The Effect of Varying Parameters on Web Vibration Analysis Accuracy

By
Kathleen Gladson

A THESIS

submitted to
Oregon State University
University Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Mechanical Engineering
(Honors Scholar)

Honors Baccalaureate of Arts in International Studies
(Honors Scholar)

Presented May 29, 2019
Commencement June 2019
AN ABSTRACT OF THE THESIS OF

Kathleen Gladson for the degree of Honors Baccalaureate of Science in Mechanical Engineering and Honors Baccalaureate of Arts in International Studies presented on May 29, 2019. Title: The Effect of Varying Parameters on Web Vibration Analysis Accuracy

Abstract approved:

____________________________________________________________________________________

Ross Hatton

The SpiderHarp is a bio-inspired electronic musical instrument that uses a localization algorithm developed at Oregon State University to translate plucks of its strings into musical notes. This localization algorithm was inspired by the way spiders locate prey in their webs. The SpiderHarp frame, sensors, and web design are also used to model the travel of vibrations in actual spider webs. In order to improve the localization algorithm and determine the web tension settings that lead to highest accuracy of the localization algorithm, the algorithm was tested at varying radial and spiral web tensions as well as various cutoff parameters for the window of data being used by the algorithm.

These experiments showed that restricting the data used by the localization algorithm to a short time period around the initial sensing of the pluck increases the accuracy of the algorithm. Also, these experiments showed that the localization algorithm was able to more accurately determine the distance of the pluck from the
center of the web when the web spiral tension was lower.

Key Words: spider webs, vibrations, string vibration, web vibration

Corresponding e-mail address: gladsonk@lifetime.oregonstate.edu
The Effect of Varying Parameters on Web Vibration Analysis Accuracy

By
Kathleen Gladson

A THESIS

submitted to
Oregon State University
University Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Mechanical Engineering
(Honors Scholar)

Honors Baccalaureate of Arts in International Studies
(Honors Scholar)

Presented May 29, 2019
Commencement June 2019
Honors Baccalaureate of Science in Mechanical Engineering and Honors Baccalaureate of Arts in International Studies project of Kathleen Gladson presented on May 29, 2019

APPROVED:

________________________
Ross Hatton, Mentor, representing Mechanical Engineering

________________________
Chet Udell, Committee Member, representing Biological and Ecological Engineering

________________________
Rebekah Lancelin, Committee Member, representing International Studies Degree Program

________________________
Toni Doolen, Dean, Oregon State University Honors College

I understand that my project will become part of the permanent collection of Oregon State University Honors College. My signature below authorizes release of my project to any reader upon request.

________________________
Kathleen Gladson, Author
## Contents

1 Introduction 3

1.1 Objective .................................................. 4

2 Background 5

2.1 Global Research ............................................. 5

3 Methods 8

3.1 Web Creation ................................................ 8
3.2 Testing Setup ................................................. 10
3.3 Data Collection .............................................. 14
    3.3.1 Localization Algorithm ................................. 16

4 Results 17

4.1 Results of Algorithm Parameter Variation ................. 18
4.2 Results of Web Tension Variation ......................... 25

5 Discussion 33

5.1 International Impact ....................................... 35

6 Conclusion 36

6.1 Future Work ................................................. 37

References 37
A Appendix: Web Building Guide

B Appendix: MATLAB Code

B.1 Web Build Instruction Generation ........................................... 53
B.2 Web Geometry Generation ...................................................... 55
B.3 Web Build Plot Generation ..................................................... 57
B.4 Converting Segment Number to Radial and Spiral Number ........... 59
B.5 Selecting Pluck Points .......................................................... 59
B.6 Pluck Localization and Analysis Main Script ............................... 60
B.7 Pluck Localization Algorithm .................................................. 62
B.8 Localization Analysis ........................................................... 65
1 Introduction

In 1880, C. V. Boys published an article documenting observations of spiders responding to the vibrations from a tuning fork [1]. Boys, who was from the Physical Laboratory in South Kensington, England, noticed that when a vibrating tuning fork was touched to a spider’s web, the spider would respond as if the tuning fork was prey.

Now, many spiders are known to use vibrations rather than sight to locate and capture prey [2]. While some spiders have also been observed creating and responding to vibration for courtship and responding to vibrations in other media such as leaves, air, and water, vibrations in spider webs are of particular interest because of their unique traits as intricate, tensioned nets that both trap prey and allow spiders to locate their prey. For over a century researchers have been documenting spider behavior. Meanwhile, other researchers have been experimenting to determine how vibrations travel through webs and how spiders sense these vibrations.

At Oregon State University, Andrew Otto and Professor Ross Hatton built a frame for holding a 4 foot diameter artificial web to use for experiments to validate theoretical models of vibration travel through webs. They made artificial webs and designed a localization algorithm to determine where a disturbance to the web originated using a set of eight accelerometers, each attached to a different radial strand near the center of the web. One of the major goals of this research is to help biologists better understand spider behavior in regards to how spiders find their prey and why spiders
choose specific tensions for different parts of their web. However, this project has also expanded to the field of music. They created an electronic instrument, dubbed the SpiderHarp, which uses the artificial spider web vibration modeling setup and a version of the localization algorithm to determine where the web has been plucked and play a note corresponding to that location on speakers in real time.

The reason for building an artificial, large-scale web testing setup was to allow for testing a variety controlled web geometries while also being able to control web tension and have sensors physically contact the web. The robustness of the artificial webs allow for the same web to be repeatedly tested without significant damage while the control over tension and geometry allows for easier correlation between the theoretical model web parameters and the physical testing web parameters.

1.1 Objective

This thesis aims to determine how different adjustable parameters affect the accuracy of the localization algorithm. The experiments test varying spiral tension and radial tension in the web and varying the amount of time before and after the pluck to cutoff data given to the localization algorithm. Then, the localization algorithm ran on each combination of test parameters. The results of the localization algorithm was compared to the known pluck locations to determine the accuracy of the localization algorithm for each combination of parameters. Analysis of the effect of changing the parameters on the accuracy of the localization algorithm will allow for optimization
of the SpiderHarp web and the SpiderHarp localization algorithm.

2 Background

In “A bioinspired adaptive spider web,” Zheng et al wrote about experimenting on artificial spider web models to replicate the effect web damage has on the energy absorption of the web [3]. They experimented with the web both when the web tension was actively and passively controlled.

In “Transmission of vibrations in funnel and sheet spider webs,” Naftilan observed the webs of a funnel-web-building spider [4]. Naftilan modeled the funnel web as a thin sheet and used the model to compare vibrations in sheet webs to vibrations in funnel webs.

2.1 Global Research

Just as spiders are found around the world, vibrations in spider webs are studied around the world. For instance, in South Korea, the National Research Foundation of Korea funded Eunseok Jeong and DaeEun Kim to complete research on vibrations in spider web [5][6]. As described in “A Bio-Inspired System to Detect String Vibration,” Jeong and Kim created a rectangular, grid-based artificial web out of optic fiber and used it to conduct experiments on how the number of intersecting strands affects the frequencies that are filtered out by the web [5]. They also measured the differences between how transverse and longitudinal vibrations travel through the
web. As described in “Detecting the Vibration in the Artificial Web Inspired by the Spider,” Jeong and Kim used a similar set up with a web made out of fishing line to test bioinspired vibration sensors modeled after the vibration sensors in spider legs [6]. With these sensors, they were able to determine the orientation of the vibration source.

Additionally, in the United Kingdom, Beth Mortimer has published numerous papers on vibrations in spider webs. “Unpicking the signal thread of the sector web spider Zygiella x-notata” and “Remote monitoring of vibrational information in spider webs” investigate spiders using a signal thread to sense web vibrations while hiding in a corner of the web [7], [8]. “Biotremology: Do physical constraints limit the propagation of vibrational information?” explains how vibrations are used by a wide variety of animals, including spiders [9]. “Tuning the instrument: sonic properties in the spiders web” suggests that spiders may use adjustable parameters, such as web tension, to assist them in vibration sensing [10].

