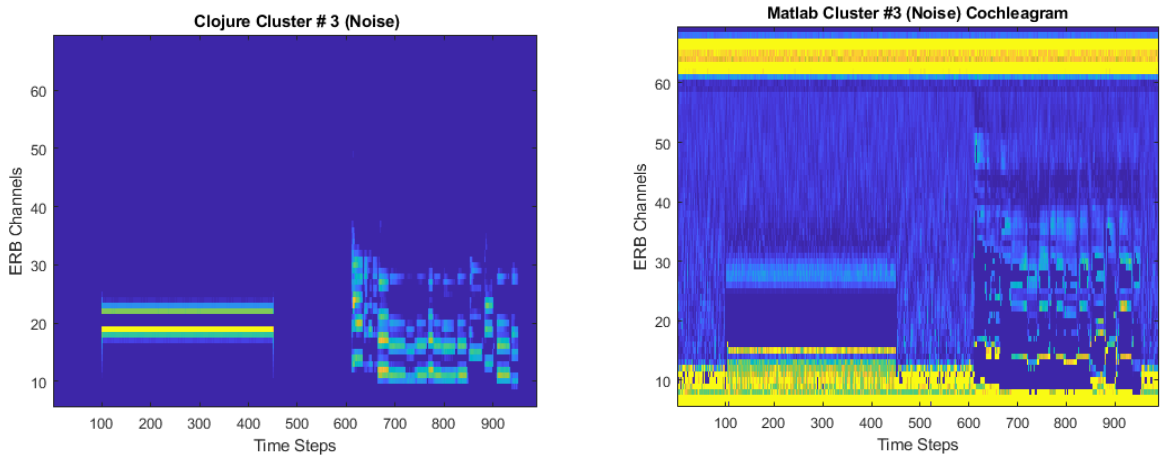


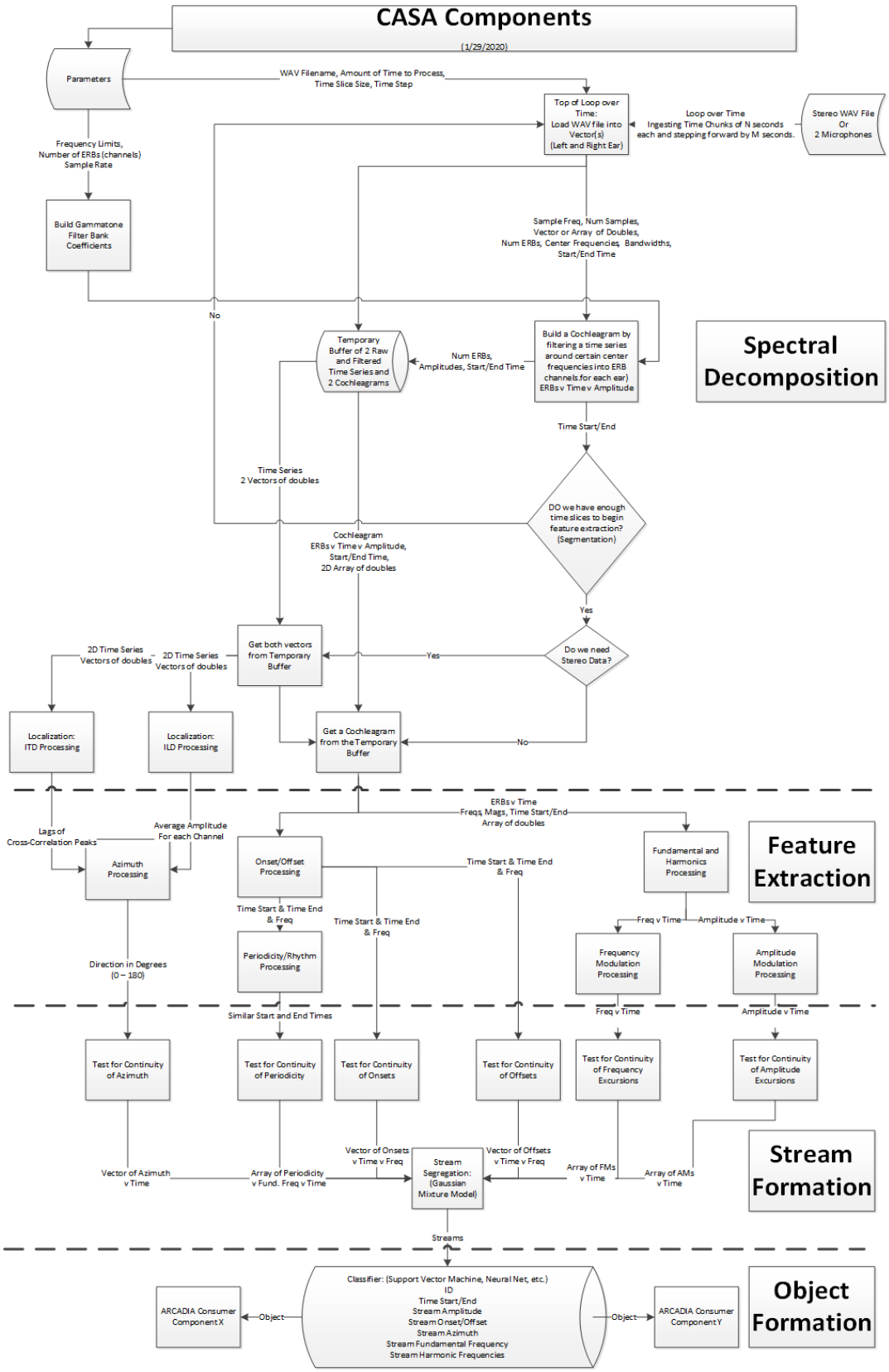
Figure 17. Cochleagram results after the Mahalanobis distance between each observation and the cluster #3's mean has been used as a threshold to mask the original cochleagram's power values. The color represents the power of the signals. Note that the tone between time segments 100 and 450 and the saxophone notes that appear between time segments 600 and 950 have not been affected by the masking or have been affected by the masking about equally. This is due to the fact that the original means for Cluster #3 were chosen within a noise region of the original cochleagram. This illustrates the ability of the Gaussian Mixture Model (GMM) to separate disparate streams when there are more clusters than there are actual streams. The left-hand surface is a product of the Clojure code, the right-hand surface is from the MATLAB® code. The difference in the appearance of the two surfaces is due to the different dynamic ranges of the power levels of each cochleagram.

