DETECTING BEE HIVE BEHAVIORAL CHANGES THROUGH FREQUENCY AND SIGNAL ANALYSIS OF AUDIO FILES

by

Preston Alexander Wilson

Honors Thesis

Appalachian State University

Submitted to the Department of Computer Science
and The Honors College
in partial fulfillment of the requirements for the degree of

Bachelor of Science

May 2019

Approved by:

_____

Rahman Tashakkori, Ph.D., Thesis Director

_____

Rick Klima, Ph.D., Second Reader

_____

Alice McRae, Ph.D., Departmental Honors Director

_____

Raghuveer Mohan, Ph.D., Departmental Honors Director

_____

Jefford Vahlbusch, Ph.D., Dean, The Honors College

ABSTRACT

Honey bees (*Apis mellifera*) are among the most important organisms in the world. The pollination they provide is crucial to the survival of many economically-significant plants, with more than 30 percent of the world's crops requiring cross-pollination in order to thrive [1]. Since the late 1990s, beekeepers around the world have reported significant declines in domestic honey bee populations. Increasingly, the majority of worker honey bees in a colony will simply leave their hives and never return, a phenomenon dubbed Colony Collapse Disorder. Though the causes still are not fully understood, Colony Collapse Disorder is thought to be a result of certain pesticides and mites.

In response, researchers have developed new methods of analysis to track the health of their hives, and one such method involves analyzing the sounds produced by a colony. Honey bees communicate through sound signals that occur within specific frequency ranges and signify different behaviors, and by determining the signals that are being produced, the behavior of a colony can be better understood [2]. This would allow for appropriate measures to be taken in the event of hive-adverse situations, such as predation or swarming.

As a part of this research, software was developed to plot the spectra of audio recordings from domestic hives. For each individual audio recording analyzed, the software computed the average magnitude throughout multiple frequency ranges along with the change in average magnitude between specific frequency ranges and exported this data to a spreadsheet for further analysis. These values were used individually and together to quickly identify audio recordings that stood out from a quantitative perspective, and these recordings were then individually analyzed in order to locate instances of strange honey bee behavior. While the software was able to quickly identify recordings with unusual-looking spectrograms, many of these were the result of external factors such as equipment movement, rather than high-frequency bee signals as desired.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude and appreciation to the following people, whose help and support allowed this project to come to fruition. My thesis mentor, Dr. Rahman Tashakkori, whose guidance and mentorship have helped me in ways that extend far beyond the scope of this work. My second reader, Dr. Klima, for his continuous help and constant availability that ensured a strong final product. My academic advisor, Dr. Dee Parks, for her contagious love of computer science as well as her counsel in shaping my career trajectory. The Appalachian State University Department of Computer Science professors as a whole, for the top-quality instruction I received during my undergraduate career. Carmen Wilson, my AP Statistics and Calculus teacher at Ashe County High School, for igniting within me a passion for mathematics and problem-solving that led me to this major. The McKinney Innovation and Joe Daniel Severt scholarships, for four years of support and funding. And most importantly, my parents, Michael and Ann Wilson, for their continual love and support in everything that I do.

CONTENTS

Chapter 1 – Introduction

The research conducted in this project made use of specialized software developed with the intent of finding specific honey bee audio signals within domestic hive audio recordings obtained at the entrance of the hives. To properly understand this form of honey bee communication, along with the process of events leading to recording retrieval and analysis, a firm understanding of basic acoustics terminology is required.

1.1 Abbreviations and Terminology

- **Audio Signal** – A representation of sound, using a series of binary numbers for digital signals.
- **DAS** – The DataAnalysis.py script, used to plot variables generated by the HBAA against each other and calculate correlation. In this research, a file's average magnitude was plotted against its transitory magnitude. Created as part of this project with the Python 2.7 programming language.
- **dBFS** – Decibels relative to full scale. Used as a unit of measurement for amplitude levels in a digital WAV file. A value of 0 dBFS is assigned to the maximum possible digital level [3].
- **Frequency** – Number of occurrences of a repeating event per unit of time, measured in hertz (Hz, equivalent to s^{-1}). Frequency is primarily what defines the pitch of a sound. Humans hear only frequencies produced between 20 and 20,000 Hz.
- **Fundamental Frequency** – The lowest frequency of a periodic waveform. When a signal is said to occur at a specific frequency, the fundamental frequency is what's being referred to.
- **Harmonics** – A positive integer multiple of the fundamental frequency.

— **HBAA** – The AudioAnalyzer.py script, also called the Honey Bee Audio Analyzer. This was the primary script used to analyze individual audio recordings, created as part of this project with the Python 2.7 programming language.

— **Magnitude** – Measure of how far a frequency's quantity differs from zero [4]. Used as a determinant of activity at the corresponding frequency.

— **Spectrogram** – A visual representation of the spectrum of frequencies of a signal as it changes with time.

— **Spectrum** – The different frequencies present in an audio file plotted against their power (measured in dBFS). A signal undergoes a mathematical algorithm known as a Fast Fourier Transform to generate a value for each band of frequencies that represents the quantity of those frequencies present. These values are interpolated to create the plot.

— **Transitory Magnitude** – Refers to the change in average magnitude between the 0 – 1000 Hz frequency range and the 1000 – 2000 Hz frequency range of an audio file. In other words, the average magnitude within the 0 – 1000 Hz frequency range of an audio file is calculated, and the average magnitude within the 1000 – 2000 Hz frequency range of an audio file is calculated. The difference between these two values is dubbed transitory magnitude.

1.2 Honey Bee Communication

Honey bees (*Apis mellifera)* have been found to produce a number of acoustic signals to communicate with each other under different behavioral contexts [5]. These specialized signals incorporate vibrations generated from the wing muscles and occur within distinct frequency ranges [6]. Most of these signals possess low fundamental frequencies between 200 to 600 Hz, though certain atypical signals may be produced at much larger frequencies. However, it is not only the frequencies produced that determine the meaning of a sound, but

also the acoustic structure of that sound, in terms of the signal pattern. Multiple studies have shown that the acoustic characteristics of a hive can be used to determine the health and activity levels of the corresponding honey bee colony [7].

1.3 The Piping Signal

Young queen bees may produce "piping", an acoustic signal that has different signatures depending on whether or not the queen is still confined. Piping emitted by emerged virgin queens is called "tooting", which possesses a fundamental frequency of 400 – 500 Hz. This signal's fundamental frequency increases with duration since emergence; i.e. if the virgin queen is freshly emerged, the fundamental frequency will be closer to 400 Hz, while if it's been several days since the queen emerged, the fundamental frequency will be closer to or even above 500 Hz [8]. The tooting signal starts with one or two pulses, each nearly a second in duration with a comparatively high amplitude and frequency, followed by several short pulses of about 0.25 second duration each. The number of short pulses is also related to the duration since the queen's emergence; from around 17 short pulses shortly after queen emergence to around 7 short pulses several days after queen emergence [8].

Piping produced by virgin queens still confined within their cells is known as "quacking." Quacking is emitted by each confined mature queen as a response to tooting, to indicate their presence and viability. The structure of quacking has been described as a series of very short pulses (each less than 0.2 seconds in duration) with a fundamental frequency of around 350 Hz, lower than that of tooting [8]. The exact purpose of quacking is unknown, though it's theorized that the signal may act as a call to worker bees to protect confined queen cells from a tooting queen, who will try to kill the occupants [5].

Worker bees also possess the ability to pipe. It was previously thought that worker piping was associated only with a lack of a queen or with hive disturbances, but the signal is increasingly being seen in undisturbed, queen-containing colonies [9]. The signal, when

produced by foragers in undisturbed conditions, is reportedly emitted once every few seconds, with a fundamental frequency of 330 – 430 Hz and little to no frequency modulation. Worker piping produced prior to and during swarms was demonstrated to possess much greater frequency variability, from 100 – 200 Hz to 500 Hz or greater, with each pipe consisting of a sound pulse lasting around one second in length [10].

1.4 Hissing

Worker piping may precede and occur during "hissing", an acoustic signal emitted by worker bees during swarms and times of distress [7]. Hissing possesses a very large fundamental frequency range of between 300 and 3600 Hz. The structure of the signal is broad brand and noisy, easily audible to the human ear and produced by very slight movements of the worker bees' wings, and typically accompanied by a rapid cessation of specific hive behaviors such as forager dancing and hive departures [11]. What's unique about hissing is that it can be indicative of very different colony events, depending on whether or not precursory piping is present. Past research suggests that a sequential and coordinated piping and hissing response is specific to the presence of potential predators close to the colony, and that the audio effect of hissing combined with the cessation of flight activity could decrease the risk of predation by birds and predatory insects [11]. Hissing without precursory piping is often present in the event of swarming and may be produced several days in advance from the actual departure [7]. Important honey bee signals are summarized in Table 1.

Table 1: Honey bee signals and what they signify [7]

Signal	**Frequency Range (Hz)**	**Signal Pattern**	**Sender**	**Significance**
Recruit	200 – 350	Pulse Sequence	Forager	Indicates existence of a quality food source
Tooting	300 – 500	Pulse Sequence	Queen	Subset of piping. Prevents hatching of further queens
Quacking	300 – 350	Pulse Sequence	Queen	Subset of piping. Indicates viability of confined, mature queen
Worker Piping	300 – 550+	Single Pulse	Scout	Triggers colony hissing to prepare to swarm
Hissing	300 – 3600	Single Pulse	Colony	General warning/defense signal. Occurs during swarming, hive attacks, and other adverse events.

Chapter 2 – Analysis of Audio Files

2.1 General Approach

A microphone connected to a Raspberry Pi computer was inserted into each domestic hive. These microphones recorded one minute of sound every four or five minutes, and these recordings were uploaded to the Appalachian State Department of Computer Science server. The audio recordings for the selected hive and time period were downloaded locally and analyzed by the HBAA and the DAS.

The HBAA created spectra of the audio files along with a spreadsheet of analyzed file data. This data was used to plot trends in hive behavior and to identify standout audio files, which were then individually looked at and listened to in the spectrogram plotting program iZotope RX 6 [12]. The DAS used the data from the HBAA spreadsheet to plot average magnitude against transitory magnitude in order to determine the relationship between the variables. Audio files that deviated from the expected relationship were also analyzed in iZotope [12]. Figure 1 summarizes the process of events for analyzing audio files within this research, and each step is described in greater detail in the following subsections.

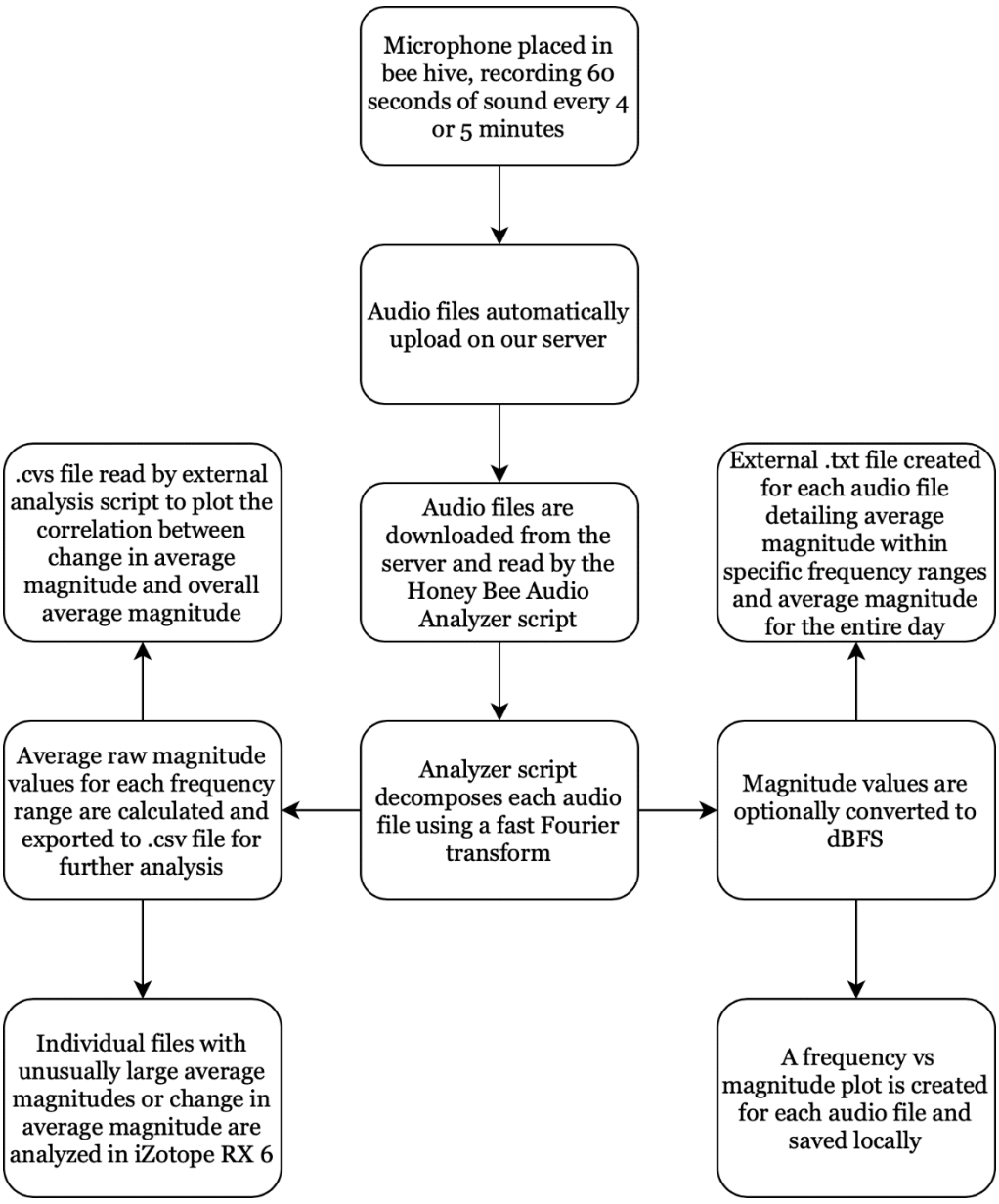

Figure 1: Flowchart detailing the process of events for analyzing honey bee audio files

2.2 Recording Colony Behavior

For the purposes of this research, several domestic western honey bee hives each had a microphone inserted inside of them and a camera placed above them, with these devices connected to Raspberry Pi computers. The cameras were strategically placed to record activity at the entrance of each hive, while the microphones recorded sounds produced inside each hive [13]. Both the camera and microphone recorded sixty seconds of information at a time, between 7:00 AM and 8:00 PM, with three to four minutes between the end of one recording and the beginning of another. These recordings were then uploaded to the Appalachian State University Department of Computer Science server, where they were stored for easy retrieval and analysis. The hives used for this project were RPI11b and RPI24 (named after the Raspberry Pi computer connected to each hive).