Friedrich Barth, who was based out of Germany and, more recently, Austria, has written and contributed to numerous papers. “Spiders of the genus Cupiennius Simon 1891 (Araneae, Ctenidae)” examines the different types of vibrations felt by a species of spiders without webs that uses the vibrations it feels through plant leaves to sense the world around it [11]. “Viscoelastic nanoscale properties of cuticle contribute to the high-pass properties of spider vibration receptor (Cupiennius salei Keys)” investigates the mechanical properties of the vibration receptors in spider legs and how these
mechanical properties might filter out vibrations above one frequency threshold and below another [12].

Samuel Zschokke, based out of Switzerland, and Kensuke Nakata, based out of Japan, have collaborated together on several papers including “Spider orientation and hub position in orb webs,” where they examine the orientation and location a spider waits in while preparing for insects to get caught in the web [13]. While they are examining this issue from the perspective of the spiders prey catching ability, asymmetry due to the hub not being at the geometric center of the web also may affect the way vibrations travel through the web. Nakata also wrote “Spatial learning affects thread tension control in orb-web spiders,” where he observed spiders using their legs to increase the tension in parts of the web that the researchers would provide with prey [14].

International collaborations are common in spider web vibration research. Researchers use their differing areas of expertise to help each other better understand spider behavior, spider anatomy, spider web mechanics, and vibration travel. Researchers also collaborate with other researchers and reference the work of other researchers to compare different varieties of spiders found in different parts of the globe. This allows researchers to compare spiders with orb webs, spiders with funnel webs, and spiders without webs.
3 Methods

3.1 Web Creation

While many artificial webs have already been made for use in Andrew Otto and Professor Ross Hatton’s research, inconsistencies between web builders and a lack of precise and accurate measurements had lead to inconsistencies in correlation to the theoretical web vibration models. As a result, care was taken to make the webs used in the following experiments as accurately and consistently as possible. Both webs used in the experiments were made by Kathleen Gladson according to a detailed set of instructions that can be found in Appendix A.

The first step in building a web is to select the desired web geometry parameters and use the web generator code found in Appendix B.2 to calculate the length of each segment in the web. This web generator allows the web to be built without tension in any of the segments. This is important because building the web under tension would add much more difficulty to the building process, increasing the likelihood of error. To adjust the final tension in the spirals when the web radials are fully tensioned, the untensioned web is built in a conical shape so the spiral segments need to stretch farther for the web to become flat. This means that webs with a steeper build angle have greater spiral tension. The inputs to the web generator code are the final desired radius of the web when tensioned, the radius of the innermost end of the spiral when untensioned, the radius of the outermost end of the spiral when untensioned, the
number of radial lines, the number of spiral turns, the angle of the cone that the web is built in under no tension, and the stretch factor of the rope used to build the radials.

The test webs were constructed of parachute cord for the radials and bungee cord of a similar diameter for the spirals in order to mimic the greatly differing elasticity between radial strands and spiral strands in spiderwebs. The radials were tied to a 1 inch diameter keyring to create the web center. The output of the web generator script was manipulated with the script in Appendix B.1 to get a list of distances between intersection points along one long spiral strand and each of the radial strands.

The radials were tied to the center ring and superglue was used to keep the knot from slipping over time. Then, each strand was hand-marked with the intersection points. The spiral intersection points were also measured and marked. Next, the spiral was attached to the radials at each intersection point using zipties. The zipties were arranged so that they all pointed the same direction. Superglue was used at each ziptied intersection point to ensure the zipties would not slip when the web was under tension.

Two webs, each with different geometries, were built for testing. Both webs were designed to have an outer radius of 24 inches when tensioned, 8 radials, 16 spirals, and a radial stretch factor of 7%. Web 1 was designed to have a 45 degree build cone angle and Web 2 was designed to have a 0 degree build cone angle as shown in the web plots in Figure 1. This means that Web 2 was flat even when untensioned
Figure 1: Plots showing the theoretical build geometry of the (a) 45° build angle for higher tension spirals and the (b) 0° build angle for lower tension spirals. They were plotted using the code found in Appendix B.3.

but Web 1 was only flat when tensioned in the web frame. Web 1 was designed to have an untensioned outermost spiral endpoint radius of 13 inches and an innermost untensioned spiral endpoint radius of 2 inches. To make the spiral in Web 2 similar in size to the spiral in Web 1 when tensioned, the spiral in Web 2 was designed to have an untensioned outermost spiral endpoint radius of 18.4 inches and an untensioned innermost spiral endpoint radius of 1.9 inches.

### 3.2 Testing Setup

Data was collected using a large-scale physical spiderweb frame and data acquisition system that had been used for previous experiments. The data collection setup consisted of an octagonal frame with attachment points for each radial. Each attachment point had a load cell attached to it to control for the radial tension and ensure
Figure 2: A web set up in the web frame used for testing.

that the tension was evenly spread throughout the radials. When used with an eight radial web, opposite attachment points were 4 feet apart with an attachment point in the middle of each side of the octagon. The web and the frame is shown in Figure 2.

The web was set up for testing by putting each radial in a tuning machine at each one of the corresponding attachment points with the outermost spiral ending at the top of the web. Each tuning machine was made up of a worm gear and pinion. Right before the tuning machine, the radials were snaked through three free spinning guide wheels, one of which was attached to the load cell as is shown in Figure 3a. First, the excess slack was taken out of each of the radials. Then, the load cells were turned
The web tension was increased incrementally, bringing each radial to up to an intermediate tension range before increasing the tension on each radial to the next intermediate tension range. This was done over a series of intermediate tensioning steps because the tension in each radial is affected by the tension of the other radials. So, the web radial tension was increased in increments until each of the load cells was at or above the desired load. Since, when first tensioned, the wraps of cord in the tuning machine would gradually tighten over time, decreasing the tension in the radials, the web needed to be tensioned above the desired tension point and then the tension needed to be decreased after the tension stabilized.

Every segment in every even spiral turn was marked with dots using a permanent marker to help with the ease and accuracy of the data collection by allowing the human plucking the web to easily see which spiral turn they were plucking. The outermost spiral turn had no dots, the next spiral turn inward had one dot in the
middle of each segment, the next spiral turn inward had no dots, the one after that had two dots in the middle of each segment, and so on with increasing numbers of dots until each spiral segment that was not next to or under a spider leg was either marked or next to a marked segment. The segments were marked while the web was under tension to prevent segments from twisting or overlapping during the marking process which could lead to the mislabeling of segments.

A 7 \( \frac{3}{8} \) inch diameter 3D printed “spider” was centered in the middle of the web to collect data as is shown in Figure 4. A stack of washers was bolted to the center of the 3D printed spider. The 3D printed spider had alligator clips at the end of each of its eight legs to clip on to the radials of the web. Also an accelerometer was attached to the end of each leg and oriented so that the z axis of the accelerometer was normal to the plane of the web. The wires for each accelerometer were fed through the keyring
in the center of the web and attached with slack to a horizontal rod behind the web center to prevent the wires from pulling on the accelerometers.

### 3.3 Data Collection

A MATLAB script was written to select 100 random numbers, each corresponding to spiral segment. The innermost two spiral turns on Web 1, the 45 degree angle build web, and the innermost three spiral turns on Web 2, the 0 degree build angle web, were excluded due to being either under the spider or next to the spider legs. The MATLAB script used is shown in Appendix B.5. These plucks were output in a format that listed the angle of each pluck, the corresponding section number for each angle, and the spiral turn number. Both the angle of the pluck and the section number of the pluck were output to prevent future confusion over the web orientation relative to the frame. The angle of the pluck was referenced from the frame and accelerometers with load cell 1 and accelerometer 1 being at the 0 degree position. The section number for the pluck angle was referenced from the web geometry with section 1 containing the longest spiral segment and section 2 containing the second longest spiral segment. Since the web was oriented with the outer endpoint of the spiral being at the highest point, the 22.5° pluck angle corresponded to section 2 and the 67.5° pluck angle corresponded to section 1.