Additionally, if one simply multiplies the probability that the values in the cochleagram belong to a particular cluster, one obtains a surface for each cluster. For completeness, and for further evaluation, these surfaces are saved in the following files:

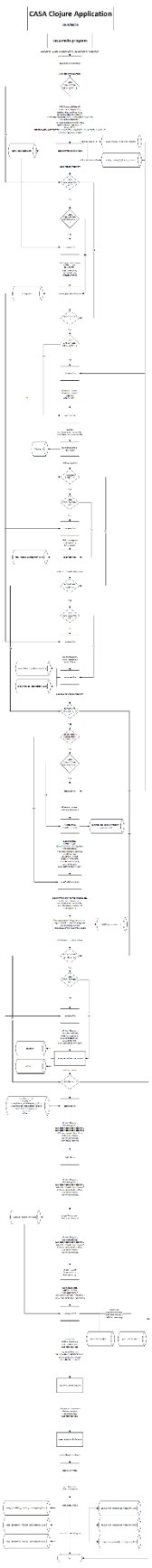
4. "*casa_probability_product_cochleagram_cluster_1.txt" (cluster/stream #1: tone)*
5. *"casa_probability_product _cochleagram_cluster_2.txt" (cluster/stream #2: saxophone)*
6. *"casa_ probability_product _cochleagram_cluster_3.txt" (cluster/stream #3: background noise)*

