**ALGORITHM FOR AUTOMATED BEE HUMMINGBIRD SOUND RECOGNITION**

Rigas Simos

*The following report examines the implementation of a MATLAB algorithm designed to detect the distinct sound patterns from Bee Hummingbird calls. All of the design, implementation and testing of the following algorithm took place during a 3-month period, as part of the assessment for the Audio Signal Processing course at Johns Hopkins University. The programming language used to implement the bird recognition algorithm is MATLAB, suitable for processing audio signals.*

# Introduction

Hummingbirds are known to be one of the smallest in size group of birds out there. The bee hummingbird in its turn is considered the smallest known bird species to exist, reaching sizes of just 6 cm for females and 5.5 for males, with their eggs being the size of a coffee bean. The bee part of its name comes from its size, which is comparable to a bee. Also, like all other hummingbirds, it produces a humming sound by flapping its wings at high frequencies audible to the human ear. The bee hummingbird specifically beats its wings from 80 beats per second up to 200 beats during courtship flight.

The bird is native of Cuba, and can only be found on that island. If you are ever able to see it with a naked eye, you will probably find it hanging around vines and bromeliads, usually in the edge of a forest. Unfortunately, not much has been done to promote their conservation on the Cuban ecosystem. The species is labeled as Near Threatened on the Conservation Status scale [1].

This is where this project comes to offer some help. This urgent need for bee hummingbird protection and population control made this project idea very desirable. If there is a way to produce an automated algorithm to detect hummingbird activity from their distinct sounds, researchers and organizations can study the species closer and help it bloom in its ecosystem.

The inspiration for this project came from another research project I came across on the internet Mutsumi Saito and Fujitsu Technologies called “Protecting Endangered Owl Species with MATLAB Audio Processing Algorithms” [2]. The article briefly discusses the approach taken to perform Blakiston’s fish owl call recognition, which is made possible by taking the Fourier transform of data segments and analysing its frequency characteristics. The article is not very thorough on its algorithm implementation and results, but it made me realize how audio signal processing can provide a solution to the very important issue of species protection, and specifically bird species.

# APPROACH

## Database preparation

To acquire a dataset for clean and accurate hummingbird audio recordings, I used the eBird database from the Cornell Lab of Ornithology. The database contains relevant information for all bird species known, including images, audio recordings, and bird characteristics. The audio data also illustrates the spectrogram of the recording through time, which was very convenient in getting a quick initial idea of the characteristic sounds of a hummingbird. The audio data was not downloadable however, so I used the Sunflower extension which allowed me to record the audio data without the addition of any external noise from my environment. I stored each recording in an “aifc” file. When I had my data, recordings set up, a total of 10 recordings of variable length, I proceeded to the pre-processing phase of the algorithm.

## Pre-Processing

I loaded the audio data on Matlab, and resampled all signals to 44.1 KHz sampling rate. Then I performed Spectral Subtraction to remove the background environmental noise from the BH recordings, through a function that estimates the noise signal from the first 0.5 seconds of the sample. In order for the ss to work accurately, the signals where cropped to remove any initial complete silence recorded in the aifc file. From the eBird website spectrograms I had already discerned that the hummingbird distinct mating call is in the range of about 8.5 – 9.5 KHz. Thus, to further reduce low frequency noise, I used a high-pass filter with a cut-off frequency of 6 KHz.