The web data acquisition system was turned on and connected to a computer. The Labview program written by Andrew Otto to collect the pluck data was opened.
Each pluck location was plucked one by one, waiting for the Labview program to finish recording each pluck before the next pluck was started. Video was recorded of the whole web plucking process for each web to allow for plucks locations to be verified later if necessary. To minimize differences between plucks, all plucks for a single web were recorded in a single session, all plucks were done by the same person, and plucks were attempted to be normal to the plane of the web.

A total of four different experiment setups were run. The first experiment used Web 1, the higher spiral tension web, at a radial tension of 120 Newtons. The second experiment used Web 1, the higher spiral tension web, at a radial tension of 100 Newtons. The third experiment used Web 2, the lower spiral tension web, at a radial tension of 120 Newtons. The fourth experiment used Web 2, the lower spiral tension web, at a radial tension of 100 Newtons.

After all plucks were collected for an experiment setup, the pluck data was run through a preliminary version of the localization algorithm to ensure that the data had been properly recorded. A preliminary version of the algorithm was used because, at the time of data collection, the MATLAB version of the algorithm had not been finished. After all the plucks were collected for all of the experiments, all of the data was run through the final localization algorithm and analyzed.
3.3.1 Localization Algorithm

Because these experiments were intended to test the existing localization algorithm on different parameters, the MATLAB implementation shown in Appendix B.7 was used. This implementation was made by Andrew Otto and adapted from the localization algorithm implementation used on the SpiderHarp. A major difference between this implementation and the implementation used on the SpiderHarp is that the SpiderHarp implementation processes the data from the accelerometers in real time. The localization algorithm implementation used in these experiments runs on data that has been previously collected. This choice was made so a single set of data could easily be analyzed over and over again with different algorithm parameters or, in future studies, with different algorithms.

The localization algorithm implementation used in these experiments starts by importing the actual pluck points and the accelerometer data. For each pluck, it cuts down the data to a specified time before and after the beginning of the pluck. The algorithm then attempts to pair the data from each accelerometer with each other accelerometer to measure how well the sets of data match. When an imaginary line is drawn between the pluck and the center of the web, the motion of the sensors should be near symmetric across that line. So, the algorithm defines pairs of legs that get similar signals when the pluck comes from one of two angles that are 180 degrees apart from each other. When the algorithm finds the pairs of accelerometers that match best with each other, it then takes the root mean squared of the data to
determine which side had stronger vibrations. This information is then combined to
determine which direction the pluck originated from.

To estimate the range of the pluck, the algorithm uses Welch’s power spectral
density estimate and uses the first peak after the peak associated with the web’s
natural frequency to determine the relative length of the segment that was plucked.
This works because that peak’s frequency corresponds to the natural frequency of
the plucked segment. The natural frequency of the plucked segment can then be
corresponded to the relative length of the segment plucked. Because segments that are
farther away from the web center are longer, the relative segment length corresponds
to the relative range of the pluck. The calculated range is normalized to a value
between 1 and 0 with 1 being the spiral turn used in the experiments that was the
closest to the web center and 0 being the spiral turn used in the experiments that
was the farthest from the web center. In this implementation, the pluck points are
also taken in by the pluck localization algorithm so the actual pluck range can be
normalized to match the units of the predicted range. Then, the actual pluck angle
and actual normalized range can be output alongside the algorithm-predicted pluck
angle and range for ease of accuracy analysis.

4 Results

The MATLAB code in Appendix B.8 was used within the MATLAB code in
Appendix B.6 to compile and analyze the results of the localization algorithm. Testing
different algorithm parameter values was easier than testing different web tension parameter values because testing different web tension parameter values required recording more sets of plucks and, potentially, building more webs, both of which can be time intensive endeavors. Because of that, the results of varying the algorithm parameters were examined first in order to determine which algorithm parameter values to use when looking at the results of the different web tension parameters.

### 4.1 Results of Algorithm Parameter Variation

The two algorithm parameters varied affected how much of the pluck data was being used by the algorithm. While the full pluck data included one second before the pluck and four seconds after the pluck as is shown in Figure 5a, the algorithm only considers the data between a given time, $t_1$, before the pluck and a given time, $t_2$, after the pluck as is shown in Figure 5b.

To visualize the overall effect of varying $t_1$ and $t_2$, a list of values from 0 to 0.2 with a step size of 0.005 were tested as values for $t_1$ while a list of values from 0.01 to 0.2 with a step size of 0.005 were tested as values for $t_2$. Each combination of $t_1$ and $t_2$ values were tested with each combination of radial and spiral web tensions. Then, the results of the localization algorithm were compared with the actual pluck locations and the accuracy was calculated for each set of web tension conditions. For the pluck angle, accuracy was measured in percent of the pluck angles that were correctly predicted by the localization algorithm. The angle accuracy averaged for
Figure 5: Plots of the raw accelerometer data over time for a single pluck with (b) and without (a) t1 and t2 shown.
the four pluck data sets was plotted for each combination of t1 and t2 values, as shown in Figure 6. For the range of the pluck, the distances between the actual and predicted pluck locations were calculated. The range accuracy averaged for the four pluck data sets was plotted for each combination of t1 and t2 values, as shown in Figure 8.

To further show the effect of the t1 and t2 values on the accuracy of the localization algorithm, the angle accuracy for each t1 value while holding t2 constant at 0.015 was plotted in Figure 7a. This same process was repeated for the range accuracy, as is shown in Figure 9a. The process was repeated for t2, holding t1 constant at 0.1 while varying t2 and plotting the resulting angle accuracy in Figure 7b and the resulting range accuracy in Figure 9b.

Because it is desirable to have a high percentage of the pluck angles predicted correctly, the maximums of the plots in Figure 7 show the desirable outcomes. Therefore, for t1, values of above 0.025 make the localization algorithm better at determining the correct angle and, for t2, a value of at or near 0.015 makes the localization algorithm most accurate. For the pluck range, accuracy was measured in distance from the actual pluck range, so the minimums of the plots in Figure 9 show the desirable outcomes. As seen in Figure 9a, the minimum point before the error starts increasing again is when t1 is 0.045, so since t1 being 0.045 also yields a highly accurate angle prediction accuracy, 0.045 is the desired value for t1. As seen in Figure 9b, a value of 0.015 for t2 leads to the minimum difference between actual and predicted range.
(a) 3D plot of average angle accuracy as t1 and t2 are varied

(b) Top view of 3D plot of average angle accuracy as t1 and t2 are varied

Figure 6: Plots showing how angle accuracy changes when t1 and t2 are varied.
(a) Varying $t_1$ when $t_2$ is held at 0.015.

(b) Varying $t_2$ when $t_1$ is held at 0.1.

Figure 7: Plots showing the average accuracy of the predicted angle output by the localization algorithm as the data analysis cutoff before (a) and after (b) is varied.
Figure 8: Plots showing how range accuracy changes as $t_1$ and $t_2$ are varied.

(a) 3D plot of average range accuracy as $t_1$ and $t_2$ are varied.

(b) Top view of 3D plot of average range accuracy as $t_1$ and $t_2$ are varied.
(a) Varying t1 when t2 is held at 0.015.

(b) Varying t2 when t1 is held at 0.1.

Figure 9: Plots showing the average difference between the predicted range output by the localization algorithm and the actual range of the pluck as the data analysis cutoff before (a) and after (b) is varied.
Overall, a t1 value of 0.045 and a t2 value of 0.015 yields the most accurate results for the localization algorithm with both predicted angle and predicted range. Therefore, these values were used to observe results by each combination of web tension parameters. To see where the algorithm failed better, results for each combination of web parameters were also viewed when t2 was 0.05. The value for t1 was not varied when looking at separate results for each combination of web tension parameters because, as seen in Figure 7a and Figure 9a. This is likely due to the time before the pluck having little influence because most of the data in the time before the pluck is near-zero noisy values as is seen in Figure 5a.