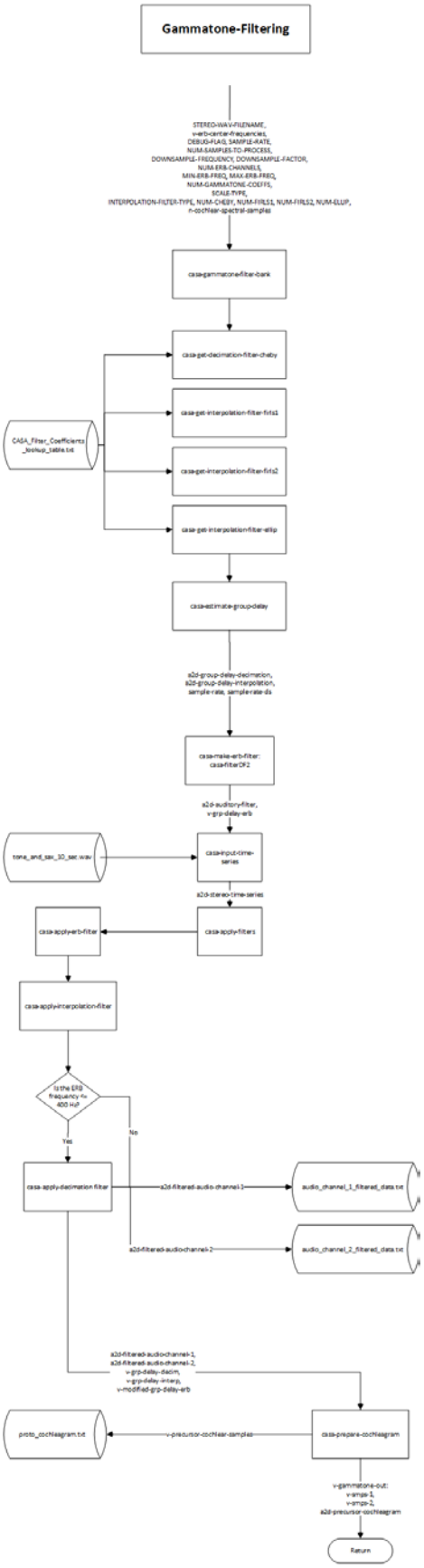
Appendix D: CASA Flowcharts

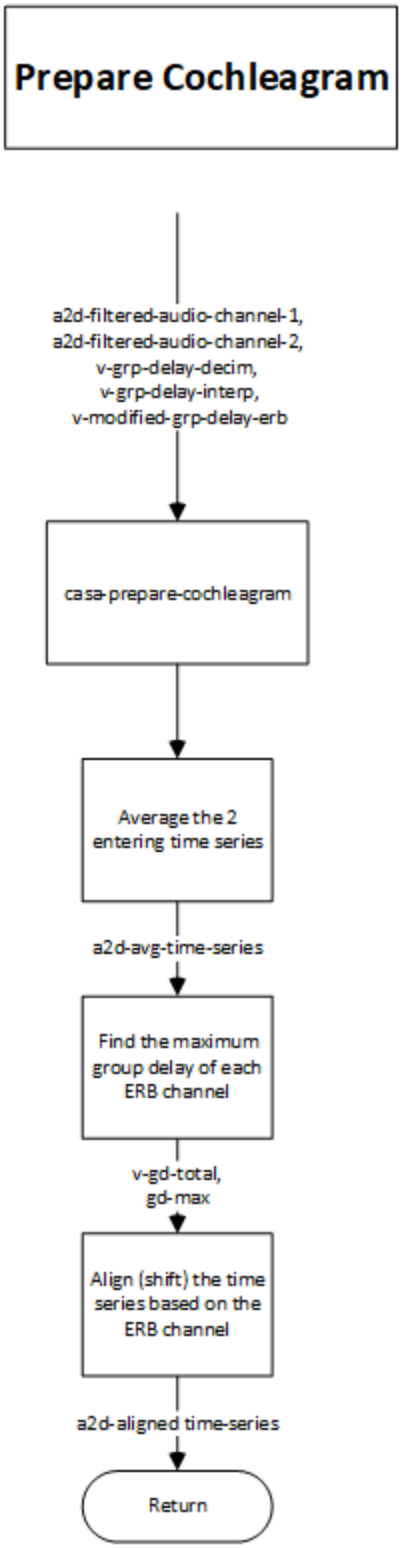
The following flowcharts illustrate the functional and data flow that comprise the CASA Clojure software application. The figures start with the overall design, followed by the individual processing components as they are encountered in the processing sequence, depicting greater detail in the implementation.