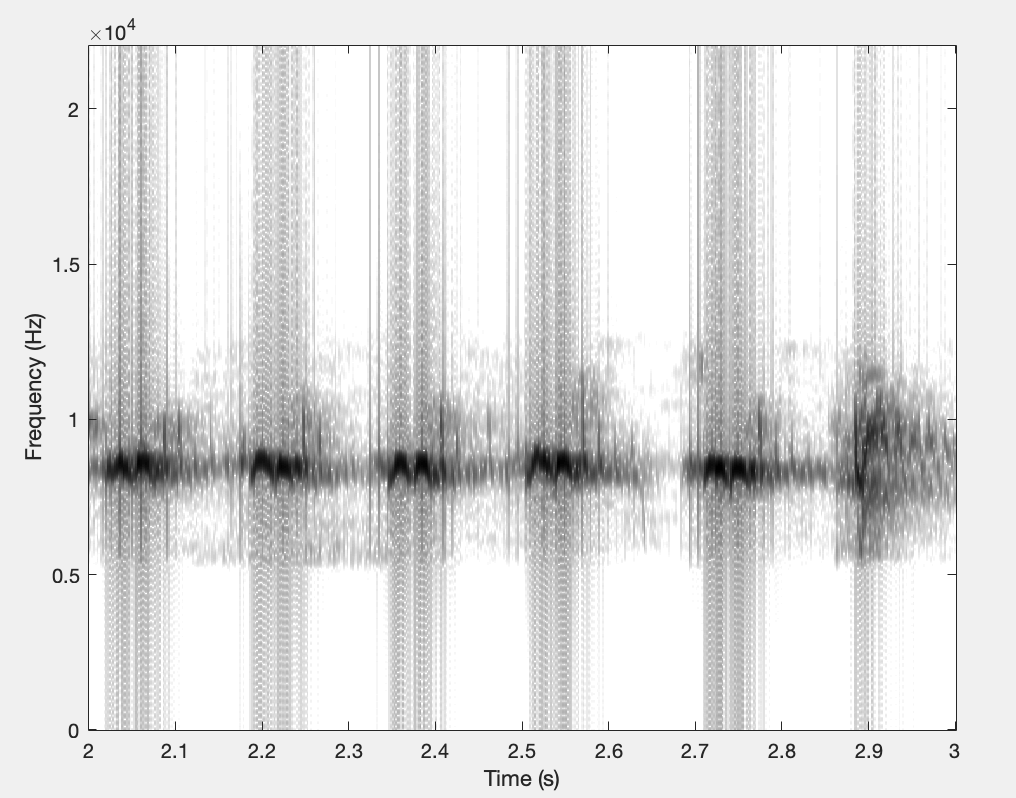


Figure 1: Distinct BH chirp sound Spectrogram (1-sec segment)

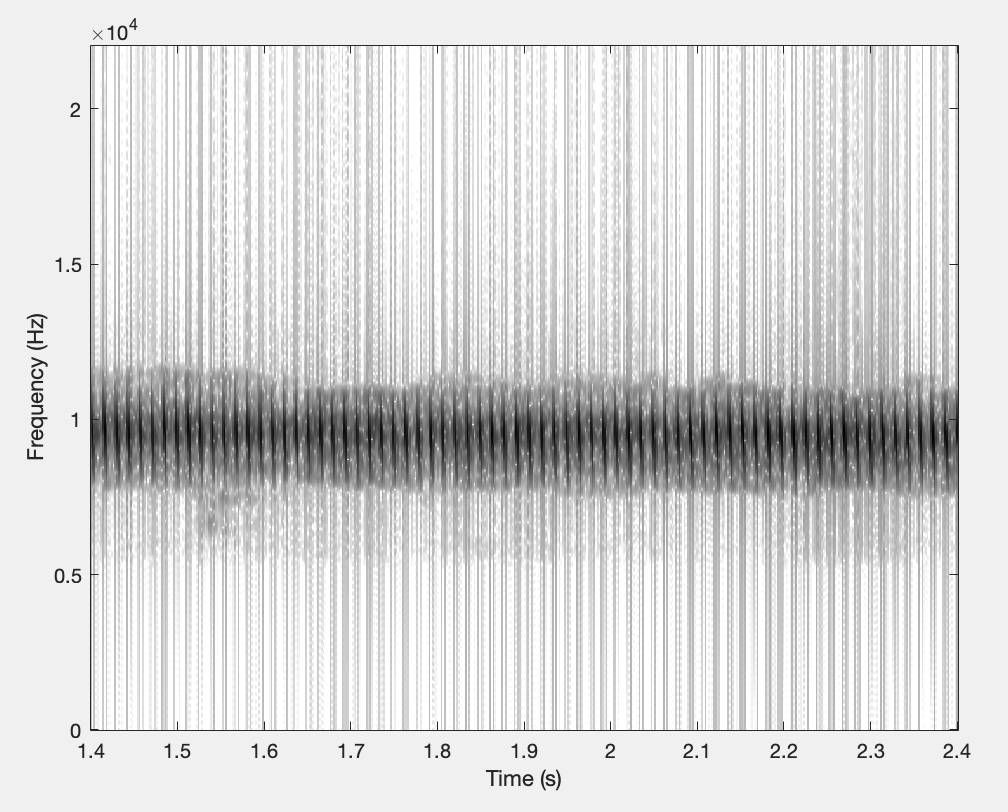


Figure 2: Distinct BH buzz sound Spectrogram (1-sec segment)

Through the pre-processing, I was able to store the processed signals and plot clean spectrograms of the recordings, and identify distinct frequency intensity patterns to perform a form of template matching. I was able to identify 2 distinct patterns, the ones illustrated on Figures 1 and 2. The duration of both of these patterns ranges from a bit more than a second to 2 seconds, with the second pattern, a type of buzz as I call it, lasting even up to 4 seconds. Looking through my spectrograms I found the cleanest instances of these patterns, and proceeded to creating a matching template for each sound pattern, that would be used to check and detect these sound patterns in test recordings.