4.2 Results of Web Tension Variation

As is shown in Figure 10, all of the webs tensions had localization angle accuracy rates of at or near 100% when t1 was 0.045 and t2 was 0.015. The two plucks that did not have their angle correctly identified in Figure 10c, were both at the same segment, which was the segment in the outermost spiral turn and in the 0 to 45 degree sector. Also, both of these plucks were misidentified as coming from the 180 to 225 degree sector, meaning they were 180 degrees off of the actual angle. In Figure 11, the percent of pluck angles correct by actual pluck angle are displayed again, this time when t2 was 0.05 so the accuracy is lower. In Figure 11 the incorrect pluck angles seem to be slightly more concentrated to one sector in Web 2 at both a radial tension of 100N and a radial tension of 120N. However, looking at these same localization
Figure 10: Accuracy of localization algorithm angle by angular section when $t_1$ is 0.045 and $t_2$ is 0.015.

algorithm results for angle accuracy but by range of the pluck in Figure 12, it becomes clear that all of the incorrect angle plucks were from the outer half of the spiral turns and, in the case of the Web 1 tests, the outermost quarter of the spiral turns. In both Figure 11 and Figure 12, the radial tension seems to have minimal or inconclusive affects on the pluck angle localization accuracy. However, the increased spiral tension in Web 1 seems to lead to incorrect results being concentrated closer to the outer edge of the web, as is seen in Figure 12.
Figure 11: Accuracy of localization algorithm angle by angular section when $t_1$ is 0.045 and $t_2$ is 0.05.
Figure 12: Accuracy of localization algorithm angle by distance from edge of web when t1 is 0.045 and t2 is 0.05.
Average Difference Between Predicted Pluck Range and Actual Pluck Range:
Web 1 at 100N Radial Tension with t1=0.045 and t2=0.015

Average Difference Between Predicted Pluck Range and Actual Pluck Range:
Web 2 at 100N Radial Tension with t1=0.045 and t2=0.015

Average Difference Between Predicted Pluck Range and Actual Pluck Range:
Web 1 at 120N Radial Tension with t1=0.045 and t2=0.015

Average Difference Between Predicted Pluck Range and Actual Pluck Range:
Web 2 at 120N Radial Tension with t1=0.045 and t2=0.015

Figure 13: Accuracy of localization algorithm range by distance from edge of web when t1 is 0.045 and t2 is 0.015.
Figure 14: Accuracy of localization algorithm range by distance from edge of web when $t_1$ is 0.045 and $t_2$ is 0.05.
Figure 15: Accuracy of localization algorithm range by pluck angle when $t_1$ is 0.045 and $t_2$ is 0.015.
Figure 16: Accuracy of localization algorithm range by pluck angle when t1 is 0.045 and t2 is 0.05.
The accuracy of the localization algorithm in terms of pluck range was noticeably different for Web 1 and Web 2. Figure 13 shows the difference between the normalized actual pluck range and the localization algorithm output range being less than 0.2 for Web 2 at both 100N and 120N of radial tension when t2 is 0.015. Figure 14 makes this difference even more clear by showing the average distance between the actual and predicted ranges being much higher for Web 1 than for Web 2 when t2 is 0.05. Figure 15 and Figure 16 display the same localization algorithm results but in terms of pluck angle, and they also make the difference between Web 1 and Web 2 clear in terms of the localization algorithm range accuracy. Figure 15 and Figure 16 also show that there is little affect of the pluck angle on the accuracy of the localization algorithm’s predicted pluck range.

5 Discussion

Altering the cutoff time before the pluck, t1, seems to lead to little change in terms of angle accuracy. This is expected because the data before the pluck is mostly near-zero noisy values and should contain little information about the pluck. However, for the both the range and the angle, there seems to be a minimum t1 time that leads to optimal results. This may be because having a little time before the pluck is recognized by the algorithm may allow the algorithm to better see the full pluck in the case of the algorithm declaring the pluck to start a little bit later than the pluck actually started. This could be due to the algorithm relying on the data to surpass
a certain threshold before recognizing that a pluck has occurred. However, for the range accuracy, a slightly larger $t_1$ value likely lead to higher accuracy because the range estimation method has a higher resolution when there is more data.

In terms of angle accuracy, the cutoff time after the pluck, $t_2$, seems to have a large effect. For small values of $t_2$, the angle accuracy is very high but the angle accuracy rapidly decreases as $t_2$ is increased. While more data leading to higher rates of inaccuracy might seem counterintuitive, since the algorithm depends on relative strength of vibrations at each leg to determine the direction the vibration originated from, the increased time of data might decrease the differences between different legs in average strengths of vibration. One way to test this would be by weighting the earlier data to be of more importance when determining which one of two directions 180 degrees apart from each other the vibration originated from. Also, vibrations bouncing back might lead to decreased accuracy when determining the pluck angle. Similarly, a longer time period of pluck data used after the pluck seems to lead to increased error in the pluck range predicted by the localization algorithm.

Plots of angle and range accuracy by angle were expected to show similar results for all angles because the web is near radially symmetric and differences might indicate flaws. Figure 10, Figure 11, Figure 15, and Figure 16 all show similar accuracy for each angle.

Because there was near perfect angle accuracy for all webs when the $t_1$ and $t_2$ values were adjusted to give optimal results, the angle accuracy in terms of pluck
range was examined at a non-deal t2 value to allow for comparison between plucks that had correctly identified angles and plucks that had an incorrectly identified angle. As shown in Figure 12, the localization algorithm output incorrect angles at a higher rate when the pluck originated from the outer portion of the web. This might be because the vibrations felt at the center of the web are weaker when the pluck is farther from the center of the web and the vibrations have more time to spread along the spiral segments before reaching the accelerometers on the legs closest to the pluck.

Overall, both radial tension values and both webs were able to have plucks localized with high accuracy when the t1 and t2 values for the algorithm were adjusted to optimal values. However, even with optimal adjustable algorithm parameters, the web with lower spiral tension, Web 2, seemed to have slightly more accurate range localization results. This difference was only exaggerated when a non-ideal t2 value was used for the algorithm.

5.1 International Impact

This research fits into a global effort to better understand spiders. The findings of this research could be used by biologists to determine if spiders use similar mechanisms to find their prey.

Also, this research affects the SpiderHarp. Because the SpiderHarp is an interactive display of the localization algorithm, it allows people of various ages and
backgrounds to interact with the study of vibrations in ways that research papers and articles cannot. Furthermore, the SpiderHarp reaches across disciplines by incorporating music, technology, biology, and physics. Because music can reach across cultural and linguistic barriers, the SpiderHarp can easily have impact outside the English-speaking community.

6 Conclusion

The results of these experiments suggest that there is an optimal window of pluck data for the localization algorithm. For this version of the localization algorithm and the web tensions tested, the optimal window appears to start 0.045 seconds before the pluck and end 0.015 seconds after the pluck. Increased time after the pluck seems to decrease accuracy both in terms of angle and range. This could be utilized by the SpiderHarp to ensure the instrument’s response time is as fast as possible.

While there appears to be little difference between different web tensions in terms of angle accuracy, the experiments using the web with lower radial tension had more accurate range localization. Therefore, webs with a shallower build cone angle should be used over higher spiral tension webs for applications such as the SpiderHarp where accurate localization is critical.
6.1 Future Work

Future work could examine the effects of adjusting different web geometry parameters such as the number of radials or the number of spiral turns. Additionally, asymmetric web geometries could be tested. As Zschokke and Nakata conclude in “Spider orientation and hub position in orb webs,” the asymmetry seen in vertical orb webs allows the spider be closer to the top of their web, which is favorable because gravity hinders their speed when climbing up and assists their speed when traveling down [13]. Other ways to better approximate spider webs seen in nature include more radials, more spiral turns, and more complex web geometries such as a separate spiral density for the hub at the center of the web. Also, a spider that has lost a leg could be simulated by removing one of the accelerometers. Testing more tension values for both the spirals and the radials could also further explore if more extreme tension variations lead to more extreme variations in the accuracy of the localization algorithm.