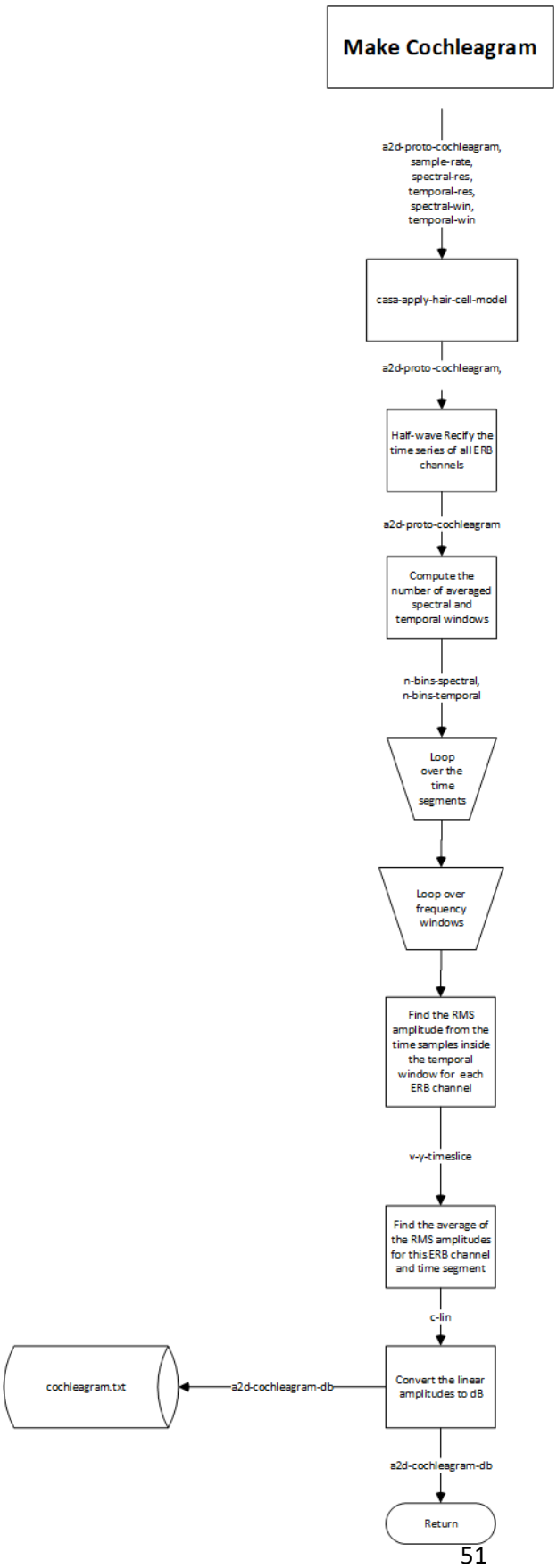


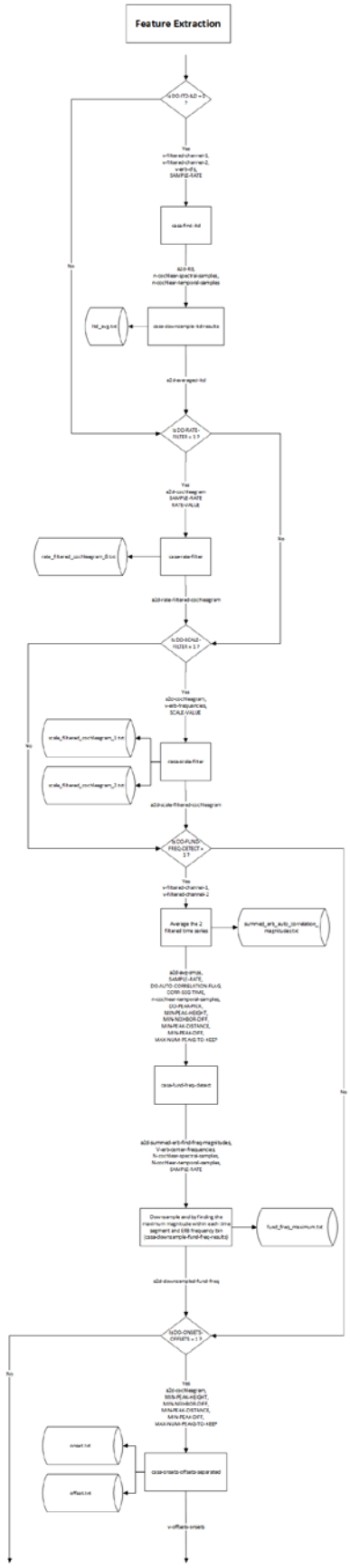
Overall flowchart of the CASA "front-end" to the ARCADIA attention-based Cognitive Architecture