Figure 2: Microphone placed in hive

2.3 Audio Analyzer

For analysis of the audio recordings a Python script called HBAA was developed as part of this research and used. The purpose of the HBAA was to grab and evaluate specific audio recordings based on parameters such as hive, start date, end date, and hour(s) within the day. Initially each chosen audio recording was read and flattened, with the audio signal and recording rate passed to a plotting function. The plotting function then created a spectral plot from the audio signal and exported specific data values of interest to a spreadsheet for further analysis. The flattened audio signal was used to create a Hann window, which underwent a Fast Fourier Transform to create a spectrum representation of the original audio signal. A default sampling rate and default sampling size of 44100 Hz and 32768 respectively were used. Average magnitudes for all frequencies within the signal were then returned, and average magnitudes within specific frequency ranges were calculated for data export.

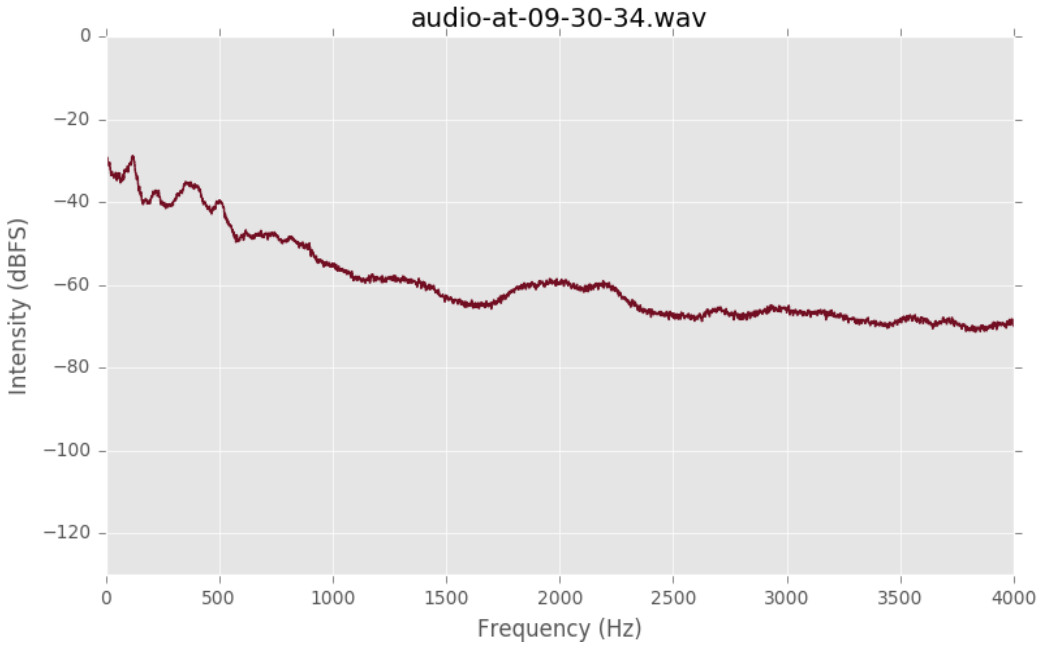

Figure 3: Spectrum of audio file for RPI11b May 1st, 2018 at 9:30 am

Within the plotting function was a parameter to optionally convert the magnitudes of the signal to dBFS which was used in the creation of the spectral plots. This conversion was not used when comparing the average magnitudes within specific frequency ranges, due to dBFS being a logarithmic measurement rather than a linear one. Figure 3 is an example of a spectral plot created from an audio file with the dBFS conversion parameter active. On the x-axis is frequency and on the y-axis is intensity, measured in dBFS. For the file above, it's clear that there was more intensity (and thus a greater average magnitude) at and below 1000 Hz than at other frequencies, suggesting that the honey bees were primarily producing signals below 1000 Hz, and especially below ~400 Hz.

Analysis of file spectra was useful for determining behavior within individual audio files but was insufficient for determining trends in activity. For this purpose, the HBAA calculated the average magnitude under 4000 Hz for a day's worth of audio files, so that trends in colony activity levels over time could be plotted. An example is shown in Figure 4.

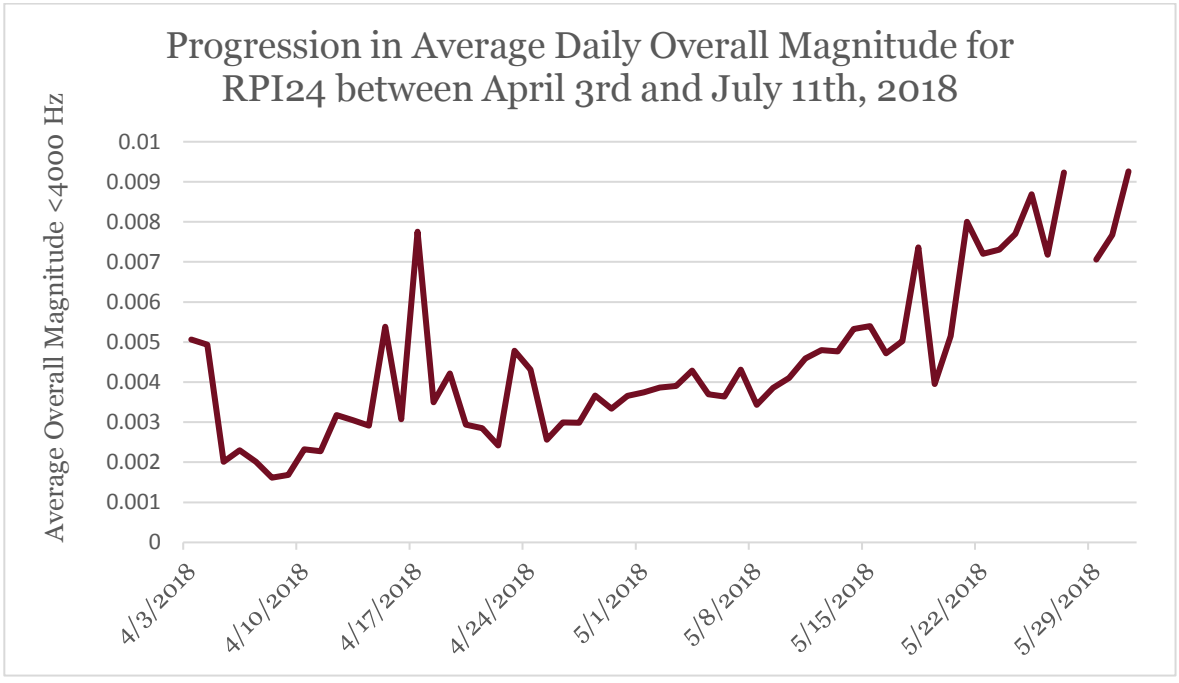

Figure 4: Example of using the overall average magnitude to see trends in hive behavior, in this case for hive RPI24 for April and May 2018

2.4 Spectrogram Plotting

For more in-depth analysis of specific standout audio files, iZotope RX 6 was used [12]. The program was used to plot the spectrogram of an individual audio file, so that changes in frequency and magnitude *with respect to time* could be seen. The same audio file from Figure 3 plotted in iZotope is shown in Figure 5.

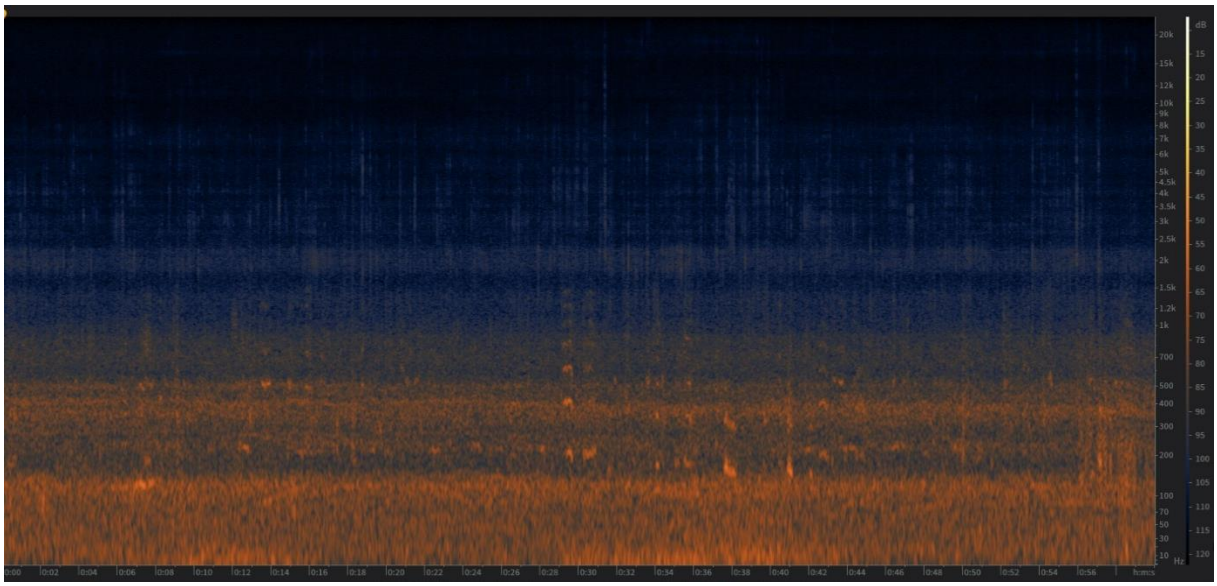

Figure 5: Spectrogram of an RPI11b audio file recorded on May 1st, 2018 at 9:30 am

On the x-axis is time, and on the y-axis is frequency. Color represents magnitude, with warmer colors indicating a larger magnitude at the corresponding frequency and time. High-frequency signals may only be heard for a very short time period, and thus may be difficult to detect within a spectrogram. A spectrogram of an audio file suspected to contain piping is shown in Figure 6, with the piping signal circled in red. The signal appears 38 seconds into the recording, with a fundamental frequency of 750 Hz and harmonics at approximately 1500 and 2250 Hz.

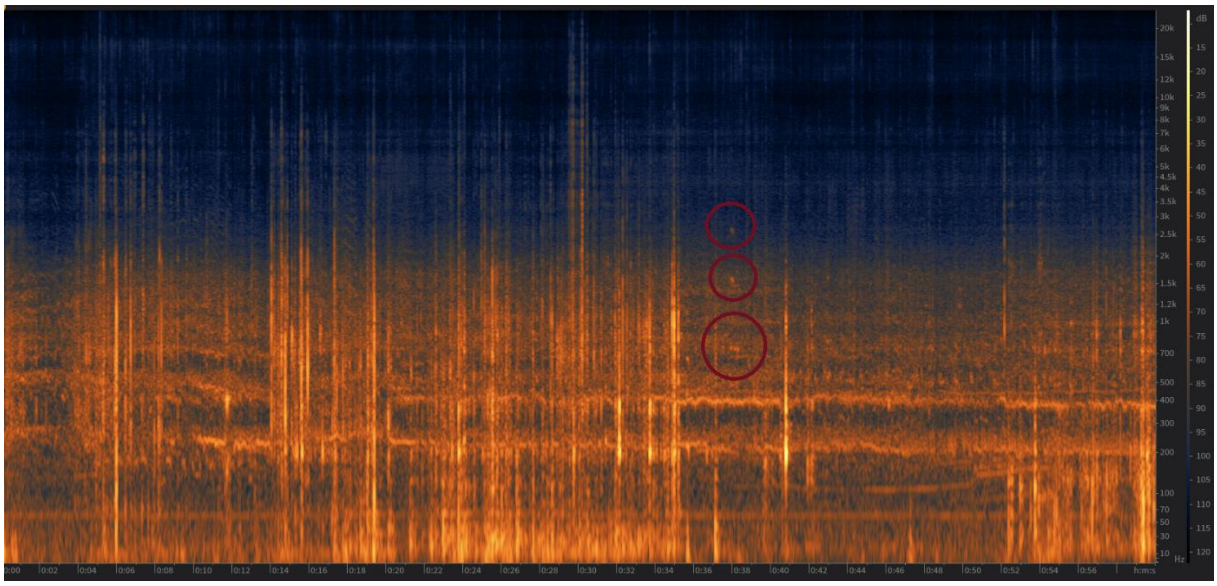

Figure 6: Suspected piping in RPI11b on April 10th, 2018 at 6:36 pm

For this reason, simply looking at the overall average magnitude within a specific frequency range was insufficient for detecting the presence of short-pattern signals such as piping and was instead better suited for detecting long-lasting signals emitted from the colony, such as hissing. If piping was suspected, either due to subsequent swarming or the gathering of the colony at the entrance, individual spectrograms of the audio files were viewed and analyzed.

The audio files used for this research underwent a de-humming algorithm to remove traces of acoustic noise emitted from the microphone. An example of how this algorithm affects the spectrogram of an audio file is shown in Figures 7 and 8.

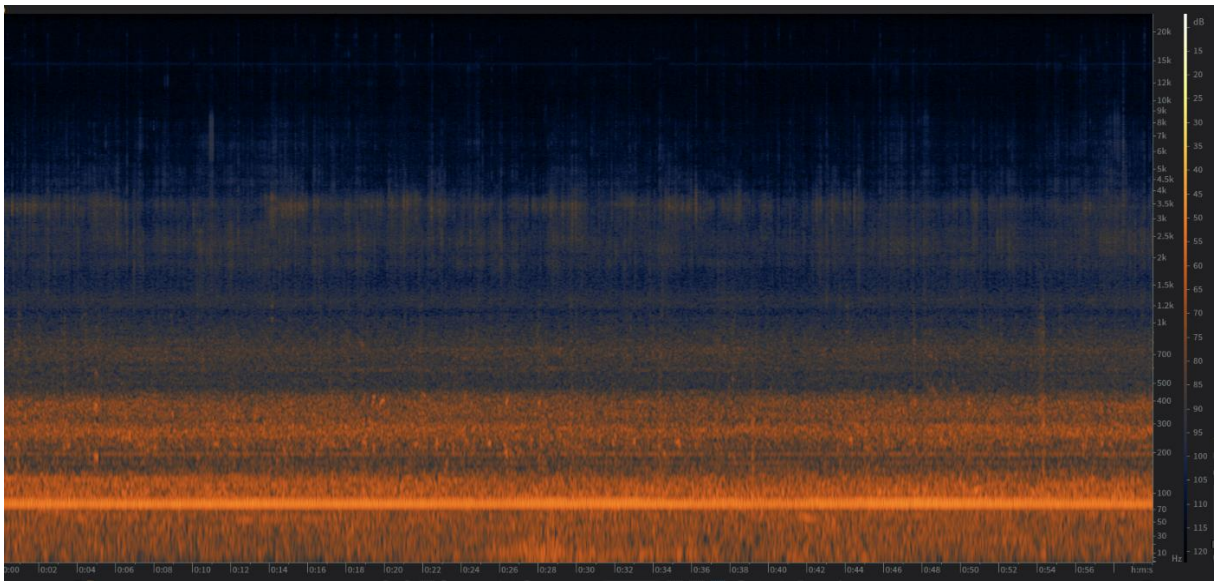

Figure 7: Spectrogram of RPI24 5/04/2018 at 2:52 pm prior to de-humming

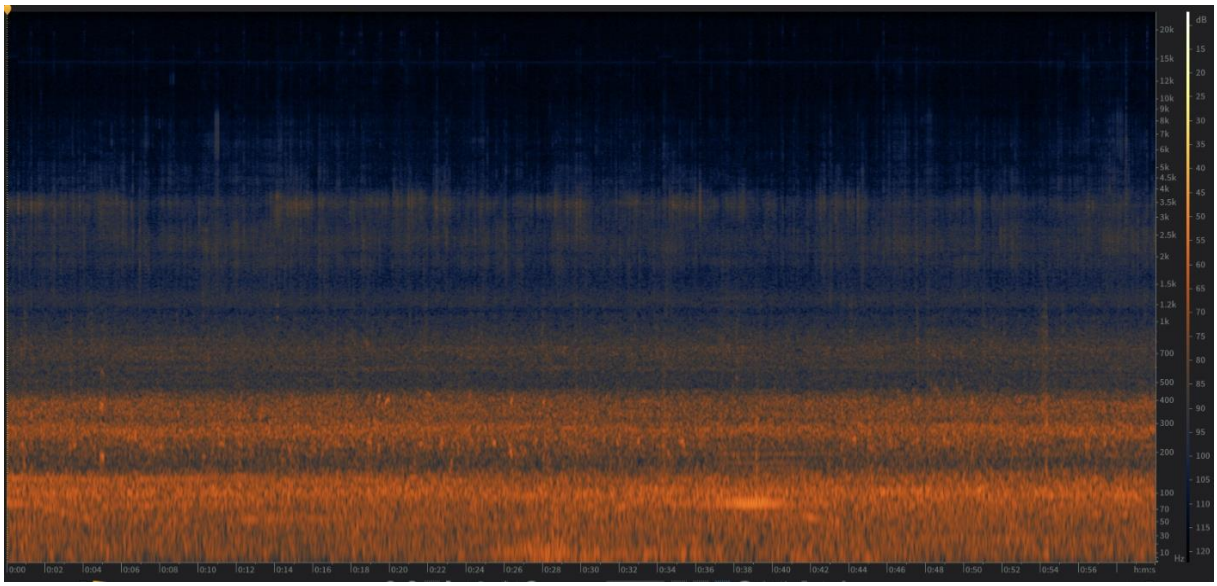

Figure 8: Spectrogram of RPI24 5/04/2018 at 2:52 pm post de-humming

From Figure 7 to Figure 8, the de-humming algorithm removed a large band of noise between 70 – 100 Hz, leading to a spectrogram more reflective of actual hive behavior for the audio file.