Additionally, different localization methods, such as time delay of arrival, could be tested. Also, the findings of this thesis could be applied to the localization system used on the SpiderHarp and tested for accuracy.
References


Appendix: Web Building Guide

1. Prepare segment lengths

(a) Generate segment lengths using the web generator MATLAB code

(b) Convert lengths to centimeters and organize by radial for the radial lengths and by 350 centimeters increments for the spirals

2. Radial Preparation

(a) Cut an approximately 4 foot long length of parachute cord for each of the radials [Figure 17]

![Figure 17: Radial cord cut to 4 feet long](image)

(b) Melt ends of radials [Figure 18]

(c) Tie radials to 1 inch diameter key ring using a slip knot [Figure 19]

   i. Loose end of slip knot should be used for radial

   ii. Fixed end should stay at ring center
iii. Ring should go through loop

iv. Knot should tighten on ring when radial end is pulled

(d) Leave 0.25 to 0.5 inches of tail on the end of the knot near the key ring

[Figure 19i]

(e) Cover knot with superglue gel on intersecting lines face after tightening knots

(f) Use metallic permanent marker to mark each radial end with dots corresponding to the radial number

(g) Extend a metric measuring tape next to the cord [Figure 20]

(h) Use metallic permanent marker to mark each intersection point on each radial to the nearest 0.5 millimeters, measuring from the key ring [Figure 21]
Figure 19: Tying the slip knot step-by-step

Figure 20: Measuring tape extended next to radial
Figure 21: Intersection point marks on radial
3. Spiral preparation

(a) Get a spool of bungee cord and ensure there is enough cord to finish the web with a single, continuous length of bungee cord [Figure 22]

![Figure 22: Bungee cord spool](image)

(b) Tie off the loose end of the bungee cord with a figure eight knot, leaving 0.5 to 1.5 inches of tail on the end of the knot [Figure 23]

![Figure 23: Tie off end of bungee cord](image)
(c) Mark the starting point of the measuring with a dark ballpoint pen or fine permanent marker [Figure 24]

Figure 24: Mark first intersection point

(d) Use a clamp to fasten the end of the bungee cord to a secure object to prevent movement while measuring [Figure 25]

Figure 25: Clamp holding down end of bungee cord

(e) Clamp near a flat surface
(f) Roll out about 370 centimeters of bungee cord [Figure 26]

Figure 26: 370 centimeters of bungee cord rolled out

(g) Spin spool to tighten the bungee cord until completely straight, then slowly spin in the reverse direction until the bungee cord has almost no tension and is almost straight. There will likely be some bends that are smaller and shallower than if there is slack in the bungee cord.

(h) If using a spool that is heavy enough to hold itself in place, begin measuring. If not, use weight to hold end of bungee cord in place on the spool side.

(i) Extend a metric measuring tape from the clamp to the spool [Figure 27]

(j) Align the 3 centimeter mark with the start point [Figure 28]

(k) Use a permanent marker or ballpoint pen to mark each point along in the 350 centimeter region
Figure 27: Measuring tape extended from clamp to spool

Figure 28: Align first mark with 3 centimeters mark

(l) Use a line around the circumference of the cord to mark the last point in the 350 centimeter region and put a dot next to the line [Figure 29]

(m) Repeat steps 3(d) through 3(l) for the next 350 centimeter region with the last point of the previous 350 centimeter region being the first point of the new 350 centimeter region.
(n) When all the intersection points have been marked on the bungee cord, cut off the end from the spool, leaving at least 2.5 inches free on the end.

[Figure 30]

(o) Tie a figure eight knot at the end of the bungee cord leaving 1 inch of tail on the end of the knot. [Figure 31]
4. Web Assembly

(a) Starting from one end of the spiral, align the first mark on the spiral bungee cord with the corresponding mark on the corresponding radial and fasten a zip tie to keep the cords secured together [Figure 32]

(b) Align the next mark on the bungee cord with the corresponding mark on
the corresponding radial and fasten radial and spiral together with a zip tie

i. Make sure to keep the bungee cord on one side of the parachute cord

ii. Make sure to keep the zip tie going the same direction

iii. Make sure the zip tie is tight

(c) Repeat step 4(b) until all intersection points are secured with zip ties

[Figure 33]

Figure 33: Web during process of connecting zip ties

(d) Cut off the loose end of the zip ties [Figure 34]

(e) Squeeze superglue gel into the intersections of the parachute cord, the bungee cord, and the zip ties

(f) Wait for glue to dry completely

(g) Stretch out web on a frame [Figure 35]

(h) Check for mistakes or defects
Figure 34: Zip tie with end cut off

Figure 35: Web on frame

(i) Break glue bonds between the radials and spiral that are outside of the zip tie

(j) Attempt to put similar amounts of tension on each radial
B Appendix: MATLAB Code

B.1 Web Build Instruction Generation

Written by Kathleen Gladson

```matlab
clear
clc

%% Web 1
% R = 24*.0254; % outer radius of fixed anchor points
% Ro = 13*.0254; % outermost radius of the capture spiral (untensioned)
% Ri = 2*.0254; % innermost radius of the capture spiral (untensioned)
% nr = 8; % number of radials
% ns = 16; % number of spiral turns
% beta = 45; % cone angle, degrees
% stretch = 0.07; % stretch factor
geo1 = spiralWeb(R,Ro,Ri,nr,ns,beta,stretch);
geo = geo1;

%% Web 2
R = 24*.0254; % outer radius of fixed anchor points
Ro = 19.75*(1-0.07)*.0254; % outermost radius of the capture spiral (untensioned) NO Ring
Ri = 2*(1-0.07)*.0254; % innermost radius of the capture spiral (untensioned) NO Ring
Ro = (19.25*(1-0.07)+0.5)*.0254; % outermost radius of the capture spiral NO Ring
Ri = (1.5*(1-0.07)+0.5)*.0254; % innermost radius of the capture spiral NO Ring
nr = 8; % number of radials
ns = 16; % number of spiral turns
beta = 0; % cone angle, degrees
stretch = 0.07; % stretch factor
go2 = spiralWeb(R,Ro,Ri,nr,ns,beta,stretch);
go = geo2;

%% Web 3
% R = 24*.0254; % outer radius of fixed anchor points
% % Ro = 19.75*(1-0.07)*.0254; % outermost radius of the capture spiral
% % (untensioned) NO Ring
% % Ri = 2*(1-0.07)*.0254; % innermost radius of the capture spiral
```
% untensioned NO Ring
% Ro = (19.25*(1-0.07)+0.5)*.0254; % outermost radius of the capture spiral
% Ri = (1.5*(1-0.07)+0.5)*.0254; % innermost radius of the capture spiral
% nr = 16; % number of radials
% ns = 16; % number of spiral turns
% beta = 0; % cone angle, degrees
% stretch = 0.07; % stretch factor
% geo3 = spiralWeb(R,Ro,Ri,nr,ns,beta,stretch);
% geo = geo3;

number_of_radial_segments = nr * (ns + 1);
number_of_spiral_segments = nr * ns - 1;

allLengths = geo.slack_lengths;
spiralLengths = allLengths(1:number_of_spiral_segments);
spiralLengths_cm = spiralLengths*100;

spirallengths_cm_350Chunks = zeros(length(spiralLengths), 1);
for k = 1:length(spiralLengths)
    if k == 1
        spiralLengths_cm_350Chunks(k) = spiralLengths_cm(k);
    else
        spiralLengths_cm_350Chunks(k) = spiralLengths_cm_350Chunks(k-1);
        if spiralLengths_cm_350Chunks(k) > 350
            spiralLengths_cm_350Chunks(k) = spiralLengths_cm(k);
        end
    end
end

spirallengths_cm_350Chunks_offset3 = spiralLengths_cm_350Chunks + 3;

allradialLengths = allLengths(number_of_spiral_segments+1:end);
radialLengths = zeros(ns+1, nr);
c = 1;
for i = 1:ns+1
    for j = 1:nr
        radialLengths(i,j)=allradialLengths(c);
c=c+1;
    end
end

radialDistFromCent = zeros(ns+1, nr);
for i = 1:ns+1
    for j = 1:nr
        radialDistFromCent(i,j)=sum(radialLengths(i:end, j));
        c=c+1;
    end
end
radialDistFromCent_cm = radialDistFromCent.*100;
radialDistFromEdgeRing_cm = radialDistFromCent_cm-(0.5*2.54);