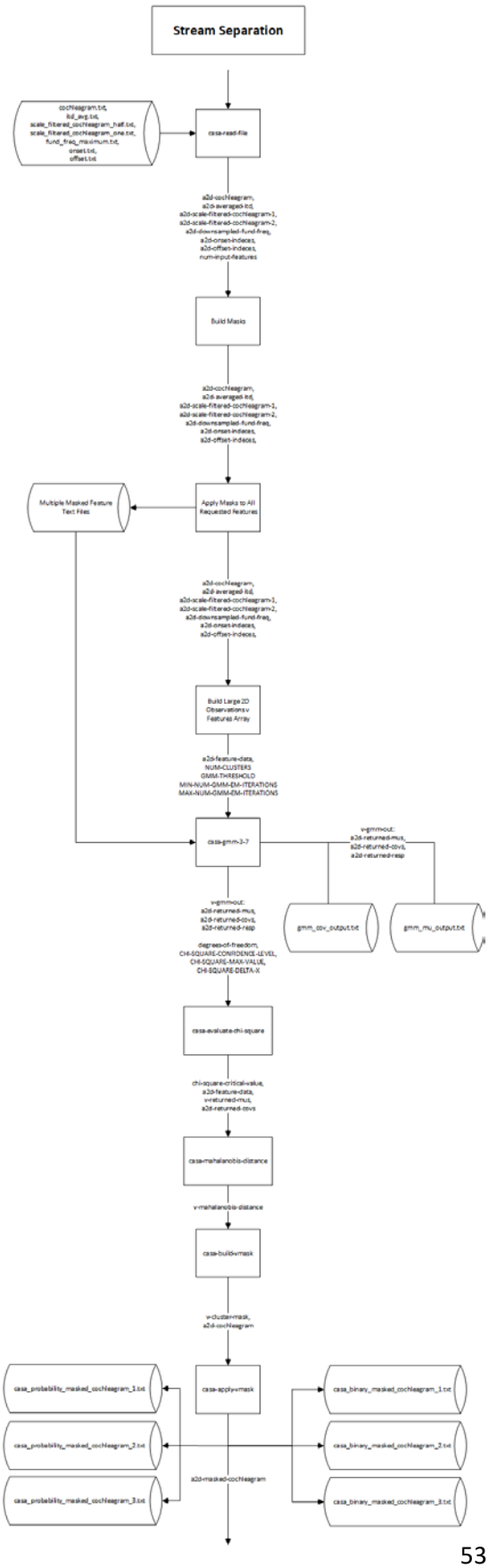

CASA Clojure Application

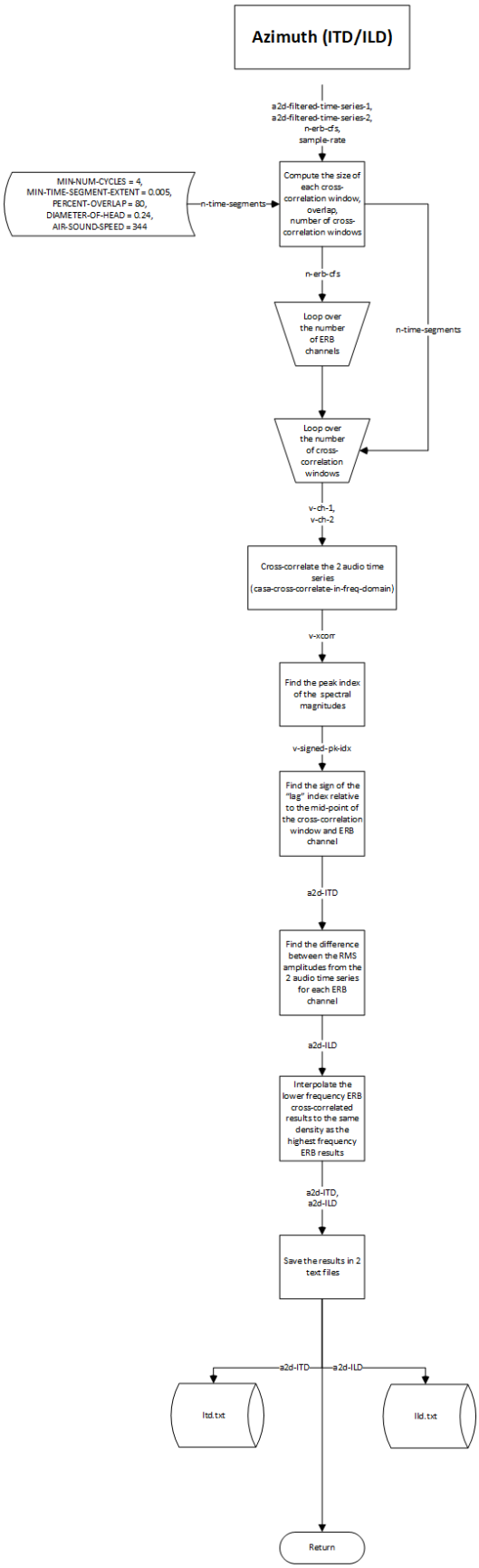

Gammatone-Filtering
STEREO-WAV-FILENAME,
v-erb-center-frequencies,
DEBUG-FLAG, SAMPLE-RATE,
NUM-SAMPLES-TO-PROCESS,
DOWNSAMPLE-FREQUENCY, DOWNSAMPLE-FACTOR,
NUM-ERB-CHANNELS,
MIN-ERB-FREQ, MAX-ERB-FREQ,
NUM-GAMMATONE-COEFFS,
SCALE-TYPE,
INTERPOLATION-FILTER-TYPE, NUM-CHEBY, NUM-FIRLS1, NUM-FIRLS2, NUM-ELLIP,
n-cochlear-spectral-samples
casa-gammatone-filter-bank
casa-get-decimation-filter-cheby
casa-get-interpolation-filter-firls1
casa-get-interpolation-filter-firls2
casa-get-interpolation-filter-ellip
CASA_Filter_Coefficients_lookup_table.txt
casa-estimate-group-delay
a2d-group-delay-decimation,
a2d-group-delay-interpolation,
sample-rate, sample-rate-ds
casa-make-erb-filter:
casa-filterDF2
a2d-auditory-filter,
v-grp-delay-erb
tone_and_sax_10_ser.wav
casa-input-time-series
a2d-stereo-time-series
casa-apply-erb-filter
casa-apply-filters
casa-apply-interpolation-filter
Is the ERB frequency <= 400 Hz?
Yes
No
casa-apply-decimation filter
a2d-filtered-audio-channel-1
audio_channel_1_filtered_data.txt
a2d-filtered-audio-channel-2
audio_channel_2_filtered_data.txt
a2d-filtered-audio-channel-1,
a2d-filtered-audio-channel-2,
v-grp-delay-decm,
v-grp-delay-interp,
v-modified-grp-delay-erb
proto_cochleagram.txt
v-precursor-cochlear-samples
casa-prepare-cochleagram
v-gammatone-out:
v-smpl-1,
v-smpl-2,
a2d-precursor-cochleagram
Return