2.5 Quantitative Frequency Range Analysis

Along with the creation of spectral plots, the HBAA created and exported spreadsheets containing quantitative information of each audio file analyzed, including average magnitude within specific frequency ranges and the change in average magnitude between frequency ranges. This information was heavily used in the analysis of the audio files to find unusual signals and events over extended periods of time. For each day analyzed, an additional text file was created containing the previously-mentioned data in a more concise format along with the overall average magnitude for an entire day's worth of audio files. All of this information was also printed to the console for each audio file analyzed. An example of the information contained within each spreadsheet is shown in Table 2.

Table 2: Information exported to spreadsheet by the HBAA program for each audio file used in analysis

File	Average magnitude < 1000 Hz	Average magnitude 1000 – 2000 Hz	Average magnitude 0 – 4000 Hz overall	Transitory Magnitude
rpi24\2018-05-01\audio\09-00-23.wav	0.00979704	0.00062821	0.002911244	-0.00916883
rpi24\2018-05-01\audio\09-05-25.wav	0.00969534	0.000727181	0.00289430	-0.00896816
rpi24\2018-05-01\audio\09-10-27.wav	0.00986313	0.000677217	0.00298044	-0.00918591

One of the problems that arose when comparing average magnitudes of audio files from different hives was that these values were susceptible to certain confounding variables, such as microphone placement in relation to the colony. For instance, a microphone placed close to the center of the hive will likely record greater magnitudes across all frequencies compared to a microphone placed at the periphery, and for this reason transitory magnitude was introduced.

Transitory magnitude is a shorthand term for looking at the *change in average magnitude* between the first two successive 1000 Hz frequency ranges (the frequency ranges where the vast majority of honey bee activity takes place) as a metric to gauge levels of colony behavior. This is ideally a better metric for comparing activity levels *between hives*, as all frequency ranges are affected by microphone placement equally. In other words, when the difference in magnitude between frequency ranges is calculated, the resulting value is based solely on the proportion of activity within the first two frequency ranges. Unaffected by microphone placement, this makes the value better suited than average magnitude for comparison between hives. A file with a large transitory magnitude is one where the average magnitude in the 1000 – 2000 Hz frequency range is significantly greater or smaller than the average magnitude in the 0 – 1000 Hz frequency range. A file with a small transitory magnitude has similar average magnitudes in both the 0 – 1000 Hz and 1000 – 2000 Hz frequency ranges.

Correlation between average magnitude and transitory magnitude was investigated in this research, using the DAS to plot the variables against each other. This script also determined which audio files followed the expected trend between these variables the least, and these audio files were individually analyzed. An example of DAS output is shown in Table 3.

Table 3: Example of DAS output, in descending order based on difference between expected and actual value in the average vs transitory magnitude plot

Difference between Expected and Actual Position in Plot	**Recording Number**	**Recording Path**
0.001416737403198	1155	'E:\\Dehummed_Thesis\\rpi24\\2018-05-25\\audio\\11-11-14.wav'
0.001410993410107	1441	'E:\\Dehummed_Thesis\\rpi24\\2018-05-31\\audio\\09-20-29.wav'
0.001410941029959	1639	'E:\\Dehummed_Thesis\\rpi24\\2018-06-05\\audio\\17-28-56.wav'

While this analysis was primarily audio-based, video files were used to view behavior at the entrance of each hive, to either substantiate or disprove suspicions that arose from the audio files [14]. Domestic hives RPI11b and RPI24 were analyzed from the beginning of May 2018 through the end of January 2019. Occasionally specific files outside of this date range were analyzed. New microphones were inserted into these hives near the end of September 2018 and were dubbed RPI11b-a and RPI24-a for their respective hives. Audio recordings of the hives (when available) recorded between 8:00 – 9:00 AM, 11:00 AM – 12:00 PM, 2:00 – 3:00 PM, and 5:00 – 6:00 PM were used for analysis, to give an accurate overview of behavior across the day for each hive.

Chapter 3 – Results

3.1 Average and Transitory Magnitude Relationship

Transitory magnitude, introduced as a method of analyzing hive activity levels *between hives*, was found to have a strong correlation with overall magnitude levels. Figure 9 shows the relationship between average overall magnitude (for all frequencies <4000 Hz) and the transitory magnitude for all audio files analyzed in RPI11b, plotted by the DAS. The R-squared value was determined to be 0.9891 for RPI11b, indicating a direct relationship between the variables. Figures 10 and 11 show this same relationship for RPI24, which had an R-squared value of 0.9847, again suggesting a direct relationship between average overall magnitude and transitory magnitude. Audio files that didn't follow this linear relationship were considered standout files and analyzed independently.

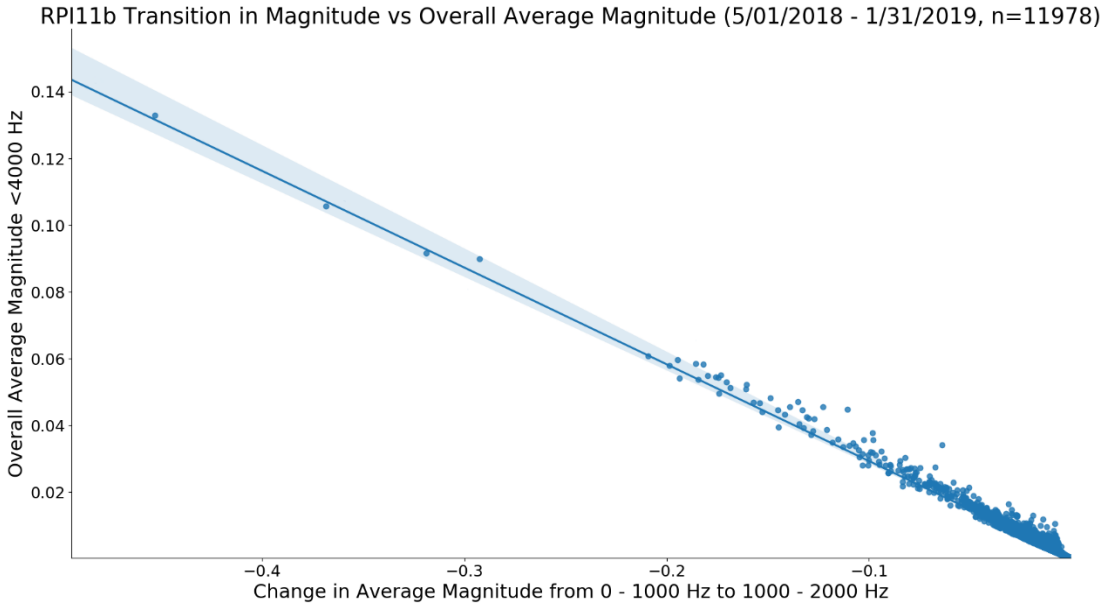

Figure 9: Plot of RPI11b average magnitude vs transitory magnitude

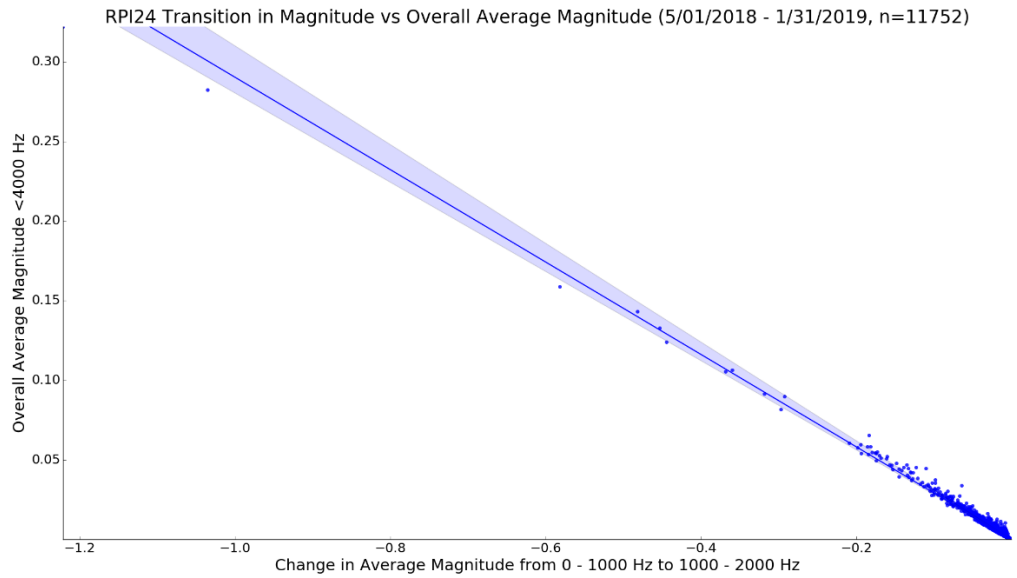

Figure 10: Plot of RPI24 average magnitude vs transitory magnitude

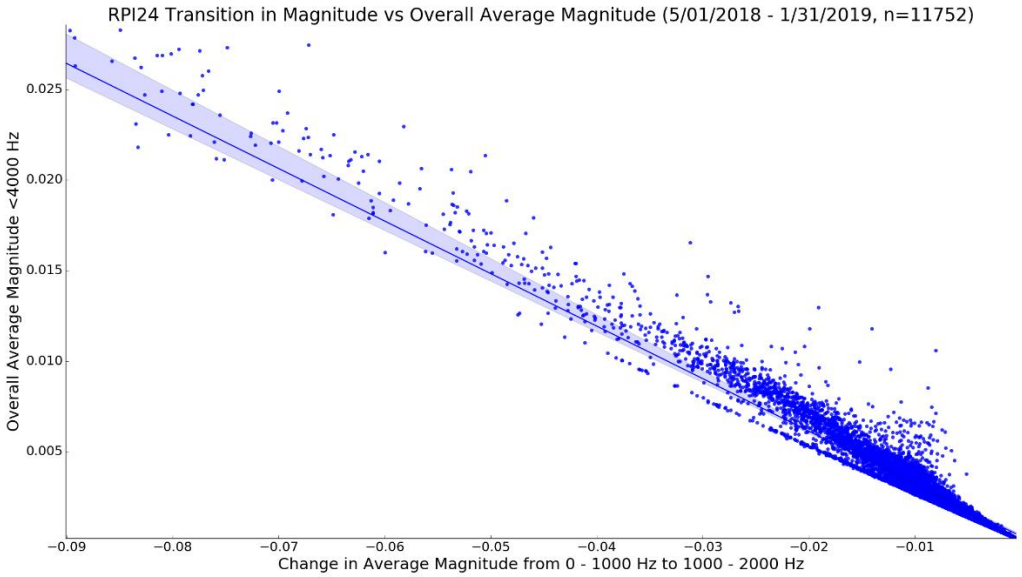

Figure 11: Plot of RPI24 average magnitude vs transitory magnitude (zoomed)

3.2 RPI11B

3.2.1 Trends in Activity

A scatterplot detailing the change in average daily magnitude over time for RPI11b/RPI11b-a is shown in Figure 12. A total of 11978 audio files were analyzed by the HBAA for this hive. An average of 43 recordings were used in the calculation of each average daily magnitude value, depending on availability.

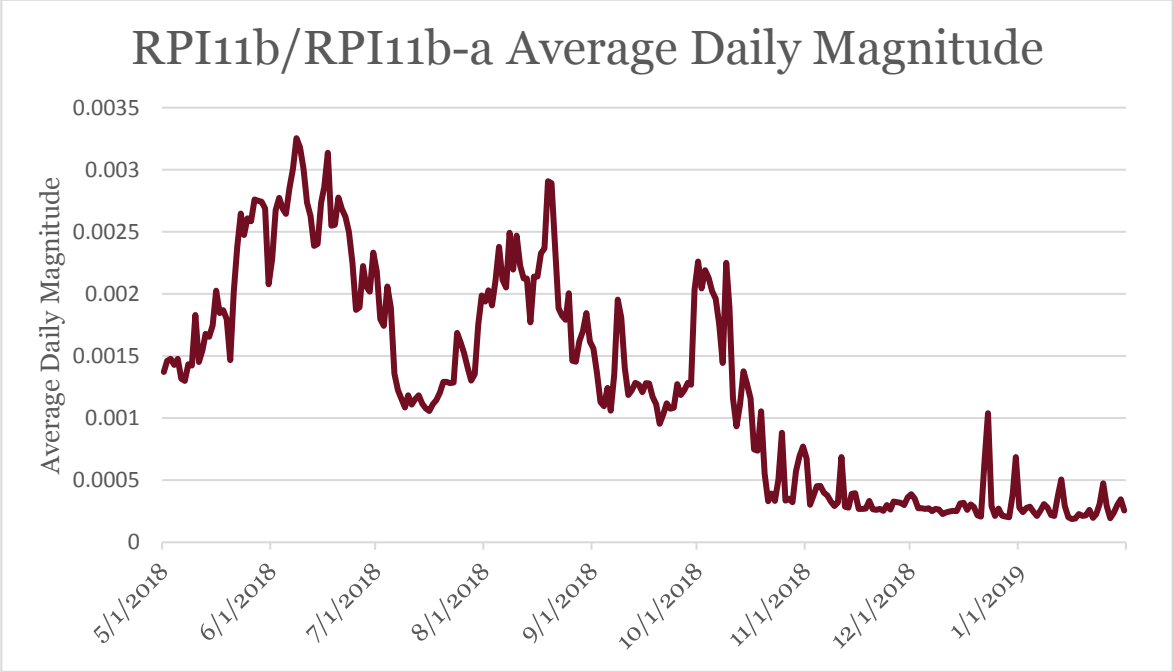

Figure 12: Plot of RPI11b change in average daily magnitude from May 2018 through January 2019

A new microphone was introduced to the hive from September 30th onwards, dubbed RPI11b-a. The introduction of RPI11b-a coincided with a sharp increase in average daily magnitude, due to the fact that RPI11b's microphone had become partially covered with propolis, a sticky combination of sap and beeswax produced by the bees to coat their hives. Activity sharply declined starting in the middle of October and did not recover for the evaluated time period, which was expected, as bees move less in the winter months [15].

