# Tone Nation:
# An Online Intonation Training Application for all Languages

Kaden Kilburg
Garrett Herring
April Tan

Team Number 6

April 25, 2021

Project Report for
HCI 575: Computational Perception

Iowa State University
Spring 2021

## ABSTRACT

Computer-Assisted Pronunciation Training (CAPT) has been established to be effective for intonation learning due to its real-time, audio-visual capabilities and its ability to provide individualized feedback. However, many existing CAPT applications are either too technical or do not adhere to sound pedagogical principles, making learning ineffective. The goal of this app was to provide users a simple way to visualize the output of their intonation and to use their visualizations against a model speaker's visualization as corrective feedback. To determine the accuracy of this app, we first tested it against a group of native speakers. The results revealed a weak pitch direction accuracy (r=0.28) but a strong pitch height accuracy (MSE = 0.07). Next, to determine the efficacy of this app, we tested it with a group of non-native speakers of English. The pre-test/post-test data revealed that the experiment group showed a slightly higher improvement than the control group. Overall, while this app shows some promise, the algorithm for the pitch direction detection needs to be improved before it can be used as a tool for intonation learning.

# 1 Introduction

Computer-Assisted Pronunciation Training (CAPT) has long been established to be effective for intonation learning due to its ability to provide individualized, instantaneous, and unlimited audiovisual feedback on linguistic form in a way that human instructors cannot. It is unsurprising, then, that a wealth of CAPT systems have been introduced to the market for language learners. For language learners and instructors who struggle with the traditional language learning classroom restrictions, CAPT not only offers the aforementioned advantages, but it also provides the opportunity for unlimited practice in a private, stress-free environment at a learner's own pace.

Nevertheless, CAPT is a relatively new field and comes with many limitations, namely, that it has yet to bridge the divide between pedagogical theories and technology. Many existing and commercially CAPT systems may initially impress both instructors and learners with their fancy systems, but eventually, fail to meet sound pedagogical or technical requirements [1]. On the same side of the coin, some CAPT systems, while very technically accurate, are too difficult or cumbersome for the average learner or instructor to use in any practical way. Both these issues may not necessarily be due to a lack of willingness to include pedagogical theories into the design; it may simply be due to a lack of discourse and multidisciplinary collaboration amongst developers, linguists, and the end-users (instructors and learners).

What, then, should developers of CAPT systems include in the design and creation of new applications? To facilitate a multidisciplinary approach, this study will borrow general guidelines from the field of second language acquisition (SLA) as a theoretical basis for a suggested framework. As suggested by [1], the three important factors to be considered in language learning are authenticity of input, opportunities for output, and accuracy of corrective feedback.

First, authenticity of input in SLA can be defined as a learner's access to substantial, varied, and accurate samples to the target language. [2] points out that the development of new phonetic categories has to be derived from available exemplars, hence the need for authentic input. Secondly, opportunities for output can be defined as a space for learners to produce speech in order to test their hypothesis on new phonetic sounds. Through production, learners are able to receive proprioceptive feedback on their own performance and make adjustments as necessary [3]. Finally, corrective feedback is the feedback that helps learners notice the discrepancies between their productions and the target language. According to [4], it is only through this awareness that can lead to the acquisition of a new sound.

Taken together, the goal of this app, therefore, is to address the pedagogical gaps identified in literature by addressing issues related to the authenticity of input, opportunities for output, and the accuracy of corrective feedback in order to create a user-friendly and pedagogically-grounded intonation training app for learners of all languages. The following section addresses previous and existing CAPT applications for intonation training, their strengths, limitations, and our strategies for improving these applications.

# 2 Related Work

## 2.1 Input and output

Current trends in intonation training using CAPT consists of (a) an audiovisual display of the model speaker's pitch contour, (b) an audiovisual display of the learner's pitch contour for comparison; and (c) a re-recording on the part of the learner in an attempt to practice output. Research supports that these strategies do contribute to improving pronunciation [5] [6]. Examples of CAPT systems used by these researchers include *WinPitchLTL* [7] developed by Pitch Instruments Inc., *VisiPitch* by Kay-Elemetrics [8] and *VICK* [9].

While many researchers have used these systems with success, there exist several issues. Primarily, many of these systems were not originally designed for pronunciation training. Rather, they were intended for specialized speech research purposes, repurposed as CAPT tools. The visual displays for these systems include oscillograms, wideband spectrograms, and waveforms. The technical learning curve for such a tool is impractical, with even trained students requiring the presence of a teacher or researcher to help interpret them (see Figure 1). Furthermore, the UI of these systems is too cumbersome to be used as practice tools.