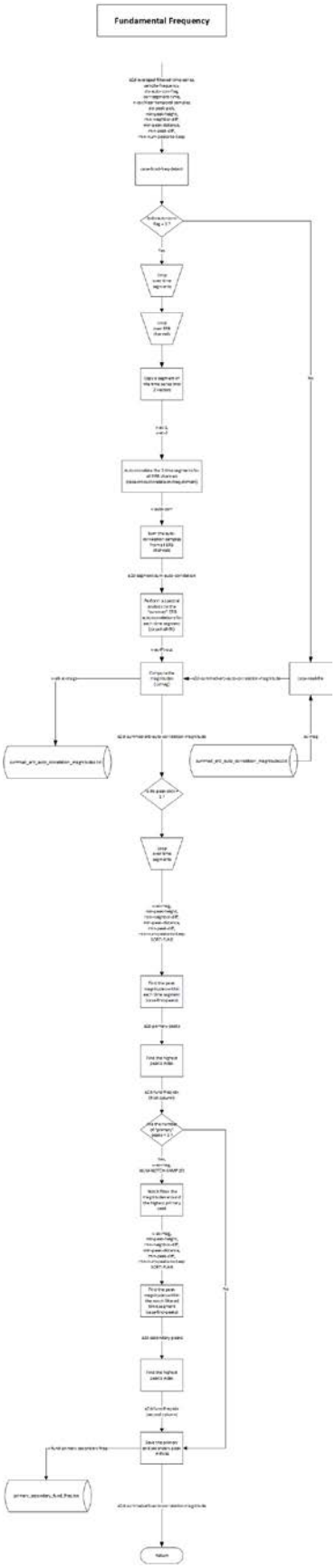

Prepare Cochleagram
a2d-filtered-audio-channel-1,
a2d-filtered-audio-channel-2,
v-grp-delay-decim,
v-grp-delay-interp,
v-modified-grp-delay-erb
casa-prepare-cochleagram
Average the 2 entering time series
a2d-avg-time-series
Find the maximum group delay of each ERB channel
v-gd-total,
gd-max
Align (shift) the time series based on the ERB channel
a2d-aligned time-series
Return


Make Cochleagram
a2d-proto-cochleagram,
sample-rate,
spectral-res,
temporal-res,
spectral-win,
temporal-win
casa-apply-hair-cell-model
a2d-proto-cochleagram,
Half-wave Recify the time series of all ERB channels
a2d-proto-cochleagram
Compute the number of averaged spectral and temporal windows
n-bins-spectral,
n-bins-temporal
Loop over the time segments
Loop over frequency windows
Find the RMS amplitude from the time samples inside the temporal window for each ERB channel
v-y-timeslice
Find the average of the RMS amplitudes for this ERB channel and time segment
c-lin
Convert the linear amplitudes to dB
a2d-cochleagram-db
cochleagram.txt
a2d-cochleagram-db
Return