Daily activity consistently rose from the beginning of May until peaking in June, before dipping and recovering in July.

3.2.2 Standout Files

The following audio files stood out from a quantitative perspective, either due to an abnormally high average magnitude (overall or within a specific frequency range), or due to some abnormality detected through transition analysis (e.g. didn't follow the expected relationship between overall average magnitude and transitory magnitude).

Sorting all RPI11b/RPI11b-a files based on largest overall magnitude showed that most were recorded in June. The spectrogram of the recording with the second largest overall magnitude, recorded on June 23rd, is shown in Figure 13, along with a screenshot from the analogous video recording in Figure 14.

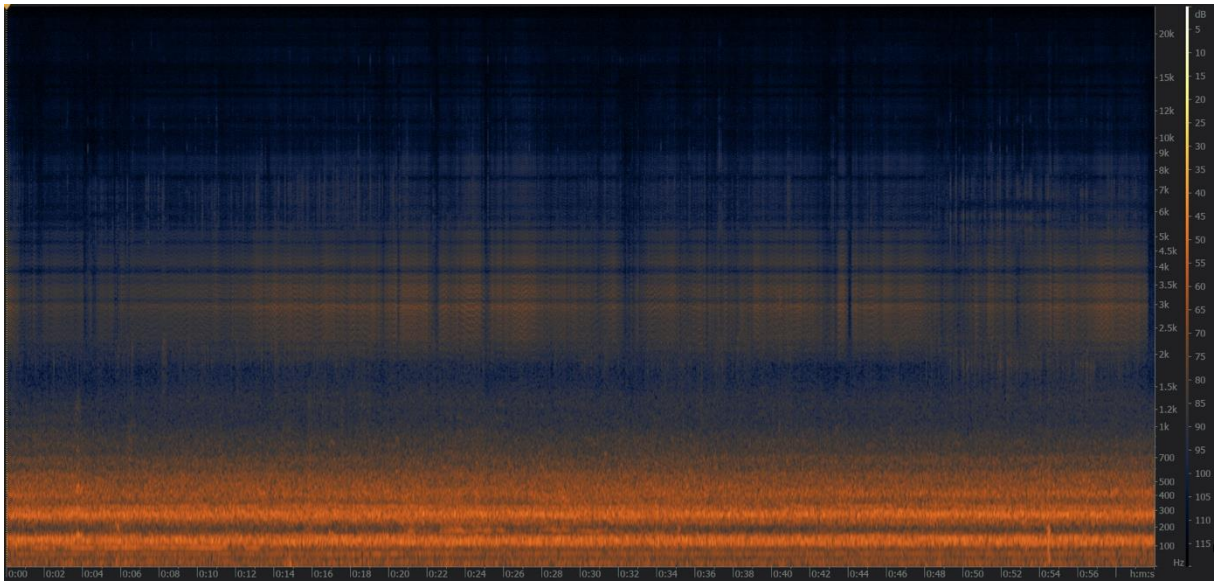

Figure 13: Spectrogram for RPI11b audio file recorded on 6/23/2018 at 5:50 pm

Figure 14: Screenshot of video recording for RPI11b on 6/23/2018 at 5:51 pm

A large amount of buzzing was present throughout the audio recording, with the majority occurring between 100 and 500 Hz. Many bees were surrounding the entrance and flying in and out of the hive in the corresponding video recording. Several drones were present throughout the video recording. No unusual activity was observed, nor were any strange signals identified within the spectrogram.

The file with the largest overall average magnitude was recorded on October 10th at 9:05 AM. The spectrogram of that recording is shown in Figure 15.

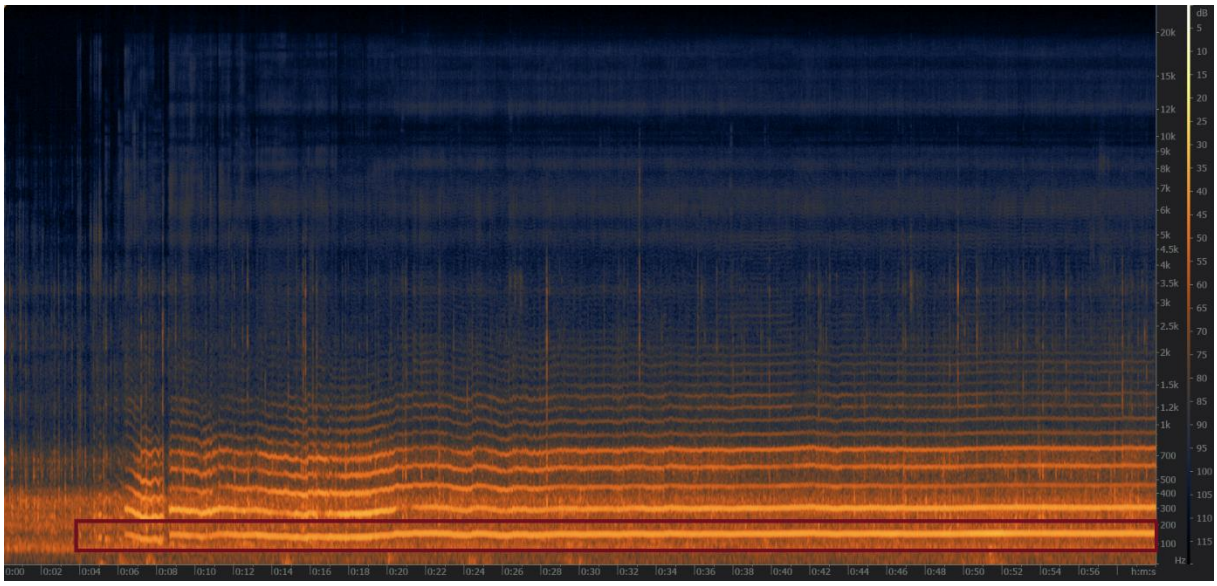

Figure 15: Spectrogram for RPI11b-a audio file recorded on 10/10/2018 at 9:05 am

The waves beginning at approximately 160 Hz, with the fundamental frequency boxed in red and harmonics appearing at positive integer multiples of this frequency, were responsible for the very large average magnitude of the recording. They were the result of a single bee moving across the microphone for the majority of the recording. No unusual behavior was seen in the corresponding video recording.

Sorting the recordings based on transitory magnitude had similar results. Most of the recordings with large transitory magnitudes were recorded in June, just as with the overall magnitude, though many were recorded in July as well. The recording with the largest overall transitory magnitude was recorded on July 15th at 2:13 PM, and the spectrogram of that recording is shown in Figure 16.

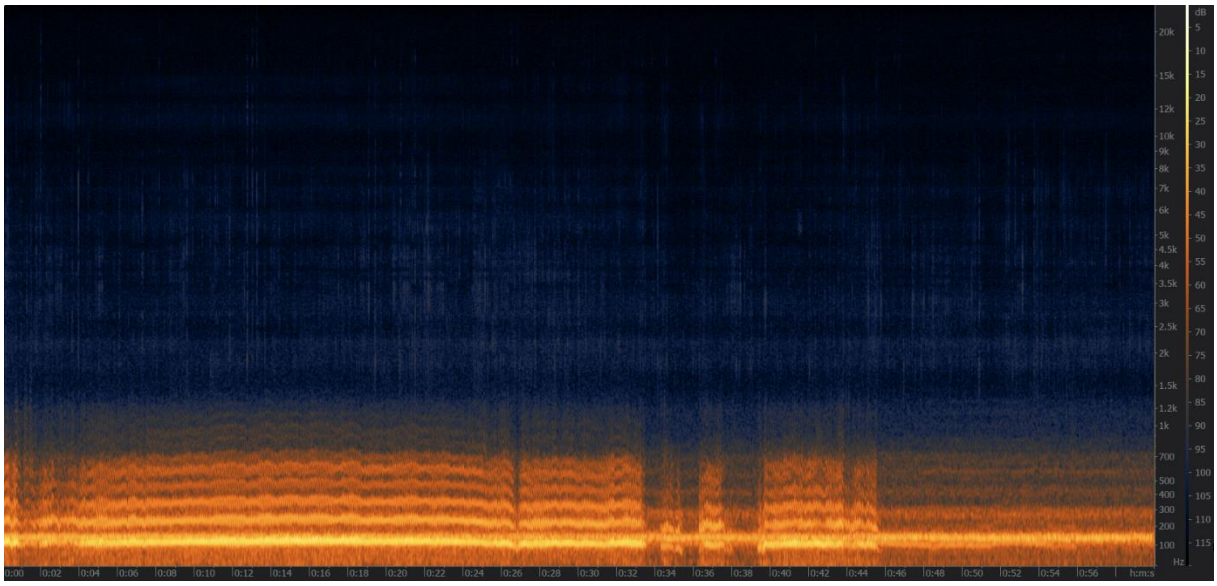

Figure 16: Spectrogram for RPI11b audio file recorded on 7/15/2018 at 2:13 pm

Just as with the recording for Figure 15, this spectrogram's odd shape was caused by a single bee close to the microphone. Here however, the fundamental frequency produced by the bee was a bit lower, between 110 and 150 Hz. For this reason, more harmonics were present, with less space between them compared to Figure 15. No odd honey bee behavior was observed in the corresponding video recording.

Another recording with a very large transitory magnitude was recorded in January, and the spectrogram of that file is shown in Figure 17.

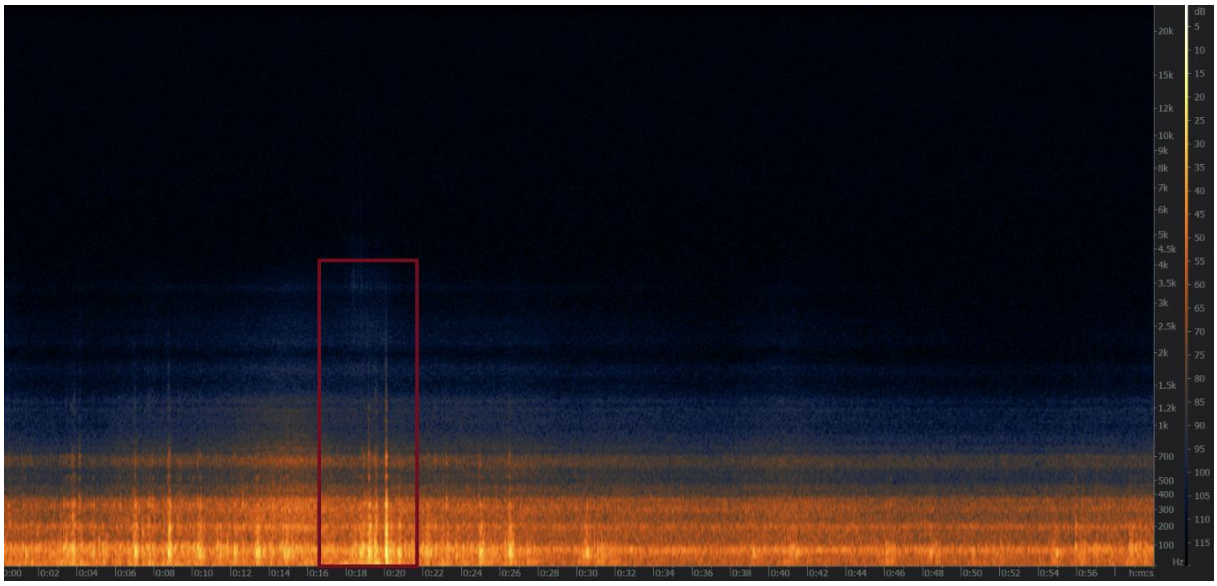

Figure 17: Spectrogram for RPI11b-a audio file recorded on 1/09/2019 at 9:31 am

The microphone or another piece of equipment was moved at the 20 second mark, boxed in red above, and was responsible for the corresponding large peak. Minor movements of the same equipment were present throughout the recording, and responsible for the many small peaks seen in the spectrogram. Light buzzing was heard throughout the recording, and no unusual honey bee behavior was seen in the corresponding video file.

The DAS was used to find the files that most greatly differed from the expected average magnitude and transitory magnitude relationship. The script found that most of these files were recorded in December and January. The file that most differed was recorded on December 10th at 9:05 AM. That recording's spectrogram is shown in Figure 18.

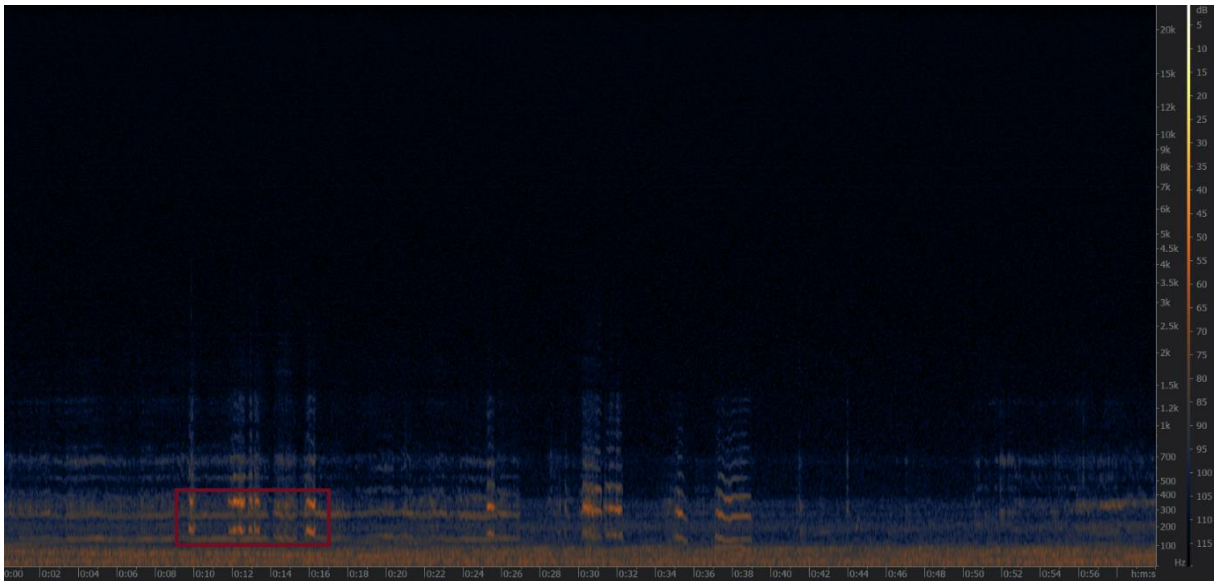

Figure 18: Spectrogram for RPI11b-a audio file recorded on 12/10/2018 at 9:05 am

This file possessed an extremely small overall average magnitude with a larger than expected transitory magnitude, caused by the presence of a single bee that moved towards the microphone intermittently throughout the recording. Some of the sounds produced by this bee are boxed in the spectrogram. No adverse behavior was seen in the corresponding video file.