**B.2 Web Geometry Generation**

Written by Andrew Otto

```matlab
function geo = spiralWeb(R,Ro,Ri,nr,ns,beta,stretch)
    % geo = SPIRALWEB(R,ro,ri,nr,ns,beta,stretch)
    % Generates simple spiral web geometry.
    
    % INPUTS:
    % R     final radius of the tensioned web
    % ro    radius of the outermost point on the capture spiral
    %       (untensioned)
    % ri    radius of the innermost point on the capture spiral
    %       (untensioned)
    % nr    number of radial lines in the web
    % ns    number of spiral turns in the web
    % beta  angle of the untensioned web (drawn in a cone)
    % stretch the approximate amount of radial stretch that will occur to
tension the web to the final radius R, from 0-1, typically
    
    % OUTPUTS:
    % geo   a structure containing information describing the untensioned
    %       web geometry
    
    r_max = R*cosd(beta)/(1+stretch); % the outer portion of the radials th
    if Ro > r_max
        error('Ro falls outside specified web boundary, decrease Ro or decre
    end
    r_step = ((Ro-Ri)/(ns-1/nr))/nr; % radial step
```
theta_step = 2*pi/nr;  % angular step (radians)
H = R*tan(beta);  % height of cone

r_current = Ro;  % outer radius of spiral for start
theta_current = 0;
ind = 1;

% outer nodes, not a part of the spiral, these are the points that are
for i = 1:nr
    x = r_max*cos(theta_current);
    y = r_max*sin(theta_current);
    z = H/R*(R-r_max);
    theta_current = theta_current + theta_step;
    nodes(ind,:) = [x y z];
    ind = ind+1;
end

% generate spiral portion of the web
for i = 1:ns
    for j = 1:nr
        x = r_current*cos(theta_current);
        y = r_current*sin(theta_current);
        z = H/R*(R-r_current);
        theta_current = theta_current + theta_step;
        r_current = r_current - r_step;
        nodes(ind,:) = [x y z];
        ind = ind+1;
    end
end

nodes(end+1,:) = [0 0 H];  % center node

spiral_elements = [(nr+1:length(nodes)-2)’ (nr+2:length(nodes)-1)’];  % spiral_ids = 1:length(spiral_elements);  % spiral element numbers
radial_elements = [(1:ind-nr-1)’ (nr+1:ind-1)’];  % node IDs that make up radial_elements = [radial_elements; (ind-nr:ind-1)’ ind*ones(nr,1)];  % elements = [spiral_elements; radial_elements];  % concatenate spiral and radial_elements = spiral_ids(end)+1:length(elements);  % radial element numbers

% element vectors, for calculating untensioned lengths
V = nodes(elements(:,2),:)-nodes(elements(:,1),:);
L = sqrt(sum(V.*V,2));
geo.slack_lengths = L;
geo.elements = elements; % connectivity
geo.nodes = nodes; % x-y-z coordinates
geo.radial_IDs = radial_ids; % radial elements
geo.spiral_IDs = spiral_ids; % spiral elements
geo.ground_IDs = 1:nr; % the nodes that will be anchored to ground

% create the nodes that will be the anchor points
thetas = linspace(0,2*pi,nr+1);
geo.anchor_points = [R*cos(thetas(1:end-1)) R*sin(thetas(1:end-1)) zeros(1,nr+1)]
end

B.3 Web Build Plot Generation

Written by Andrew Otto

function plotGeo(geo)
% PLOTGE0(geo)
% Plots the untensioned web geometry geo in the current figure.
% INPUTS:
% geo structure containing web geometry information obtained from a web geometry generating function such as spiralWeb()

cf
set(gcf,'color','w')
axes('position',[0 0 1 1])

% spiral line elements
spiral.X = [geo.nodes(geo.elements(geo.spiral_IDs,1),1)... geo.nodes(geo.elements(geo.spiral_IDs,2),1)]';
spiral.Y = [geo.nodes(geo.elements(geo.spiral_IDs,1),2)... geo.nodes(geo.elements(geo.spiral_IDs,2),2)]';
spiral.Z = [geo.nodes(geo.elements(geo.spiral_IDs,1),3)... geo.nodes(geo.elements(geo.spiral_IDs,2),3)]';

% radial line elements
radial.X = [geo.nodes(geo.elements(geo.radial_IDs,1),1)....
geo.nodes(geo.elements(geo.radial_IDs,2),1)';
radial.Y = [geo.nodes(geo.elements(geo.radial_IDs,1),2)...
geo.nodes(geo.elements(geo.radial_IDs,2),2)]';
radial.Z = [geo.nodes(geo.elements(geo.radial_IDs,1),3)...
geo.nodes(geo.elements(geo.radial_IDs,2),3)]';

% draw the elements
colors = lines(2); % colors for the two types of elements
spir = line(spiral.X,spiral.Y,spiral.Z,'color',colors(1,:), 'linewidth',1)
rad = line(radial.X,radial.Y,radial.Z,'color',colors(2,:), 'linewidth',1)

% draw lines showing the tensioning displacement
anchor_lines.X = [geo.nodes(geo.ground_IDs,1) geo.anchor_points(:,1)]';
anchor_lines.Y = [geo.nodes(geo.ground_IDs,2) geo.anchor_points(:,2)]';
anchor_lines.Z = [geo.nodes(geo.ground_IDs,3) geo.anchor_points(:,3)]';
line(anchor_lines.X,anchor_lines.Y,anchor_lines.Z,'color',colors(2,:),
     'lineStyle',':','linewidth',1)
hold on
plot3(geo.nodes(:,1),geo.nodes(:,2),geo.nodes(:,3),'k.' )
plot3(geo.anchor_points(:,1),geo.anchor_points(:,2),...
     geo.anchor_points(:,3),'k.' )
hold off

legend([spir(1) rad(1) ],'spiral','radial','location','northeast')

% axis view
view(3)
axis equal
axis tight
axis off

% figure title
annotation('textbox',[0.45 0.9 .1 .1],'string','Slack Web Geometry','horizcenter', 'vcenter','textalignment', 'center')
end
B.4 Converting Segment Number to Radial and Spiral Number

Written by Kathleen Gladson

function loc = segment2Location(seg_list, nr, ns)

% This function returns the angle and distance from the center given a segment list, the number of radials, and the number of spirals. From % to right the output columns are angle number, spiral number, polar an % in degrees, and original segment number

loc = zeros(length(seg_list), 4);
for i = 1:length(seg_list)
    loc(i, 1) = mod(seg_list(i), nr);
    loc(i, 2) = floor(seg_list(i)./nr) + 1;
    if loc(i, 1) == 0
        loc(i, 2) = loc(i, 2) - 1;
        loc(i, 1) = 8;
    end
    loc(i, 3) = pi/2 - pi/nr - (loc(i,1)-1)*(2*pi/nr);
    if loc(i,3) < 0
        loc(i,3) = loc(i,3) + 2*pi;
    end
    loc(i, 3) = rad2deg(loc(i, 3));
    loc(i,4) = seg_list(i);
end
end

B.5 Selecting Pluck Points

Written by Kathleen Gladson

clear
clc

nr = 8;  % Number of radials
ns = 16;  % Number of spirals

nrs = nr * (ns + 1);  % number_of_radial_segments
nss = nr * ns - 1;  % number_of_spiral_segments
nuis = 3;  \% number of unavailable innermost spirals

nas = nss - (nuis * nr - 1);  \% Number of available spiral segments

nrp = 100;  \% number of random points to generate

list_of_pluck_points = randi([1,nas], nrp, 1);

pluck_locations = segment2Location(list_of_pluck_points, nr, ns);

dlmwrite('W2_t100_0514_act.txt', pluck_locations);