Since both the patterns last more than a second but close to that, both of the templates are 1 second in duration so they can be able to detect these patterns in any duration they appear.

## Template Creation for Matching

The technique used to create the templates is called the Spectral Ensemble Average Voice Print (SEAV), an approach that was used in *H. Tyagi et al "Automatic identification of bird calls using Spectral Ensemble Average Voice Prints”.[3]* My algorithm uses the same technique with a few modifications and choice of some different parameters. The one second patterns are split in frames of 20msec and an overlap of 10msec, they are windowed and the N-point fast Fourier transform for each frame is computed. Since the templates are 1 second in duration, 20msec was determined to be an ideal frame size for segmenting the template and performing frame-wise examination of its Fourier characteristics. An overlap of 50% was the ideal choice for overlap-adding the windowed frames, and making sure all frequency characteristics of the sound pattern are taken into account. The value of N was determined to be 1024, the smallest power of 2 above the product of the sampling rate and frame size. Choosing a power of 2 as N speeds up and improves the quality of the FFT. The magnitude of the FFT is then taken.

If there are K frames, K vectors are formed, each one being the magnitude of the FFT of each frame. From these vectors, we compute an average ensemble vector computed like in the way presented in the following figure:

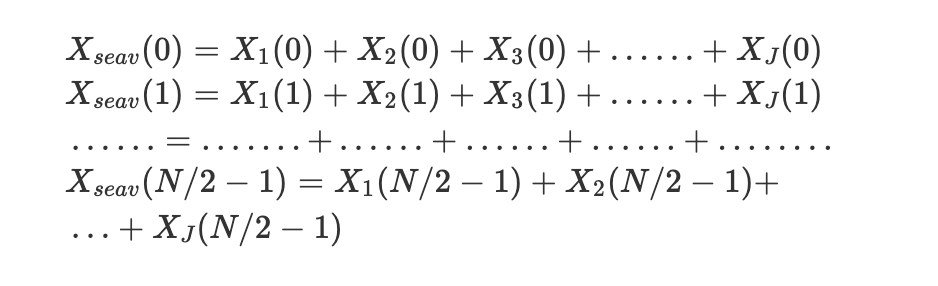


Figure 3: The computation of the SEAV vector related to the templates created.

As you can see, half of the samples are taken into account since the second half is just a mirror image, representing the negative frequency values.

The resulting vector is normalized and the result is used as a template. The two SEAV template vectors are plotted and depicted in figures 4 and 5. As you can see both templates are one narrow peak. That makes sense, since the bird sounds in question are of a specific range of frequency.