3.2.3 RPI11b Hive Conclusions

All other viewed spectrograms from RPI11b that stood out quantitatively were similar to those described above. The hive overall seemed to have relatively stable activity levels throughout the analyzed time period, with a bit of a dip in the winter months as expected [15]. The majority of audio recordings with unusual looking spectrograms were the result of equipment movement or bees moving near the microphone. It's possible that piping or hissing were present in some of the analyzed audio recordings; if so however, the recordings did not stand out from a quantitative perspective.

3.3 RPI24

3.3.1 Trends in Activity

A scatterplot detailing the change in average daily magnitude over time for RPI24 is shown in Figure 19. A total of 11752 audio files were analyzed by the HBAA for this hive. An average of 43 recordings were used in the calculation of each average daily magnitude value, again depending on availability. A new microphone, RPI24-a, was introduced on October 1st.

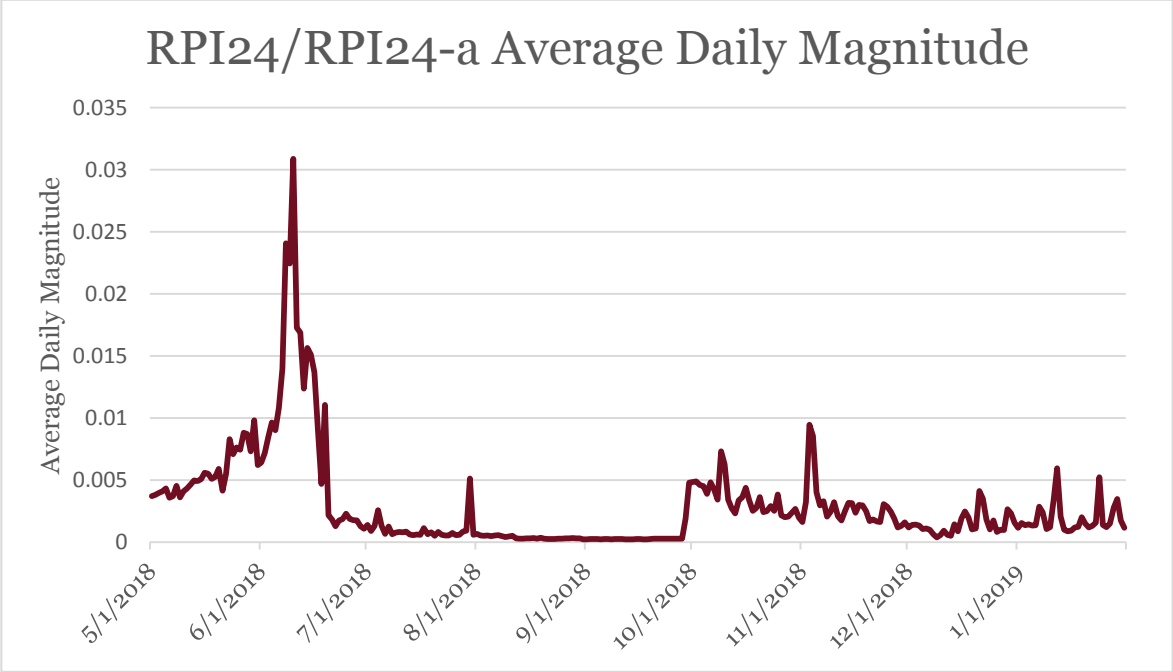

Figure 19: Plot of RPI24 change in average daily magnitude from May 2018 through January 2019

Average daily magnitude was nearly zero from the beginning of July until the introduction of RPI24-a, due to significant coverage of the microphone by the bees with propolis. For this reason, the daily average magnitude values from July until the beginning of October are inaccurate, and not reflective of the true hive audio activity levels for the time period. That being said, it's evident from the plot that activity sharply peaked in early June, and never recovered even after the new microphone was inserted. Activity was slightly greater in October and November compared to December and January, again as expected [15].

3.3.2 Standout Files

Sorting the over 11,000 RPI24/RPI24-a audio files analyzed by the HBAA based on smallest transitory magnitude, i.e. the change in average magnitude between 0 – 1000 Hz and 1000 – 2000 Hz, resulted in multiple audio files recorded on September 24th, 2018 between 2 – 3 PM rising to the top. When viewed individually in iZotope, these audio files displayed very small amounts of overall activity.

Figure 20 shows the spectrogram for the audio file with the smallest transitory magnitude, while Figure 21 shows a screenshot from the analogous video file.

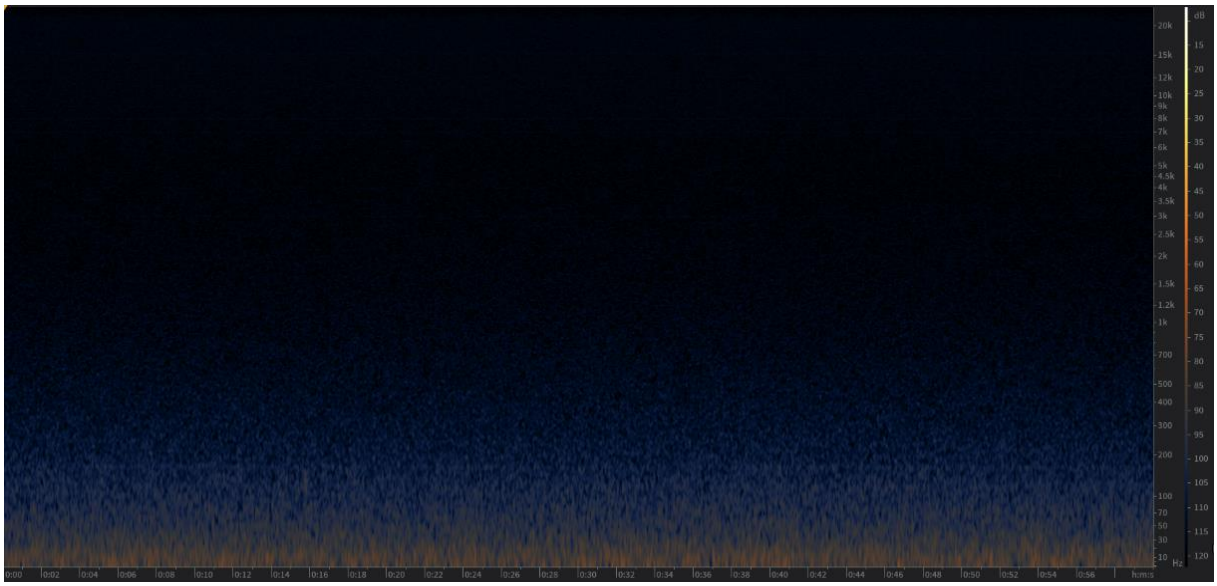

Figure 20: Spectrogram for audio file from RPI24 recorded on 9/24/2018 at 2:57 pm

Figure 21: Screenshot from RPI24 video file recorded on 9/24/2018 at 2:57 pm

Very small amounts of buzzing were present from listening to the audio file, while the analogous video file showed a mostly inactive hive with several dead bees at the entrance surrounded by a few slow-moving workers. Very few audio signals could be detected within the spectrogram, especially with such small magnitudes. All of the 100 audio files with the smallest amounts of transitory magnitude took place within a week of September 24th, 2018. The large accumulation of propolis over the microphone is likely responsible for the low transitory magnitudes around this time period, as the hive had a new uncovered microphone placed within it only a week later, but video files for the time period also indicated low levels of colony activity.

Looking at the audio files with the largest transitory magnitude produced considerably different spectrograms. These audio files were more scattered in date, though the majority were recorded within the first two weeks of June 2018. This is very consistent with the daily average magnitude scatterplots. The spectrogram for the audio file with the largest overall transitory magnitude is shown in Figure 22.

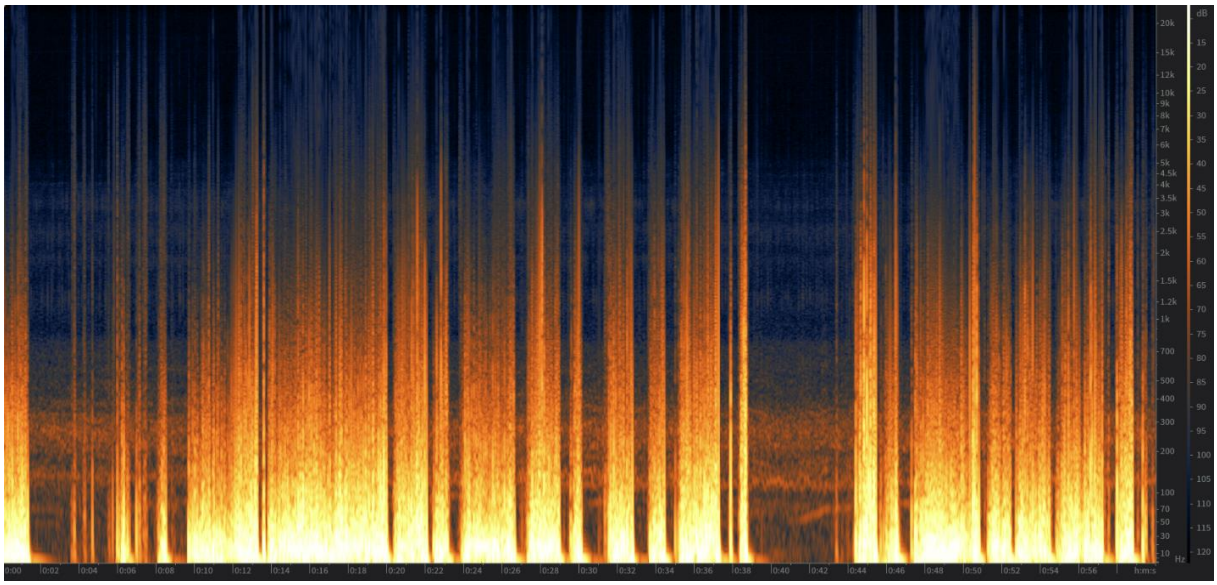

Figure 22: Spectrogram for audio file from RPI24 recorded on 6/22/2018 at 5:19 pm

Figure 23: Screenshot from RPI24 video file recorded on 6/22/2018 at 5:19 pm

The video file shows relatively large amounts of activity immediately outside the hive. Many bees are congregated at the entrance, and quite a few drones are present. However, listening to the audio file suggests that the unusual spectrogram is the result of the microphone being moved, and that the large peaks and frequencies did not originate from the honey bee communications.

The DAS was then used to find audio files within the first two weeks of June 2018 (when activity levels were highest) that *least* followed the expected relationship between overall average magnitude and transitory magnitude. An audio file recorded on June 5th was present, with the third greatest deviation from the linear regression line out of all files analyzed for the hive. The spectrogram of that file is shown in Figure 24.

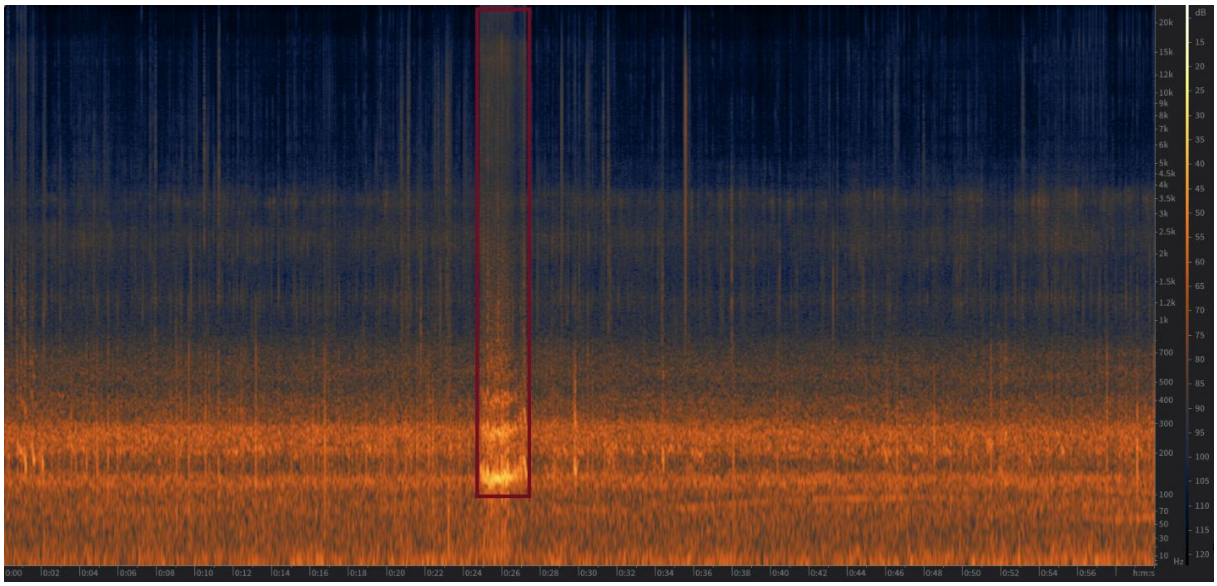

Figure 24: Spectrogram for audio file from RPI24 recorded on 6/05/2018 at 5:28 pm

This audio file displayed a relatively high overall magnitude but possessed a relatively low transitory magnitude between 0 – 1000 Hz and 1000 – 2000 Hz. In other words, there was a smaller change in average magnitude between the frequency ranges than expected. A bee buzzed loudly right next to the microphone halfway through the audio file, boxed in red above, producing a temporary spike in magnitude across all frequencies above 100 Hz and was responsible for the digression.

Another audio file recorded on June 1st at 9:55 AM deviated heavily from the expected average magnitude to transitory magnitude relationship. The spectrogram of that file is shown in Figure 25.