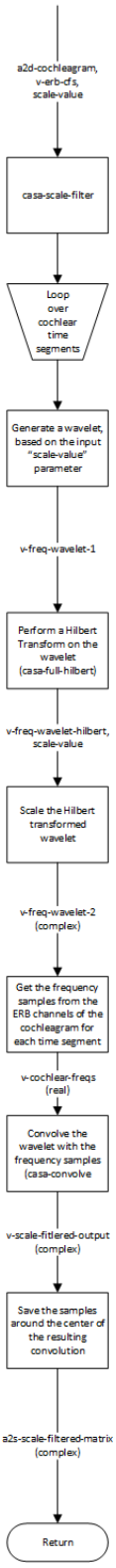

Feature Extraction

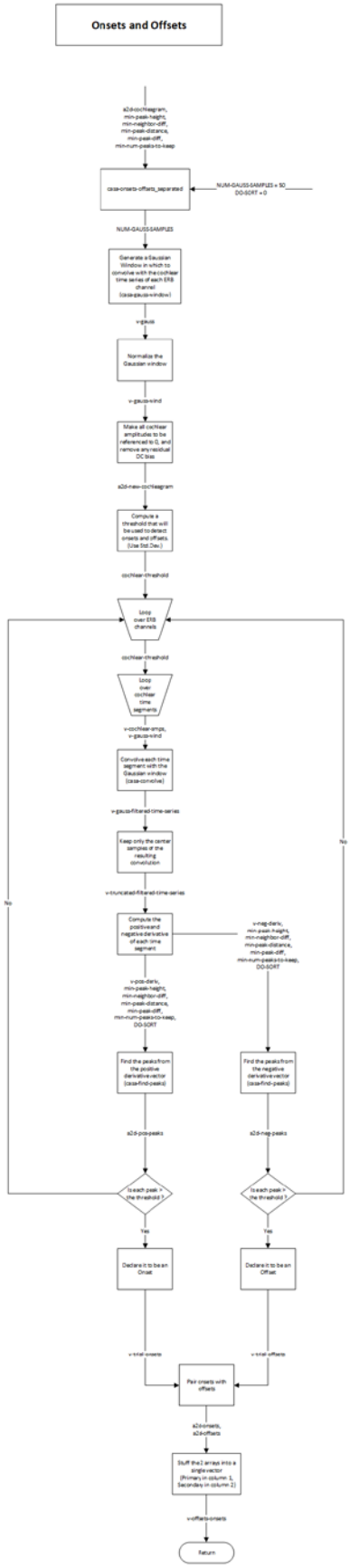

Stream Separation
cochleagram.txt,
fcf_avg.txt,
scale_filtered_cochleagram_half.txt,
scale_filtered_cochleagram_one.txt,
fund_freq_max=num.txt,
onset.txt,
offset.txt
casa-read-file
a2d-cochleagram,
a2d-averaged-fcf,
a2d-scale-filtered-cochleagram-1,
a2d-scale-filtered-cochleagram-2,
a2d-downsampled-fund-freq,
a2d-onset-indeces,
a2d-offset-indeces,
num-input-features
Build Masks
a2d-cochleagram,
a2d-averaged-fcf,
a2d-scale-filtered-cochleagram-1,
a2d-scale-filtered-cochleagram-2,
a2d-downsampled-fund-freq,
a2d-onset-indeces,
a2d-offset-indeces,
Multiple Masked Feature Text Files
Apply Masks to All Requested Features
a2d-cochleagram,
a2d-averaged-fcf,
a2d-scale-filtered-cochleagram-1,
a2d-scale-filtered-cochleagram-2,
a2d-downsampled-fund-freq,
a2d-onset-indeces,
a2d-offset-indeces,
Build Large 2D Observations v Features Array
a2d-feature-data,
NUM-CLUSTERS
GMM-THRESHOLD
MIN-NUM-GMM-EM-ITERATIONS
MAX-NUM-GMM-EM-ITERATIONS
casa-gmm-3-7
v-gmm-out:
a2d-returned-mus,
a2d-returned-covs,
a2d-returned-resp
gmm_cov_output.txt
gmm_mu_output.txt
v-gmm-out:
a2d-returned-mus,
a2d-returned-covs,
a2d-returned-resp
degrees-of-freedom,
CHI-SQUARE-CONFIDENCE-LEVEL,
CHI-SQUARE-MAX-VALUE,
CHI-SQUARE-DELTA-X
casa-evaluate-chi-square
chi-square-critical-value,
a2d-feature-data,
v-returned-mus,
a2d-returned-covs
casa-mahalanobis-distance
v-mahalanobis-distance
casa-build-v-mask
v-cluster-mask,
a2d-cochleagram
casa_probability_masked_cochleagram_1.txt
casa_probability_masked_cochleagram_2.txt
casa_probability_masked_cochleagram_3.txt
casa-apply-v-mask
casa_binary_masked_cochleagram_1.txt
casa_binary_masked_cochleagram_2.txt
casa_binary_masked_cochleagram_3.txt
a2d-masked-cochleagram