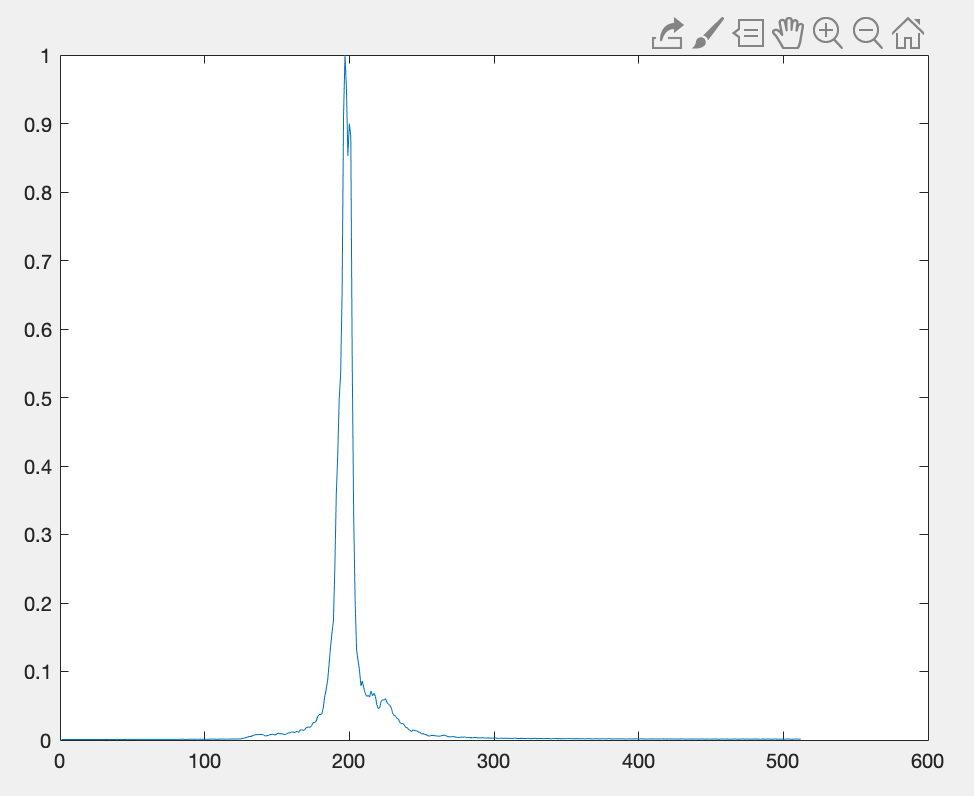


Figure 4: Template SEAV vector for chirp sound pattern

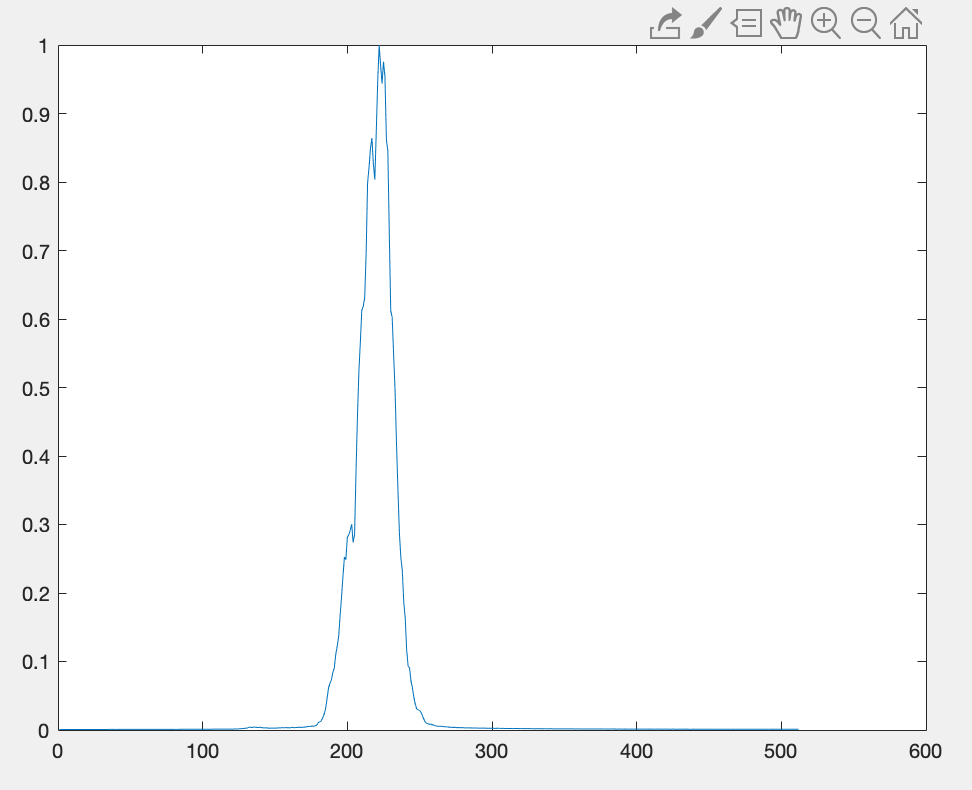


Figure 5: Template SEAV vector for buzz sound pattern

To compute the threshold of similarity to be used for hummingbird detection, each template is compared with a template of the same pattern sound taken from a different recording. For each of the two patterns, the Euclidean distance of the SEAV vectors of the similar templates is computed. The values for the distance are not used as a similarity threshold, this is adjusted through testing, but they give a general idea of where the threshold should be close to.

After doing comparisons for inaccurate and accurate templates, I managed to adjust my similarity threshold to ideal values. For the chirp sound, the threshold is placed at a value of 2.5, while for the buzz sound the threshold is at 1.5.

# Testing and Discussion

## Testing

For the final testing, I created 3 different datasets: one with Bee Hummingbird (BH) sounds, with 5 recordings ranging from a minute to 1.5 minutes, one with bird sounds from other native Cuban birds, such as the parakeet, the trogon, the flamingo, and the Cuban emerald (total of 6 recordings) again of similar durations, and a third one with 5 recordings of random sounds, such as music and human speech (taken from Project 1 of the course).