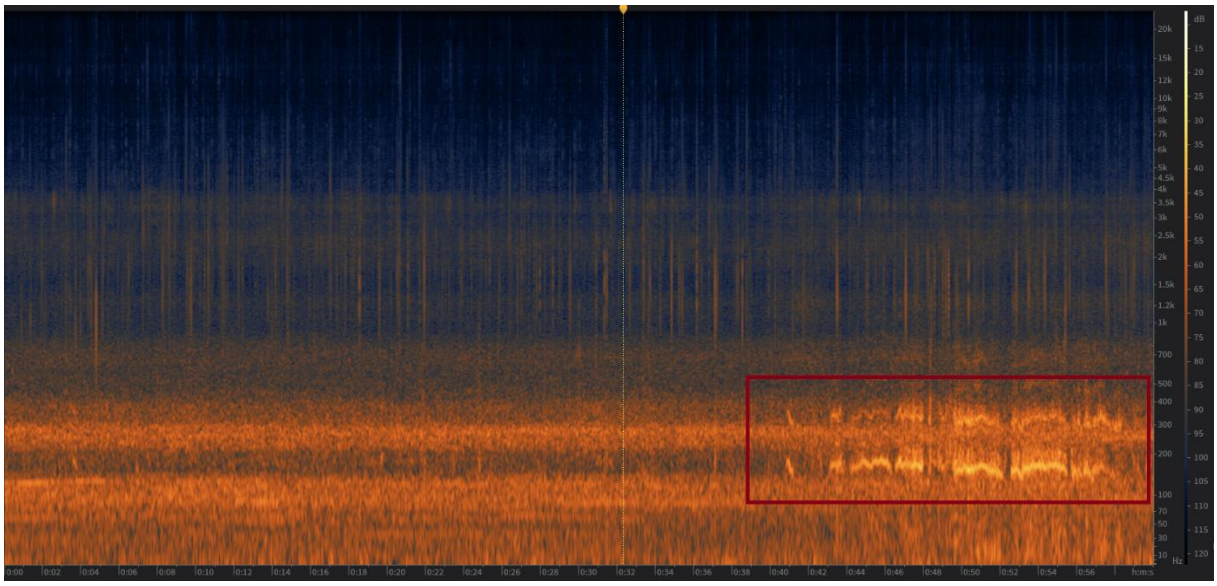

Figure 25: Spectrogram for audio file from RPI24 recorded on 6/01/2018 at 9:55 am

The audio file displayed a relatively large average magnitude compared to surrounding recordings but featured a significantly larger transitory magnitude than expected. In other words, the drop in average magnitude from the 0 – 1000 Hz range and the 1000 – 2000 Hz range was larger than expected. This can be explained by the last 20 seconds of the recording, during which a bee flew close to the microphone and generated a large amount of activity in the 100 – 400 Hz frequency range.

An audio file recorded on January 9th, 2019 at 9:36 AM also deviated heavily from the expected average magnitude to transitory magnitude relationship. The spectrogram of that recording is shown in Figure 26.

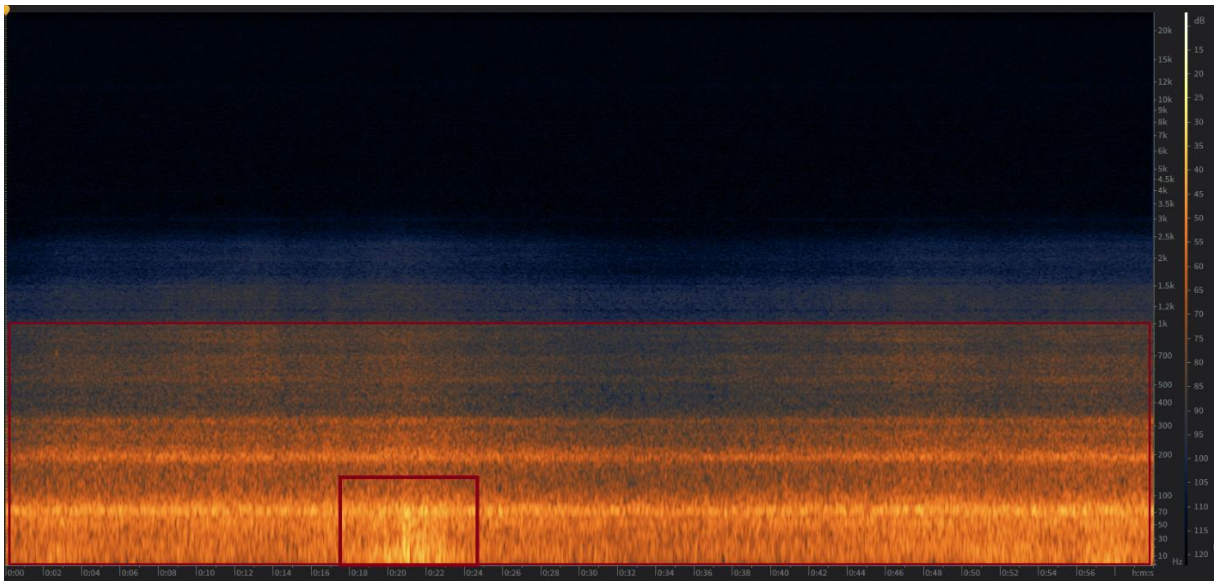

Figure 26: Spectrogram for audio file from RPI24-a recorded on 1/09/2019 at 9:36 am

A deep, low-frequency background hum was present throughout the recording. It briefly picked up in intensity between 20 and 23 seconds, boxed in red in the figure. No honey bee buzzes were heard throughout the recording. The overall recording had a moderately high average magnitude, but an extremely large transitory magnitude, as practically all of the relevant frequencies produced in the recording were at or below 1000 Hz. The humming had the highest magnitude between 0 – 300 Hz.

An audio recording taken on October 7th, 2018 at 2:27 PM exhibited moderate overall magnitude but had a lower-than-expected transitory magnitude. The spectrogram of the recording is shown in Figure 27.

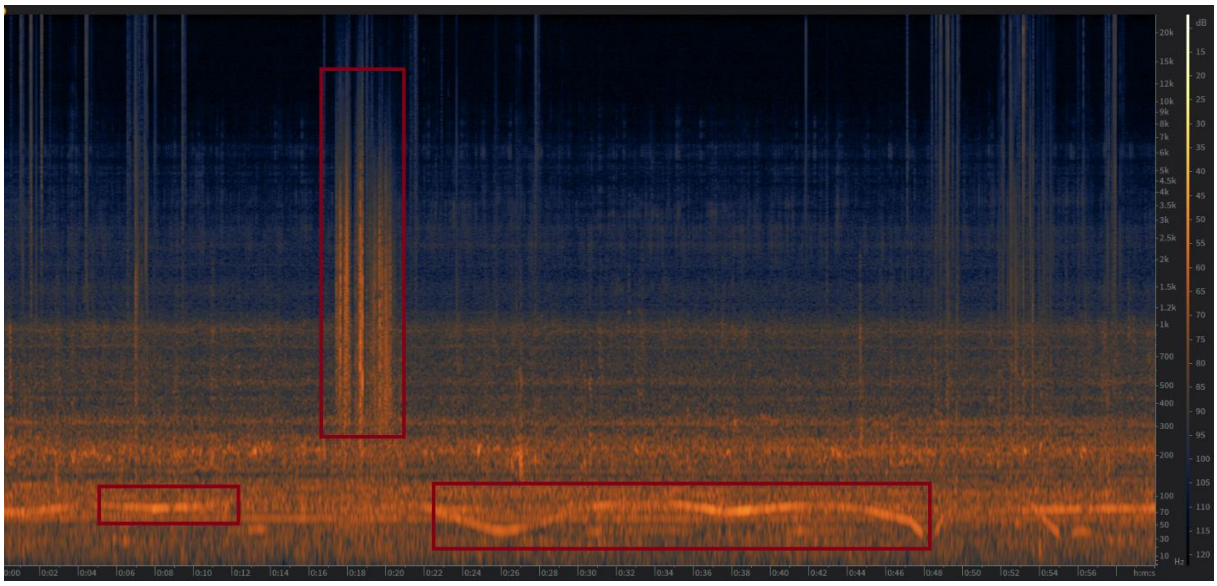

Figure 27: Spectrogram for audio file from RPI24-a recorded on 10/07/2018 at 2:27 pm

The spectrogram's unusual looking signals, boxed in red above, did not originate from honey bees, and instead were the result of modification of the project set up (i.e. movement of the microphone and video recorder).

An audio file from December 21st, 2018, recorded at 9:41 AM, strongly deviated from the expected average magnitude to transitory magnitude relationship, and produced a strange-looking spectrogram, shown in Figure 28.

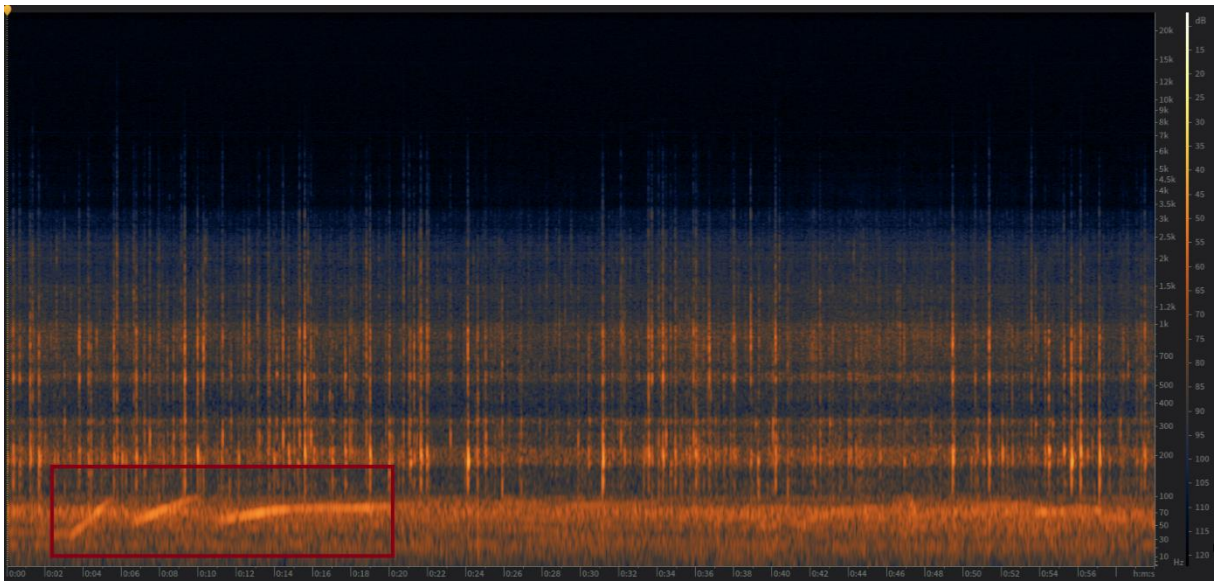

Figure 28: Spectrogram for audio file from RPI24-a recorded on 12/21/2018 at 9:41 am

Regular beats near the microphone were heard throughout the recording, as if the domestic hive setup was being modified or something was lightly hitting the hive near the microphone. No bee activity was heard throughout the recording.

The spectrogram of another audio recording with odd peaks due to equipment is shown in Figure 29. This audio recording was quickly identified after sorting the audio recordings from highest to lowest transitory magnitude and was the only recording within the top 20 from the month of July. The very large peaks all resulted from significant microphone movement.

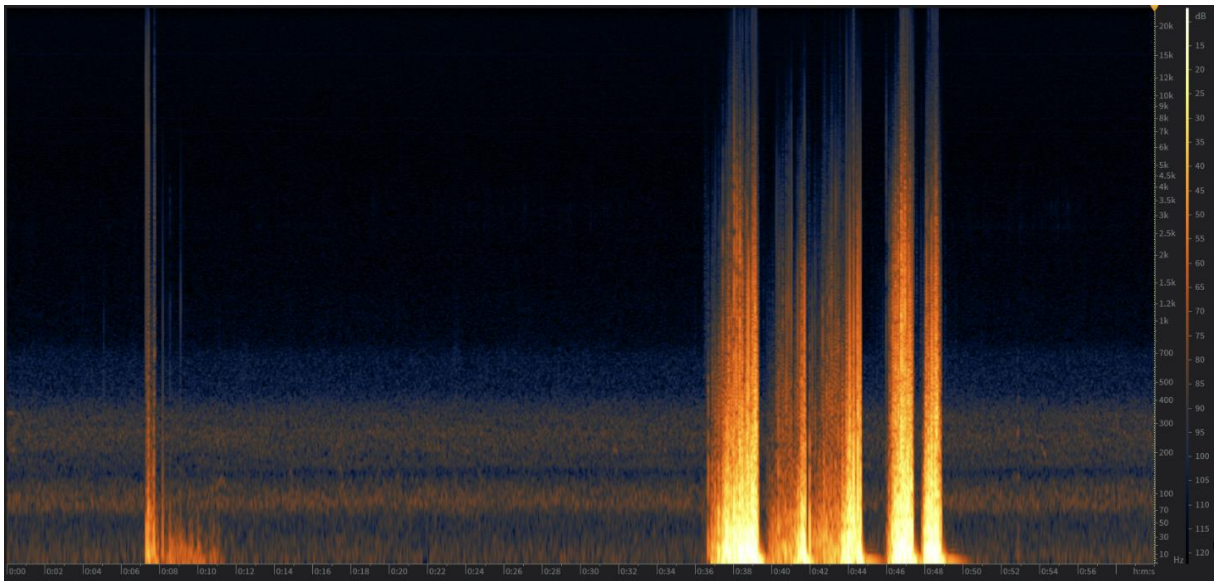

Figure 29: Spectrogram for audio file from RPI24 recorded on 7/16/2018 at 9:12 am

Another audio recording with very high transitory magnitude is shown in Figure 30. This recording is unique in that while there was constant honey bee buzzing throughout the recording (with a fundamental frequency of 300 Hz), microphone movement that generated intermittent large peaks was also present. Regular honey bee movement was present in the corresponding video file, though blurry.

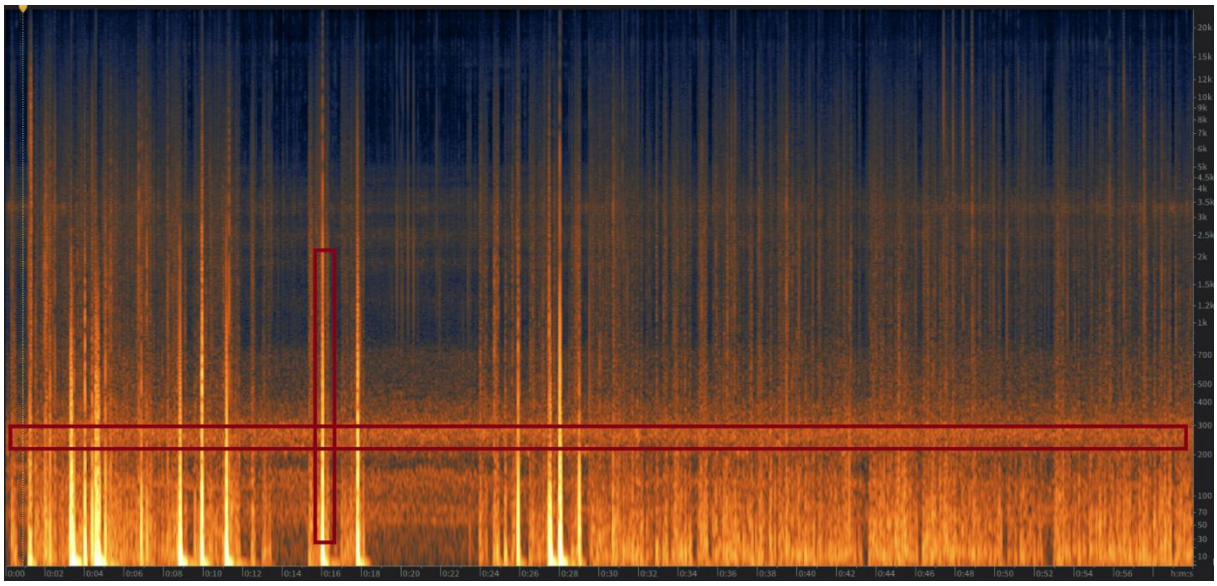

Figure 30: Spectrogram for audio file from RPI24 recorded on 6/19/2018 at 2:17 pm

Figure 31: Screenshot from RPI24 video file recorded on 6/19/2018 at 2:17 pm

Several RPI24 audio files that coincided with wasps flying outside the entrance were analyzed. Figures 32 and 33 show the spectrogram and corresponding video recording for one of these files, recorded on September 16th, 2017 at 10:05 AM.