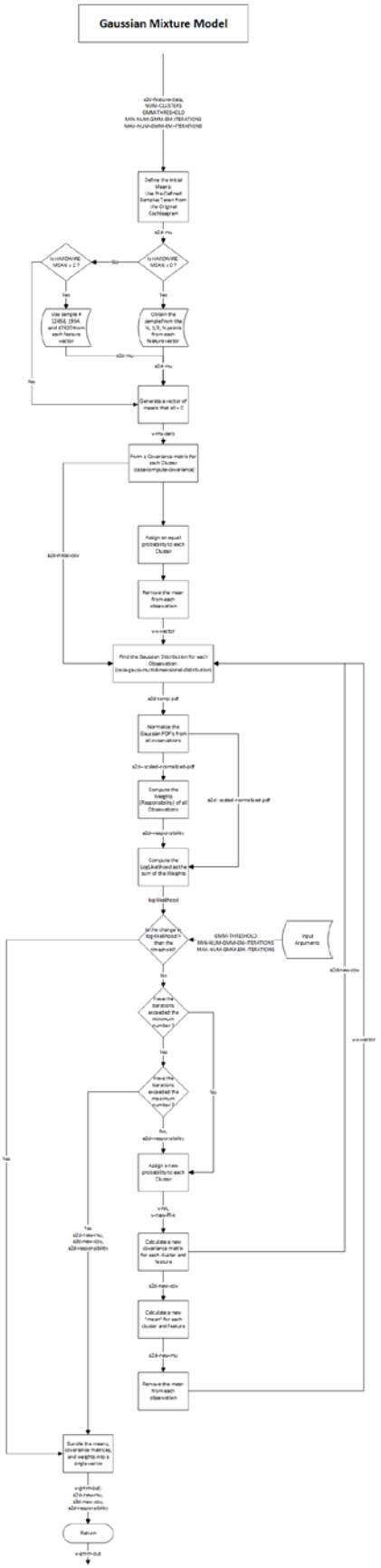

Azimuth (ITD/ILD)
a2d-filtered-time-series-1,
a2d-filtered-time-series-2,
n-erb-cfs,
sample-rate
MIN-NUM-CYCLES = 4,
MIN-TIME-SEGMENT-EXTENT = 0.005,
PERCENT-OVERLAP = 80,
DIAMETER-OF-HEAD = 0.24,
AIR-SOUND-SPEED = 344
n-time-segments
Compute the size of each cross-correlation window, overlap, number of cross-correlation windows
n-erb-cfs
Loop over the number of ERB channels
n-time-segments
Loop over the number of cross-correlation windows
v-ch-1,
v-ch-2
Cross-correlate the 2 audio time series (casa-cross-correlate-in-freq-domain)
v-xcorr
Find the peak index of the spectral magnitudes
v-signed-pk-idx
Find the sign of the "lag" index relative to the mid-point of the cross-correlation window and ERB channel
a2d-ITD
Find the difference between the RMS amplitudes from the 2 audio time series for each ERB channel
a2d-ILD
Interpolate the lower frequency ERB cross-correlated results to the same density as the highest frequency ERB results
a2d-ITD,
a2d-ILD
Save the results in 2 text files
a2d-ITD
a2d-ILD
itd.txt
ild.txt
Return


Fundamental Frequency

Scale Filter
(Call this twice, once for each scale factor)




Onsets and Offsets
a2d-cochleagram, min-peak-height, min-neighbor-diff, min-peak-distance, min-peak-diff, min-num-peaks-to-keep
casa-onsets-offsets_separated
NUM-GAUSS-SAMPLES = 50
DO-SORT = 0
NUM-GAUSS-SAMPLES
Generate a Gaussian Window in which to convolve with the cochlear time series of each ERB channel (casa-gauss-window)
v-gauss
Normalize the Gaussian window
v-gauss-wind
Make all cochlear amplitudes to be referenced to 0, and remove any residual DC bias
a2d-new-cochleagram
Compute a threshold that will be used to detect onsets and offsets. (Use Std.Dev.)
cochlear-threshold
Loop over ERB channels
cochlear-threshold
Loop over cochlear time segments
v-cochlear-amps, v-gauss-wind
Convolve each time segment with the Gaussian window (casa-convolve)
v-gauss-filtered-time-series
Keep only the center samples of the resulting convolution
v-truncated-filtered-time-series
Compute the positive and negative derivative of each time segment
v-neg-deriv, min-peak-height, min-neighbor-diff, min-peak-distance, min-peak-diff, min-num-peaks-to-keep, DO-SORT
v-pos-deriv, min-peak-height, min-neighbor-diff, min-peak-distance, min-peak-diff, min-num-peaks-to-keep, DO-SORT
Find the peaks from the positive derivative vector (casa-find-peaks)
Find the peaks from the negative derivative vector (casa-find-peaks)
a2d-pos-peaks
a2d-neg-peaks
Is each peak > the threshold ?
Is each peak > the threshold ?
No
No
Yes
Yes
Declare it to be an Onset
Declare it to be an Offset
v-total-onsets
v-total-offsets
Pair onsets with offsets
a2d-onsets, a2d-offsets
Stuff the 2 arrays into a single vector (Primary in column 1, Secondary in column 2)
v-offsets-onsets
Return


Gaussian Mixture Model
a2d-feature-data,
NUM-CLUSTERS,
GMM-THRESHOLD,
MIN-NUM-GMM-EM-ITERATIONS,
MAX-NUM-GMM-EM-ITERATIONS
Define the Initial Means (Use the Predefined Samples Taken from the Original Cochleagram
a2d-mu
Is NUM-OBS MEAN < 2.5
Is NUM-OBS MEAN > 0.5
Yes
No
Yes
Use sample # 22652, 2304, and 67229 from each feature vector
Obtain the sample from the ¼, 1/3, ½ points from each feature vector
a2d-mu
a2d-mu
No
Generate a vector of means that all= 0
v-mu-zero
Form a Covariance matrix for each Cluster (assume equal covariance)
Assign an equal probability to each Cluster
a2d-initial-cov
Remove the mean from each observation
v-v-vector
Find the Gaussian Distribution for each Observation (scipy.stats.multivariate_normal.html#multivariate-normal-distribution)
a2d-temp-pdf
Normalize the Gaussian PDF's from all observations
a2d-scaled-normalized-pdf
Compute the Weights (Responsibility) of all Observations
a2d-scaled-normalized-pdf
a2d-responsibility
Compute the Log-Likelihood as the sum of the Weights
log-likelihood
Is the change in log-likelihood > than the threshold?
GMM-THRESHOLD
MIN-NUM-GMM-EM-ITERATIONS
MAX-NUM-GMM-EM-ITERATIONS
Input Arguments
a2d-new-cov
No
Have the iterations exceeded the maximum number?
Yes
Have the iterations exceeded the maximum number?
Yes
No
No
a2d-responsibility
Assign a new probability to each Cluster
v-mu
v-new-Phi
Yes
a2d-new-mu,
a2d-new-cov,
a2d-responsibility
Calculate a new Covariance matrix for each cluster and feature
v-v-vector
a2d-new-cov
Calculate a new "mean" for each cluster and feature
a2d-new-mu
Remove the mean from each observation
Bundle the means, covariance matrices, and weights into a single vector
v-gmm-out,
a2d-new-mu,
a3d-new-cov,
a2d-responsibility
Return
v-gmm-out