B.6  Pluck Localization and Analysis Main Script

Written by Andrew Otto, modified by Kathleen Gladson

clear all
clc

folders = {
  'W1_t120_0514_FixedNames\'
  'W1_t100_0514\'
  'W2_t120_0514\'
  'W2_t100_0514\'
};
datafiles = {
  'W1_t120_0514_'
  'W1_t100_0514_'
  'W2_t120_0514_'
  'W2_t100_0514_'
};
pluckfiles = {
  'Web1PluckPointsMay14th'
  'Web1PluckPointsMay14th_t100N'
  'Web2PluckPointsMay14th_t120N'
  'Web2PluckPointsMay14th_t100N'
};

data_info = {
Web 1 at 120N Radial Tension
Web 1 at 100N Radial Tension
Web 2 at 120N Radial Tension
Web 2 at 100N Radial Tension

fs = 1000; % sampling frequency of the accelerometers
n_plucks = 100; % number of plucks, depends on experiment
thresh = .1; % threshold to trigger pluck onset, should be a fixed value

% time parameters that can vary for tuning purposes
% t1 = [0, 0.01, 0.015, 0.02, .05, 0.1, 0.2]; % how much time before the pluck onset
% t2 = [0.01, .015, 0.02, 0.05, 0.1, 0.2]; % how much time after the pluck onset
% t1 = .045; % how much time before the pluck onset to use in calculations
% t2 = [.015, .05]; % how much time after the pluck onset to use in calculations
% t1 = 0:0.005:0.2; % how much time before the pluck onset to use in calculations
% t2 = 0.01:0.005:0.2; % how much time after the pluck onset to use in calculations

avg_t_error_matrix = zeros(length(t1),length(t2));
avg_range_error_matrix = zeros(length(t1),length(t2));

for j = 1:length(t1)
    for k = 1:length(t2)
        sum_t_error = 0;
        sum_range_error = 0;
        for i = 1:length(folders)
            datafileprefix = [folders{i} datafiles{i}];
            pluckfile = [folders{i} pluckfiles{i}];
            [t_actual,t_predicted,r_actual,r_predicted] = pluckError(datafileprefix,pluckfile);
            t_error = sum(t_actual==t_predicted); % total successful or unsuccessful
            r_results = [r_actual r_predicted];
            results_file_name = [datafiles{i} ' localization_results ']
            results_file = [t_actual,t_predicted,r_actual,r_predicted];

            data_title_1 = [data_info{i} ' with t1=' char(string(t1(j)))]
            data_title_2 = [char(string(t_error))] % of All Pluck Angle
            data_title = {data_title_1, data_title_2};
            [side_by_side, correct_by_theta, correct_by_radius, range_by_theta, range_by_radius] = pluckError(datafileprefix,pluckfile);
            sum_t_error = sum_t_error + t_error;
            sum_range_error = sum_range_error + avg_range_error;
        end
        avg_t_error_matrix(j,k) = sum_t_error/4;
    end
end
avg_range_error_matrix(j,k) = sum_range_error/4;
end

% % Plot Error
% figure
% plot(t1, avg_t_error_matrix(:,2), 'LineWidth',2)
% title('Angle Error When Varying t1 (Time Before Pluck)
% xlabel('Time before pluck (t1)')
% ylabel('Percent of Pluck Angles Correct Averaged for All Trials')
% ylim([80 100])
%
% figure
% plot(t2, avg_t_error_matrix(5,:), 'LineWidth',2)
% title('Angle Error When Varying t2 (Time After Pluck)
% xlabel('Time after pluck (t2)')
% ylabel('Percent of Pluck Angles Correct Averaged for All Trials')
% ylim([80 100])
%
% figure
% plot(t1, avg_range_error_matrix(:,2), 'LineWidth',2)
% title('Range Error When Varying t1 (Time Before Pluck)
% xlabel('Time before pluck (t1)')
% ylabel('Average Difference Between Actual and Predicted Pluck Ranges')
% ylim([0 1])
%
% figure
% plot(t2, avg_range_error_matrix(5,:), 'LineWidth',2)
% title('Range Error When Varying t2 (Time After Pluck)
% xlabel('Time after pluck (t2)')
% ylabel('Average Difference Between Actual and Predicted Pluck Ranges')
% ylim([0 1])

B.7 Pluck Localization Algorithm

Written by Andrew Otto

function [t_actual,t_predicted,r_actual,r_predicted,r_int] = pluckError
% load in known pluck information
pluck_information = dlmread([pluckfile '.txt']);
theta_actual = pluck_information(:,4);
range_actual = pluck_information(:,3);
range_actual = (range_actual-min(range_actual))/range(range_actual);
% range_actual_relative = nan(size(range_actual));
% [~,sort_ind] = sort(range_actual);
% range_actual_relative(sort_ind) = 1:length(range_actual_relative);
% range_actual_relative = (range_actual_relative-min(range_actual_relative));

% set up leg pairing indices
pairs(:,:,1) = [1 2; 3 8; 4 7; 5 6]; % leg pairs at first angular sector
pairs(:,:,2) = [2 3; 1 4; 5 8; 6 7]; % leg pairs for second angular sector
pairs(:,:,3) = [3 4; 2 5; 1 6; 6 8]; % leg pairs for third angular sector
pairs(:,:,4) = [4 5; 3 6; 2 7; 1 8]; % leg pairs for fourth angular sector
pair_idx = squeeze(sub2ind([8 8],pairs(:,:,1),pairs(:,:,2))); % linear

theta = 22.5:45:360; % angular spacing of sectors
leg_theta = 0:45:360;
leg_theta = leg_theta(1:end-1); % dont need the repeat 360 degree value

theta_predicted = zeros(n_plucks,1);
range_predicted = zeros(n_plucks,1);

for i = 1:n_plucks
    filename = [datafileprefix num2str(i) '.txt'];
    % load data for a pluck
    data = dlmread(filename);
    % pick data subset to process using time clip parameters
    id1 = find(rms(data,2)>thresh,1);
    data = data(id1-round(t1*fs):id1+round(t2*fs),:);

    % orientation estimation
    % compute correlation coefficients of time domain signals
    r = corrcoef(data); % measures how well each leg correlates with each other
    r = r.^2; % use r^2 values
    % search pairings for strongest one
    symmetry_strength = sum(r(pair_idx)); % sum up correlations for each
    [~,symmetry_idx] = max(symmetry_strength); % max pairing identifies
    sym_angle = theta(symmetry_idx);
    % identify loudest direction using 2 norm (or RMS)
    loudness = rms(data);
% find polygon centroid from loudness and leg angles
[x,y] = pol2cart(deg2rad(leg_theta),loudness);
pgon = polyshape(x,y); % make polygon
[X,Y] = centroid(pgon); % find centroid
[T,R] = cart2pol(X,Y); % get centroid angle
T = rad2deg(T); % convert back to degrees
T(T<0) = 360+T(T<0); % make angle span 0-360 rather than +/-180

% check which side the centroid points relative to the symmetry axis
angle_diff = abs(sym_angle-T);
if angle_diff < 90
    theta_predicted(i) = sym_angle;
else
    theta_predicted(i) = sym_angle + 180;
end

%%% range estimation
% compute power spectrums
[pxx,f] = pwelch(data,[],[],[],fs);
pxx = mean(pxx,2);
% find frequency of first peak in power spectrum to use as range in
[~,range_predicted(i)] = findpeaks(pxx,'npeaks',1);

range_predicted_relative = nan(size(range_predicted));
[~,sort_ind] = sort(range_predicted);
range_predicted_relative(sort_ind) = 1:length(range_predicted_relative);
range_predicted_relative = (range_predicted_relative-min(range_predicted_relative))
range_predicted_relative = (range_predicted_relative-min(range_predicted_relative));