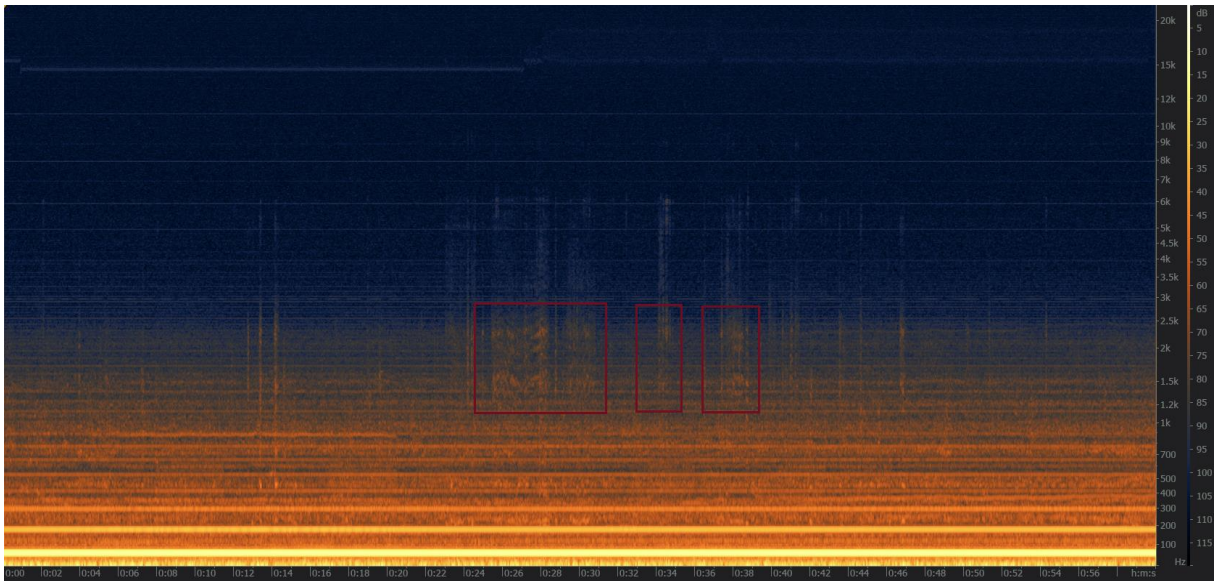

Figure 32: Spectrogram for audio file from RPI24 recorded on 9/16/2017 at 10:05 am

Figure 33: Screenshot from RPI24 video file recorded on 9/16/2017 at 10:05 am

The wasp that appeared at the hive entrance is boxed in Figure 33. The wasp was attacked by several bees shortly after appearing in the video recording, and not seen afterwards. The presence of the wasp seemed to coincide with the higher frequency signals boxed in Figure 32, suggesting that these signals were the result of the bees defending their hive from the intruder. The fundamental frequency of these signals was between 1200 and 1500 Hz, much

larger than honey bee sounds produced under normal hive conditions. This may be an example of the hissing signal, though this isn't definite.

3.3.3 RPI24 Hive Conclusions

All other viewed spectrograms from RPI24 that stood out quantitatively were similar to those described above. Very few strange instances of honey bee behavior were seen or heard in any of the recordings, and no unusual audio signals were detected between May 2018 and January 2019 in RPI24 using the HBAA or the DAS. RPI24 demonstrated relatively unstable levels of activity on a day-by-day basis in terms of average magnitude when compared to RPI11b, partly due to a more significant amount of propolis that covered the original microphone. Just as with RPI11b, the majority of audio recordings with unusual looking spectrograms were the result of equipment movement or bees moving near the microphone. Hissing was potentially present in the recordings where wasps arrived at the hive entrance, though these recordings were known to contain the wasps prior to this project and did not occur within the date range analyzed by the HBAA. No piping was detected in any of the audio recordings.

Chapter 4 – Conclusion

4.1 Summary

The HBAA did ultimately succeed in detecting unusual audio recordings, but not fully in the desired manner. The intent of the program was to quickly locate audio recordings that contained unusual honey bee signals, such as piping and hissing, by quantifying the amount of activity present in different frequency ranges. Though the HBAA did succeed in quantifying overall activity and activity within specific frequency ranges, the ultimate limitation ended up being the nature of the desired honey bee signals themselves.

Piping and hissing were the most desired signals within this project, as each could potentially indicate a change in colony behavior that would demand attention. However, as piping generally appears as a pulse sequence produced by a single bee at a time, the overall change in magnitude caused by the signal is likely small. Perhaps for this reason, the software was unable to isolate recordings that contained the signal, as something as common as microphone movement was likely to have had a greater effect on magnitude than piping. It's possible that none of the analyzed audio recordings contained piping, but also unlikely, due to the variety of conditions in which the signal is produced.

The software was also unable to isolate any hissing-containing recordings. It's possible that none of the audio recordings analyzed by the HBAA contained the signal, due to the specific hive-adverse conditions that are thought to be required for its production. As hissing is a broad brand signal produced by the entire colony, it's expected to have a more profound effect on average magnitude than piping. Hissing was potentially produced by the bees in a few recordings from 2017, though these recordings weren't within the project file date range and thus weren't analyzed by the HBAA.

The software was, however, able to quickly find recordings that possessed large amounts of overall honey bee activity, due to the fact that these recordings had large average magnitudes. It's possible that certain desired bee signals like piping were obfuscated in these spectrograms.

For the audio recordings analyzed, the change in average magnitude from 0 – 1000 Hz to 1000 – 2000 Hz was found to have a strong correlation with the overall average magnitude between 0 – 4000 Hz. This could in the future allow for the transitory magnitude to be used as a more environmentally-resistant indicator of overall activity when compared to average magnitude, which is more likely to be affected by variables such as wind and rain, though more research needs to be done on the subject. More research also needs to be conducted to determine how effective transitory magnitude is at comparing activity levels between hives that differ in microphone placement.

4.2 Future Work

Ultimately one of the problems with the methods of analysis utilized in this research is the fact that certain signals of interest, such as piping, appear only briefly and may be emitted by only a single bee at a time. Because of this, these signals likely don't raise the average magnitude to a significant degree, which makes them difficult to detect by the software. One possible solution to this limitation is modifying the software to detect changes in average magnitude within specific frequency ranges on a smaller time scale. In other words, rather than simply calculating the average magnitude within a frequency range for the entire audio recording, the average magnitude of that range for the first n seconds of the recording could be calculated and compared to the average magnitude within that range for the next n seconds, and on. With only a single frequency range being analyzed at once, and on a much smaller time scale than the entirety of the full recording, small changes in magnitude from high-frequency signals could ideally be more easily identified.

The software used in this research could also be employed in the future to determine exactly how much signals like piping and hissing affect magnitude, overall and within specific frequency ranges. An audio recording with known piping and that same recording with the piping removed could both be analyzed by the software and quantitatively compared. Then, depending on the difference in magnitudes between the two, the software could be enhanced to look for similar recordings. For example, if the presence of piping is found to on average increase the overall magnitude by 5%, the software could look for audio recordings that have a similar magnitude increase, compared to their predecessors. This would require a large number of recordings with known specific signals, however. Creating a database of known events with corresponding recordings is a possible solution that could help identify similar events in the future.

Bibliography

[1] G. Chavarria, "Pollinator Conservation," *Renewable Resources Journal*, vol. 17, pp. 18-22, 1999.

[2] O. Duran, D. Howard, G. Hunter, and K. Stebel, "Signal processing the acoustics of honeybees to identify the queenless state in hives," *Proceedings of the Institute of Acoustics*, vol. 35, pp. 290-297.

[3] J. Price, *Understanding dB*, 31-Jul-2007. [Online]. Available: http://www.jimprice.com/prosound/db.htm. [Accessed: 28-May-2019].

[4] R. G. Lyons, *Understanding Digital Signal Processing*. Prentice Hall, 2011.

[5] W. H. Kirchner, "Acoustical communication in honeybees," *Apidologie*, vol. 24, no. 3, pp. 297–307, 1993.

[6] S. C. Pratt, S. Kühnholz, T. D. Seeley, and A. Weidenmüller, "Worker piping associated with foraging in undisturbed queenright colonies of honey bees," *Apidologie*, vol. 27, no. 1, pp. 13–20, 1996.

[7] A. Qandour, I. Ahman, D. Habibi, and M. Leppard, "Remote Beehive Monitoring Using Acoustic Signals," *Acoustics Australia*, vol. 42, no. 3, 2014.

[8] A. Michelsen, W. H. Kirchner, B. B. Andersen, and M. Lindauer, "The tooting and quacking vibration signals of honeybee queens: a quantitative analysis," *Journal of Comparative Physiology A*, vol. 158, no. 5, pp. 605–611, 1986.

[9] C. Thom, D. C. Gilley, and J. R. Tautz, "Worker piping in honey bees (Apis mellifera): the behavior of piping nectar foragers," *Behavioral Ecology and Sociobiology*, vol. 53, no. 4, pp. 199–205, 2003.

[10] T. Seeley and J. F. Tautz, “Worker piping in honey bee swarms and its role in preparing for liftoff,” *Journal of Comparative Physiology A: Sensory, Neural, and Behavioral Physiology*, vol. 187, no. 8, pp. 667–676, 2001

[11] M. S. Sarma, S. Fuchs, C. Werber, and J. Tautz, “Worker piping triggers hissing for coordinated colony defence in the dwarf honeybee Apis florea,” *Zoology*, vol. 105, no. 3, pp. 215–223, 2002.

[12] iZotope RX 6. (2017). iZotope, Inc.

[13] M. Crawford, “Automated Collection of Honey Bee Hive Data using the Raspberry Pi”, M.S. Thesis, Dpt. of Computer Science, Appalachian State Univ., Boone, NC, 2017

[14] D. Kale, “Automated Beehive Surveillance Using Computer Vision,” M.S. Thesis, Dpt. of Computer Science, Appalachian State Univ., Boone, NC, 2015.

[15] R. Oliver, "Understanding Colony Buildup and Decline: Part 13a - Scientific Beekeeping", Scientific Beekeeping, 2016. [Online]. Available: http://scientificbeekeeping.com/understanding-colony-buildup-and-decline-part-13a/. [Accessed: 04- Jun- 2019].