Testing for the Cuban Emerald recording is especially important, because this is the other Hummingbird species that can be found in Cuba. Since this bird is in the same species as the Bee Hummingbird, the might have similar calls. Thus, it is important to see whether the algorithm can distinguish between a BH sound, and sound from another Hummingbird species;

In running the testing script, the test recordings are segmented in 1-sec frames, with a 50% overlap between frames. Since the templates are of 1 second duration, it made sense to segment the recordings like that to compare each second taken from the test recordings with the matching templates created. The 50% overlap allows for a higher resolution in detecting a sound pattern, since a pattern can start and end in different 1-sec frames.

In the case of the BH sound samples, I manually created a vector pair for each test recording, containing the initial and final time in seconds, of time periods in the recording where a Bee Hummingbird can be spotted. I managed to do this by looking at the Spectrogram of each recording, and identifying where a BH sound pattern exists. I thus created a set of timestamps, each set related to one of the BH test recordings. The algorithm keeps track of whether a BH sound was detected in these time periods to determine the accuracy of the algorithm in detecting BH sounds. These are the results:

* In BH test sample 1, out of 8 time intervals, sound was detected in all 8
* In BH test sample 2, out of 8 time intervals, sound was detected in all 8
* In BH test sample 3, out of 9 time intervals, sound was detected in all 9
* In BH test sample 4, out of 7 time intervals, sound was detected in 6.
* In BH test sample 5, out of 5 time intervals, sound was detected in 4.

Note: Samples 4 and 5 were taken from the *xeno-canto* database, not from eBird, which provides recordings with worse quality and much more environmental noise. This could explain the algorithm’s failure to detect all sound instances.

**Overall the total accuracy out of all BH recordings is calculated to be ~95%.**

In the test dataset containing sound from other native Cuban birds, the algorithm detected a match in 3 time instances on the 4th test sample recording. This error can be serious. It is one thing if the algorithm fails to detect a real BH sound, and another, much worse, if it detects BH sounds where there aren’t any. I attempted to listen to the recording and check whether a BH sound is actually heard, but could not confirm it with certainty, even though multiple bird sounds take place in the recording, often simultaneously, so it could be easily the case. It is possible the de-noising and high-pass filtering applied to the recording isolates a ‘hidden’ BH sound, which is then detected.

On the bright side, in test recording number 6, which is the recording containing sounds from the Cuban Emerald (the other Hummingbird species in Cuba), the algorithm did not detect any BH sounds.

In test dataset containing random sounds, the algorithm successfully detects no BH sound.

## Discussion

The SEAV approach is only one and relatively simple approach to this issue. Through my research I came across many automated analysis techniques for bird identification and classification. All these implementations dealt with classifying a large number of species instead of focusing on one.

One of these approaches was introduced by Anderson et al and involves the integrating of DTW in template matching, to better fit songs that are of variable length and speed. Other methods include sequence modelling techniques which train the algorithm with certain data. One of the is the HMM, Trifa et al, which is widely used for human speech recognition too, and GMM, by Kwan et al [4].

What I learnt from my journey on this project is that you don’t always need to make your algorithm as complex as possible to maximize its performance and accuracy. In fact, it is desirable to minimize the computational load of your code without of course compromising the performance entirely. Another thing that I learned is that noise removal is critical in sound pattern recognition. A good noise removal algorithm to clean up the recordings analyzed can be a game changer in the accuracy of a sound identification algorithm.

##### References

1. G. Eason, B. Noble, and I.N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil.
2. “Protecting Endangered Owl Species with MATLAB Audio Processing Algorithms.” *MATLAB & Simulink*, www.mathworks.com/company/newsletters/articles/protecting- endangered-owl-species-with-matlab-audio-processing- algorithms.html.
3. H. Tyagi, R. M. Hegde, H. A. Murthy and A. Prabhakar*, "Automatic identification of bird calls using Spectral Ensemble Average Voice Prints,"*2006 14th European Signal Processing Conference, Florence, 2006, pp. 1-5.
4. Stastny, J., Munk, M. & Juranek, L. Automatic bird species recognition based on birds vocalization. *J AUDIO SPEECH MUSIC PROC.* **2018,**19 (2018). https://doi.org/10.1186/s13636-018-0143-
5. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (*references*)
6. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.I.S. Jacobs and C.P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.
7. K. Elissa, “Title of paper if known,” unpublished.
8. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
9. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
10. M. Young, The Tec