% function outputs
t_actual = theta_actual;
t_predicted = theta_predicted;
r_actual = range_actual;
r_predicted = range_predicted_relative;
r_int = pluck_information(:,3);
B.8 Localization Analysis

Written by Kathleen Gladson

```
function [side_by_side, correct_by_theta, correct_by_radius, range_by_radius] = localization_analysis(t_actual, r_actual, t_predicted, r_predicted, theta)

theta = 22.5:45:360; % angular spacing of sectors

n_plucks = length(t_actual);
t_results = zeros(n_plucks, 1);
side_by_side = zeros(n_plucks, 9);
% t_actual, t_predicted, 180 deg off, 45 degree off, r_actual, r_predicted,
% difference between actual and predicted, abs differences

for i = 1:n_plucks
    side_by_side(i, 1) = t_actual(i);
    side_by_side(i, 2) = t_predicted(i);
    if t_predicted(i) == t_actual(i)
        t_results(i) = 1;
    else
        if abs(side_by_side(i, 1) - side_by_side(i, 2)) == 180
            side_by_side(i, 4) = 1;
        elseif abs(side_by_side(i, 1) - side_by_side(i, 2)) == 360/theta
            side_by_side(i, 5) = 1;
        end
    end
    side_by_side(i, 3) = t_results(i);
    side_by_side(i, 6) = r_actual(i);
    side_by_side(i, 7) = r_predicted(i);
    side_by_side(i, 8) = r_actual(i) - r_predicted(i);
    side_by_side(i, 9) = abs(r_actual(i) - r_predicted(i));
end

correct = sum(t_results);
incorrect = n_plucks - correct;

off_by_180 = sum(side_by_side(:, 4));
incorrect_off_by_180_percent = off_by_180 / incorrect * 100;

off_by_45 = sum(side_by_side(:, 5));
incorrect_off_by_45_percent = off_by_45 / incorrect * 100;
```

correct_by_theta = zeros(3, length(theta));
correct_by_theta(1, :) = theta;

for i2 = 1:length(t_predicted)
    if side_by_side(i2, 3) == 1
        for i3 = 1:length(theta)
            if correct_by_theta(1, i3) == side_by_side(i2, 1)
                correct_by_theta(2, i3) = correct_by_theta(2, i3) +
                correct_by_theta(3, i3) = correct_by_theta(3, i3) +
            end
        end
    else
        for i3 = 1:length(theta)
            if correct_by_theta(1, i3) == side_by_side(i2, 1)
                correct_by_theta(3, i3) = correct_by_theta(3, i3) +
            end
        end
    end
end

for i4 = 1:length(theta)
    correct_by_theta(4, i4) = correct_by_theta(2, i4) / correct_by_theta(2, i4)
end

% Plot Correct Theta by Theta
figure
polarhistogram('BinEdges', deg2rad(0:45:360), 'BinCounts', correct_by_theta,
title({'Percent of Pluck Angles Correct by Actual Pluck Angle:' triangles'
thetaticks(0:45:315)

% Distance Analysis
correct_by_radius = zeros(4,4);
correct_by_radius(1,:) = 0.25:0.25:1;

for i2 = 1:n_plucks
    if side_by_side(i2, 3) == 1
        if 0.25 > side_by_side(i2, 6) && side_by_side(i2, 6) >= 0
            correct_by_radius(2, 1) = correct_by_radius(2, 1) +
            correct_by_radius(3, 1) = correct_by_radius(3, 1) +
        elseif 0.5 > side_by_side(i2, 6) && side_by_side(i2, 6) >=
            correct_by_radius(2, 2) = correct_by_radius(2, 2) +
correct_by_radius(3, 2) = correct_by_radius(3, 2) +
    elseif 0.75 > side_by_side(i2, 6) && side_by_side(i2, 6) >=
        correct_by_radius(2, 3) = correct_by_radius(2, 3) +
        correct_by_radius(3, 3) = correct_by_radius(3, 3) +
    elseif 1 >= side_by_side(i2, 6) && side_by_side(i2, 6) >= 1
        correct_by_radius(2, 4) = correct_by_radius(2, 4) +
        correct_by_radius(3, 4) = correct_by_radius(3, 4) +
    end
else
    if 0.25 > side_by_side(i2, 6) && side_by_side(i2, 6) >= 0
        correct_by_radius(3, 1) = correct_by_radius(3, 1) +
    elseif 0.5 > side_by_side(i2, 6) && side_by_side(i2, 6) >=
        correct_by_radius(3, 2) = correct_by_radius(3, 2) +
    elseif 0.75 > side_by_side(i2, 6) && side_by_side(i2, 6) >=
        correct_by_radius(3, 3) = correct_by_radius(3, 3) +
    elseif 1 >= side_by_side(i2, 6) && side_by_side(i2, 6) >= 1
        correct_by_radius(3, 4) = correct_by_radius(3, 4) +
    end
end
end

for i4 = 1:4
    if correct_by_radius(3, i4) ~= 0
        correct_by_radius(4, i4) = correct_by_radius(2, i4) / corre
    end
end

% Plot Correct Theta by Radius
figure
histogram('BinEdges',0:0.25:1,'BinCounts',correct_by_radius(4, :)')
title({'Percent of Pluck Angles Correct by Actual Pluck Range: ' tri
xlabel('Normalized Distance From Outermost Spiral')
xticks(0:0.25:1)
ylabel('Percentage of Time Pluck Angel is Correct (%)'
%

% side_by_side

avg_range_error = mean(side_by_side(:,9))

range_by_radius = zeros(4,4);
range_by_radius(1,:) = 0.25:0.25:1;
for i5 = 1:n_plucks
    if 0.25 > side_by_side(i5, 6) && side_by_side(i5, 6) >= 0
        range_by_radius(2, 1) = range_by_radius(2, 1) + side_by
        range_by_radius(3, 1) = range_by_radius(3, 1) + 1;
    elseif 0.5 > side_by_side(i5, 6) && side_by_side(i5, 6) >= 0.25
        range_by_radius(2, 2) = range_by_radius(2, 2) + side_by
        range_by_radius(3, 2) = range_by_radius(3, 2) + 1;
    elseif 0.75 > side_by_side(i5, 6) && side_by_side(i5, 6) >= 0.5
        range_by_radius(2, 3) = range_by_radius(2, 3) + side_by
        range_by_radius(3, 3) = range_by_radius(3, 3) + 1;
    elseif 1 >= side_by_side(i5, 6) && side_by_side(i5, 6) >= 1
        range_by_radius(2, 4) = range_by_radius(2, 4) + side_by
        range_by_radius(3, 4) = range_by_radius(3, 4) + 1;
    end
end

for i6 = 1:4
    if range_by_radius(3, i6) ~= 0
        range_by_radius(4, i6) = range_by_radius(2, i6) / range_by
    end
end

% Plot Correct Range by Radius
figure
histogram('BinEdges',0:0.25:1,'BinCounts',range_by_radius(4,:))
title({'Average Difference Between Predicted Pluck Range and Actual'})
xlabel('Normalized Distance From Outermost Spiral')
xticks(0:0.25:1)
ylabel('Average Difference from Actual Normalized Distance')
ylim([0 1])

range_by_theta = zeros(4, length(theta));
range_by_theta(1, :) = theta;

for i2 = 1:length(t_predicted)
    for i3 = 1:length(theta)
        if range_by_theta(1, i3) == side_by_side(i2, 1)
            range_by_theta(2, i3) = range_by_theta(2, i3) + side_by
            range_by_theta(3, i3) = range_by_theta(3, i3) + 1;
        end
    end
end
for i4 = 1:length(theta)
    range_by_theta(4, i4) = range_by_theta(2, i4) / range_by_theta(1, i4)
end

% Plot Correct Range by Theta
figure
polarhistogram('BinEdges', deg2rad(0:45:360), 'BinCounts', range_by_theta)
title({'Average Difference Between Predicted and Actual Pluck Range'})
thetaticks(0:45:315)
rlim([0 1])

end