Appendix – Complete Code for Programs

AudioAnalyzer.py

```
from __future__ import division

__author__ = 'Preston Wilson'


import os
import numpy as np
from scipy import signal
from matplotlib import pyplot as plt
import glob
import wavio

'''
Credit for part of the Spectrum Algorithm to user edwin and user Frank Zalkow on
Stack Overflow from:
http://stackoverflow.com/questions/35636426/plot-spectrum-of-a-wave-as-in-audacity
'''

AA_LOWER_LIMIT = -130  # constant lower limit y-value


# creates (frequency, value) points from passed signal values
def plot_freq(y, rate=44100, s=32768, title="null", dbfs=False):
    # Plots Amplitude Spectrum for y(t)
    # param y: Signal
    # param rate: Sampling rate, default 44100 Hz (number of samples of audio
carried per second)
    # param s: Sampling size, default 32768
    # param title: Title of spectral plot
    plt.style.use('ggplot')
    plt.figure(num=1, figsize=(9, 5), dpi=80)
    plt.subplot(1, 1, 1)  # call new subplot for spectrogram (which is never shown)

    window = signal.hanning(s)
    spectrum, freqs, bins, _ = plt.specgram(y, NFFT=s, Fs=rate, noverlap=s*0.5,
                                            cmap=plt.cm.binary,
                                            sides='onesided',
                                            window=window,
                                            mode='magnitude')
    # plt.show()

    average_magnitude_across_all_periodograms = np.mean(spectrum, axis=1)
    magnitudes = (average_magnitude_across_all_periodograms * 2) / np.sum(window)
    if dbfs:
        magnitudes_dbfs = compute_dbfs(magnitudes, 32767.0)  # 32767 is the maximum
integer that 16bit wav file uses to
        create_plot(freqs, magnitudes_dbfs, title)
        return freqs, magnitudes_dbfs
    else:
        magnitudes_avg = magnitudes / 32767.0
        # store amplitude. 32767 = 0 dBFS, it is 100% loud (0 in a 16bit wav file =
-infinity loud)
        create_plot(freqs, magnitudes_avg, title)
        return freqs, magnitudes_avg
```

```
# creates plot from (freq, dBFS) points that will be saved locally
def create_plot(freq, mag_dbfs, title):
    plt.cla()  # clear the spectrogram subplot. Gets info from it above and then is
cleared here.
    plt.subplot(1, 1, 1)  # start anew without spectrogram. Got info from it, don't
want to plot it.
    ax = plt.gca()
    ax.grid(True)
    plt.ylim([-130, 0])  # -96dBFS is the noise floor for 16bit audio,
    # but our recordings are so quiet we have to go below that.
    plt.xlim([0, 4000])

    plt.plot(freq, mag_dbfs, '-', color='#720e22', drawstyle='steps-mid')

    plt.title(title)
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Intensity (dBFS)')


# computes dbfs values from passed data
def compute_dbfs(data, max_integer):
    dbfs_data = 20.0 * np.log10(data / max_integer)
    return dbfs_data


# calculates average dBFS value for each 1000-Hz frequency range interval, writes
output to log and console
def avg_decibels(plot_vals):
    x_plot_vals = plot_vals[0]
    y_plot_vals = plot_vals[1]

    write_data = []

    print 'X-vals: ', x_plot_vals
    print 'Y-vals: ', y_plot_vals
    print 'Length: ', len(x_plot_vals)

    under_thousand_vals = []
    one_two_thousand_vals = []
    two_three_thousand_vals = []
    three_four_thousand_vals = []
    all_vals_under_four = []

    for i in range(0, len(x_plot_vals)):

        if x_plot_vals[i] < 1000:
            under_thousand_vals.append(y_plot_vals[i])
            all_vals_under_four.append(y_plot_vals[i])
        if 1000 <= x_plot_vals[i] < 2000:
            one_two_thousand_vals.append(y_plot_vals[i])
            all_vals_under_four.append(y_plot_vals[i])
        if 2000 <= x_plot_vals[i] < 3000:
            two_three_thousand_vals.append(y_plot_vals[i])
            all_vals_under_four.append(y_plot_vals[i])
        if 3000 <= x_plot_vals[i] < 4000:
            three_four_thousand_vals.append(y_plot_vals[i])
            all_vals_under_four.append(y_plot_vals[i])

    avg_under_thousand = 0
    avg_one_two = 0
    avg_two_three = 0
    avg_three_four = 0
    avg_overall = 0
```

```
    # calculates average decibel value within frequency ranges
    try:
        avg_under_thousand = sum(under_thousand_vals) /
float(len(under_thousand_vals))
    except Exception as i:
        print('Possibly no frequency values under 1000: ', i)
    try:
        avg_one_two = sum(one_two_thousand_vals) /
float(len(one_two_thousand_vals))
    except Exception as i:
        print('Possibly no frequency values 1000-2000: ', i)
    try:
        avg_two_three = sum(two_three_thousand_vals) /
float(len(two_three_thousand_vals))
    except Exception as i:
        print('Possibly no frequency values 2000-3000: ', i)
    try:
        avg_three_four = sum(three_four_thousand_vals) /
float(len(three_four_thousand_vals))
    except Exception as i:
        print('Possibly no frequency values 3000-4000: ', i)
    try:
        avg_overall = sum(all_vals_under_four) / len(all_vals_under_four)
    except Exception as i:
        print 'Possibly no frequency values at all: ', i

    print 'Average under 1000 Hz: %.5f' % avg_under_thousand
    write_data.append('\nAverage under 1000 Hz: %.5f' % avg_under_thousand + '\n')

    print 'Average 1000 - 2000 Hz: %.5f' % avg_one_two
    write_data.append('\nAverage 1000 - 2000 Hz: %.5f' % avg_one_two + '\n')

    print 'Average 2000 - 3000 Hz: %.5f' % avg_two_three
    write_data.append('\nAverage 2000 - 3000 Hz: %.5f' % avg_two_three + '\n')

    print 'Average 3000 - 4000 Hz: %.5f' % avg_three_four
    write_data.append('\nAverage 3000 - 4000 Hz: %.5f' % avg_three_four + '\n')

    print 'Average Hz overall: %.5f' % avg_overall
    write_data.append('\nAverage Hz overall: %.5f' % avg_overall + '\n')

    avg_dbfs = [avg_under_thousand, avg_one_two, avg_two_three, avg_three_four,
avg_overall]

    return avg_dbfs, write_data


# supplemental video analysis, optional and not currently used in generation of
spectral plots
def vid_evaluator(vid_file):
    try:
        file_size = os.path.getsize(vid_file)
        mb_file_size = (float(file_size) / 1048576)

        # amount of light in video file could factor into size calculation,
creating less accuracy.
        print('\nNOTE: Video analysis is based entirely on file size. \n'
              'Under 20 MB: Small activity.\n'
              '20 - 30 MB: Moderate activity.\n'
              '30 - 40 MB: Large activity.\n'
              'Above 40 MB: Very large activity.\n'
              'Your video file size: %.5f' % mb_file_size)
```

```
        if mb_file_size < 10:
            print('Small amounts of activity expected based on video size (ignore
if no video input given)')
        elif 10 <= mb_file_size < 30:
            print('Moderate amounts of activity expected based on video size
(ignore if no video input given)')
        elif 30 <= mb_file_size < 40:
            print('Large amounts of activity expected based on video size (ignore
if no video input given)')
        elif 40 <= mb_file_size:
            print('Very large amounts of activity expected based on video size
(ignore if no video input given)')
    except Exception as e:
        print('no video file or non-existent video file supplied to evaluator,
defaulting')


# calculates total audio activity score, summation of all 1000-Hz frequency
interval average intensities;
# calculates differences in average intensity between the first three 1000-Hz
frequency intervals for
# interval transition analysis
def aud_evaluator(decibel_list):
    total_audio_activity = 0
    numeric_data = []
    write_data = []
    for i in range(0, 4):
        total_audio_activity += decibel_list[i]

    tot_aud = 'Total audio activity score: %.5f' % total_audio_activity
    numeric_data.append(total_audio_activity)
    write_data.append('\n' + tot_aud + '\n')
    print(tot_aud)

    first_dif = 'Difference between first two ranges: %.5f' % (decibel_list[1] -
decibel_list[0])
    numeric_data.append(decibel_list[1] - decibel_list[0])
    write_data.append('\n' + first_dif + '\n')
    print(first_dif)

    second_dif = 'Difference between second two ranges: %.5f' % (decibel_list[2] -
decibel_list[1])
    numeric_data.append(decibel_list[2] - decibel_list[1])
    write_data.append('\n' + second_dif + '\n\n')
    print(second_dif)

    transition_one = decibel_list[1] - decibel_list[0]
    transition_two = decibel_list[2] - decibel_list[1]

    print '\nActivity Analysis based on average decibel value change in frequency
range transition:'
    print '(Usually the transition from Under 1k to 1-2k, and 1-2k to 2-3k is most
significant)'
    print 'Change from Under 1000 range to 1000-2000 range: %.5f' % transition_one
    print 'Change from 1000-2000 range to 2000-3000 range: %.5f' % transition_two

    if transition_one < -10:
        if transition_two < -8:
            print('Large to very large amounts of activity based on transition
analysis')
        else:
            print('Moderate amounts of activity based on transition analysis')
    else:
```

```
            print('Small amounts of activity based on transition analysis')
    return numeric_data, write_data


# takes all data generated from the other functions, and writes out the numerical
data only
def output_to_file_data(aud_file, filename, avg_decibels_data, aud_evaluator_data):
    aud_file.write(filename + ",")
    for avd in avg_decibels_data:
        aud_file.write(str(avd) + ",")
    for aed in aud_evaluator_data:
        aud_file.write(str(aed) + ",")
    aud_file.write("\n")


# takes all data generated from the other functions, and writes it all to the
output file
def output_to_file_text(aud_file, filename, avg_decibels_print,
aud_evaluator_print):
    aud_file.write("Detailed Values for Audio Files in Directory\n")
    aud_file.write('\n~~~Details for ' + filename + '~~~\n')
    for d in avg_decibels_print:
        aud_file.write(d)
    for j in aud_evaluator_print:
        aud_file.write(j)


# function to return all directories containing rpi
# takes base directory containing all rpi directories
def get_rpis(directory, rpi="rpi"):
    rpi_dirs = []
    for i in directory:
        if os.path.isdir(i) and rpi in i:
            rpi_dirs.append(i)
    return rpi_dirs


# function to return all directories containing a date
# requires a full original path, along with rpi directory
# e.g. C:\Users\wilsonpa5\Documents and rpi21
def list_dates(basic_path, directory):
    date_dirs = []
    new_directory = os.path.join(basic_path, directory)
    for i in os.listdir(new_directory):
        if os.path.isdir(os.path.join(new_directory, i)):
            date_dirs.append(i)
    return date_dirs


# gets all passed dates after a start date and before an end date
# requires a simple list of dates: not full path names
# e.g. ["2016-07-04", "2017-05-06"]
# requires format YYYY-MM-DD to work properly, and returns them as a list
def get_dates_within_time(date, start_time, end_time):
    split_start = start_time.split('-')
    start_year = split_start[0]
    start_month = split_start[1]
    start_day = split_start[2]
    split_end = end_time.split('-')
    end_year = split_end[0]
    end_month = split_end[1]
    end_day = split_end[2]
    dates_within_range = []
```

```
    for d in date:
        after_start = False
        before_end = False
        split_date = d.split('-')  # split date is current directory date
        if int(split_date[0]) >= int(start_year):  # if current directory year is
not less than start year
            if int(split_date[0]) > int(start_year):  # current directory is after
start time!
                after_start = True
            else:  # same year date
                if int(split_date[1]) > int(start_month):
                    after_start = True
                elif int(split_date[1]) == int(start_month):  # same month in same
year
                    if int(split_date[2]) >= int(start_day):
                        after_start = True
        if int(split_date[0]) < int(end_year):
            before_end = True
        elif int(split_date[0]) == int(end_year):  # current directory year is same
as end year
            if int(split_date[1]) < int(end_month):
                before_end = True
            elif int(split_date[1]) == int(end_month):  # current directory month
is same as end month
                if int(split_date[2]) <= int(end_day):
                    before_end = True

        if after_start and before_end:
            dates_within_range.append(d)
    return dates_within_range


# creates plots for all files in a given directory and saves them without showing
them one by one;
# calculates audio and interval transition scores.
# by default will evaluate files in every hour
def evaluate_directory(global_data_file, aud_directory='', hours=range(24),
to_dbfs=False):
    try:
        # split directory into each level
        new_dir_list = aud_directory.split('\\')
        new_dir = ''  # directory for the output log to be stored in
        for i in range(0, len(new_dir_list) - 1):
            new_dir += str(new_dir_list[i]) + '\\'

        details_file = open(new_dir + 'AudioFilesDetails.log', 'w')  # plot details
will be placed in here

        avg_intensity_summation = 0
        num_days = 0

        for aud_file in glob.iglob(aud_directory):
            split_aud = aud_file.split('\\')
            time_only = split_aud[len(split_aud) - 1]
            hour_only = time_only.split('-')[0]
            if int(hour_only) in hours:
                all_info = wavio.read(aud_file)
                data = all_info.data.flatten()
                rate = all_info.rate
                sampling_width = all_info.sampwidth

                print '\n~~~~~Start of New Audio File~~~~~~\nAudio file = ',
aud_file
```

```
                try:
                    title_parts = aud_file.split('\\')
                    formatted_title = title_parts[len(title_parts) - 2] + '-at-' +
title_parts[len(title_parts) - 1]
                except Exception as e:
                    formatted_title = aud_file

                plot2 = plot_freq(data, rate, 32768, title=formatted_title,
dbfs=to_dbfs)

                if not os.path.exists(new_dir + 'UpdatedPlots/'):
                    os.makedirs(new_dir + 'UpdatedPlots/')
                formatted_title = formatted_title.split('.')[0]
                plt.savefig(new_dir + 'UpdatedPlots/' + '-
'.join(formatted_title.split(" ")).replace(':', '-') + '.png')
                print('||| Information for how frequent values are found in certain
frequency ranges '
                      'for audio file (x-axis) |||\n')
                a_v_data, a_v_print = avg_decibels(plot2)
                avg_intensity_summation += a_v_data[4]
                num_days += 1
                a_e_data, a_e_print = aud_evaluator(a_v_data)

                output_to_file_text(details_file, aud_file, a_v_print, a_e_print)
# write everything to output file
                output_to_file_data(global_data_file, aud_file, a_v_data, a_e_data)
# same but only for numbers

        avg_intensity_day = avg_intensity_summation / num_days
        avg_intense_string = '\nAverage Intensity for day: ' +
str(avg_intensity_day)
        print avg_intense_string
        details_file.write(avg_intense_string)

    except Exception as e:
        print('Unable to evaluate given directory - Check the path is appropriate:
', e)


# main function that executes all instructions and puts everything together
# pass desired parameters here to analyze specific files
def execute_all(rpi="rpi22", start_date="2016-05-01", end_date="2016-05-02",
hours=[8], dbfs_con=False, drive='E:'):
    try:
        # usage case example: rpi21\2016-07-01\*.wav
        # use backslashes for windows, along with file type in all cases
        all_dirs = os.listdir(".")
        print all_dirs
        rpi_dirs = get_rpis(all_dirs, rpi)
        print rpi_dirs

        basic_dir = drive + "\\Dehummed_Thesis\\"  # change as needed
        final_path_dates = []  # full directories for all date directories
        for i in rpi_dirs:
            inner_dates = list_dates(basic_dir, i)
            for k in get_dates_within_time(inner_dates, start_date, end_date):
                print k
                final_path_dates.append(os.path.join(basic_dir, i, k, "audio"))
        print final_path_dates

        # plot details but only numbers
        data_file =
open("C:\\Users\\wilsonpa5\\Documents\\AudioAnalyzerv2\\NumericDataDetails.csv",
```

```
'w')

data_file.write("FILENAME,AVG_UNDER_THOUSAND,AVG_ONE_TWO,AVG_TWO_THREE,AVG_THREE_FO
UR,AVG_OVERALL,"
                        "TOTAL_AUDIO_ACTIVITY,FIRST_DIF,SECOND_DIF\n")
        for i in final_path_dates:
            evaluate_directory(data_file, i + "\\*.wav", hours, to_dbfs=dbfs_con)
        data_file.close()
    except Exception as e:
        print 'Exception: ', e


if __name__ == '__main__':
    # pass arguments here for specific date range/hour range to check
    # param rpi: the hive to check
    # param start_date: the starting day (inclusive) to check in YYYY-MM-DD format
    # param end_date: the ending day (inclusive) to check in YYYY-MM-DD format
    # param hours: a list of the hours to check; any audio files starting at an
included hour will be analyzed
    # param dbfs_con: optionally convert raw audio values to dBFS; useful for plot
creation, not for direct comparison
    execute_all(rpi="rpi24", start_date="2018-06-22", end_date="2018-06-22",
hours=[9, 11, 14, 17], dbfs_con=True, drive='E:')
```

DataAnalysis.py

```
from matplotlib import pyplot as plt
import pandas as pd
import seaborn as sns
import scipy as scp

__author__ = 'Preston Wilson '


if __name__ == '__main__':
    bee_date = pd.read_csv("RPI24_ALL_DATA_THESIS.csv", index_col=0)
    x_axis = 'FIRST_DIF'
    y_axis = 'AVG_OVERALL'
    dif = bee_date[x_axis]
    aove = bee_date[y_axis]

    x_list = dif.values.tolist()
    y_list = aove.values.tolist()

    # plt.figure()
    # print sdif, aove
    # plt.plot(sdif, aove)

    # Show the results of a linear regression within each data set
    pl = sns.lmplot(x=x_axis, y=y_axis, data=bee_date)
    pl.set(xlim=(min(dif.values.tolist()), max(dif.values.tolist())),
           ylim=(min(aove.values.tolist()), max(aove.values.tolist())))

    slope, intercept, r_value, p_value, std_err = scp.stats.linregress(dif, aove)

    all_y_difs = []
    total_vals = 0

    for i in range(0, len(y_list)):
        expected_y = float(slope) * float(x_list[i]) + float(intercept)
        actual_y = float(y_list[i])
        all_y_difs.append([abs(actual_y - expected_y), i, bee_date.index[i]])
        total_vals += 1

    print 'number of values: ', total_vals
    print 'slope: ', slope
    print 'intercept: ', intercept
    print 'r_value: ', r_value
    print 'r_squared: ', (r_value * r_value)
    print 'p_value: ', p_value
    print 'std_err: ', std_err

    for i in sorted(all_y_difs, reverse=True):
        print i
    ax = plt.gca()
    ax.set_title('RPI24 Transition in Magnitude vs Overall Average Magnitude
(5/01/2018 - 1/31/2019, n=11752)',
                 fontsize=25)
    ax.set_xlabel('Change in Average Magnitude from 0 - 1000 Hz to 1000 - 2000 Hz',
fontsize=22)
    ax.set_ylabel('Overall Average Magnitude <4000 Hz', fontsize=22)
    ax.tick_params(labelsize=18)
    plt.show()
    plt.savefig('CSV_Scatterplot.png')
```