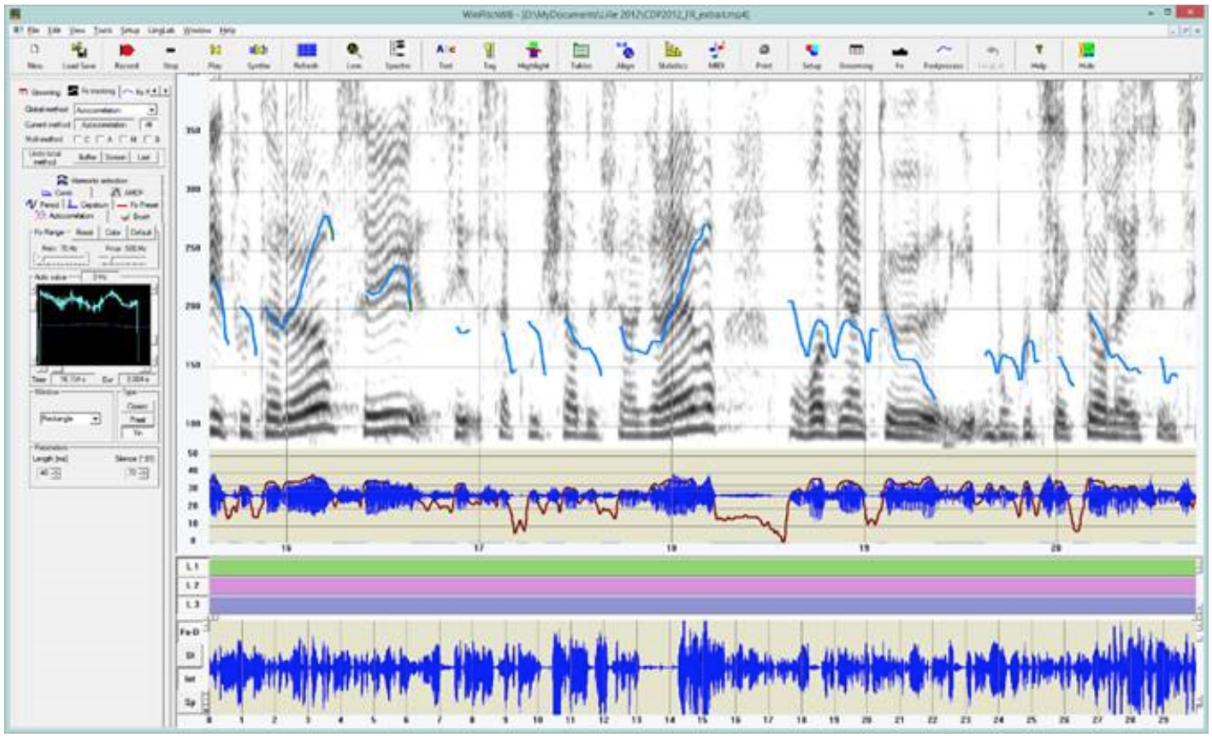

Figure 1: A screenshot of *WinPitchLTL*

## 2.2 Corrective Feedback

As mentioned previously, language learners require accurate corrective feedback to learn a new sound. Another issue with the CAPT systems mentioned above is that they utilize a direct compar-

ison method (placing two contours side-by-side), which wrongly presupposes that the only correct way to pronounce something should be the model speaker's way. Furthermore, direct comparisons do not provide a solution on how to correct these presumed errors, leading students to make random attempts at producing the right sound, which could actually lead to negative pedagogical implications.

In this regard, *WinPitchLTL* has been widely praised for its ability to provide a wide range of corrective features. For example, teachers can highlight different segments in different colors to call attention to a particular error. It also allows teachers to perform actions such as modifying the prosodic parameters of a student's production so that they can hear the correct prosody in his/her own voice. However, as everything is done manually, this is very time-consuming and defeats the purpose of a CAPT. It also requires a certain level of technical knowledge on the teacher's part to use these systems. Another CAPT system, *BetterAccentTutor*, automates the prosodic feedback of learners, offering both model speaker's and student's production on the screen, and pointing out relevant features and detailed explanations rather than providing prescriptive assessments. The limitations of this system, however, are that it uses preset exercises and recordings, limiting input. This makes it impossible to practice with user-sourced recordings or other languages. Additionally, the output of the learner's pitch contour is not immediate, creating a costly delay between production and the visual display.

***Study Intonation* App**

A previous attempt of a language learning application via intonation training was *Study Intonation*. This application structures training sessions into courses that consist of model audio files for a particular language and marked with spoken text. It offers a real-time display of user recorded audio mapped on top of the model speaker's audio. Due to *Study Intonation*'s [10] successful implementation, we take inspiration from its data pipeline and display.

# 3 Experimental Platform

To counter the issues of input mentioned above, we aim to allow users to be able to upload their own recordings, thereby increasing the variability and authenticity of input. Regarding output, our application aims to include only the necessary visual information for intonation learning (in this case, smoothed pitch contours), and to leave extraneous information out of the picture for ease of interpretability. This straightforward and minimalist display that plots pitch over time (see Figure 2) is familiar to most users and hopefully minimizes the cognitive overload for language learners. Additionally, the testing of output will be instantaneous. In other words, users will be able to watch the movements of their pitch contours in real-time, thereby increasing opportunities for the testing of output.

Regarding corrective feedback, our application aims to model the strategy used by *BetterAccentTutor* to call out areas of discrepancies rather than making prescriptive assessments of whether a learner's intonation is "right" or "wrong." We will do this by using important pitch cues (e.g. height, direction, movement) borrowed from the field of pronunciation, and to indicate when a learner significantly deviates away from the model speaker with familiar icons and colors.

Our team consists of one Applied Linguistics and Technology graduate, one Software Engineering

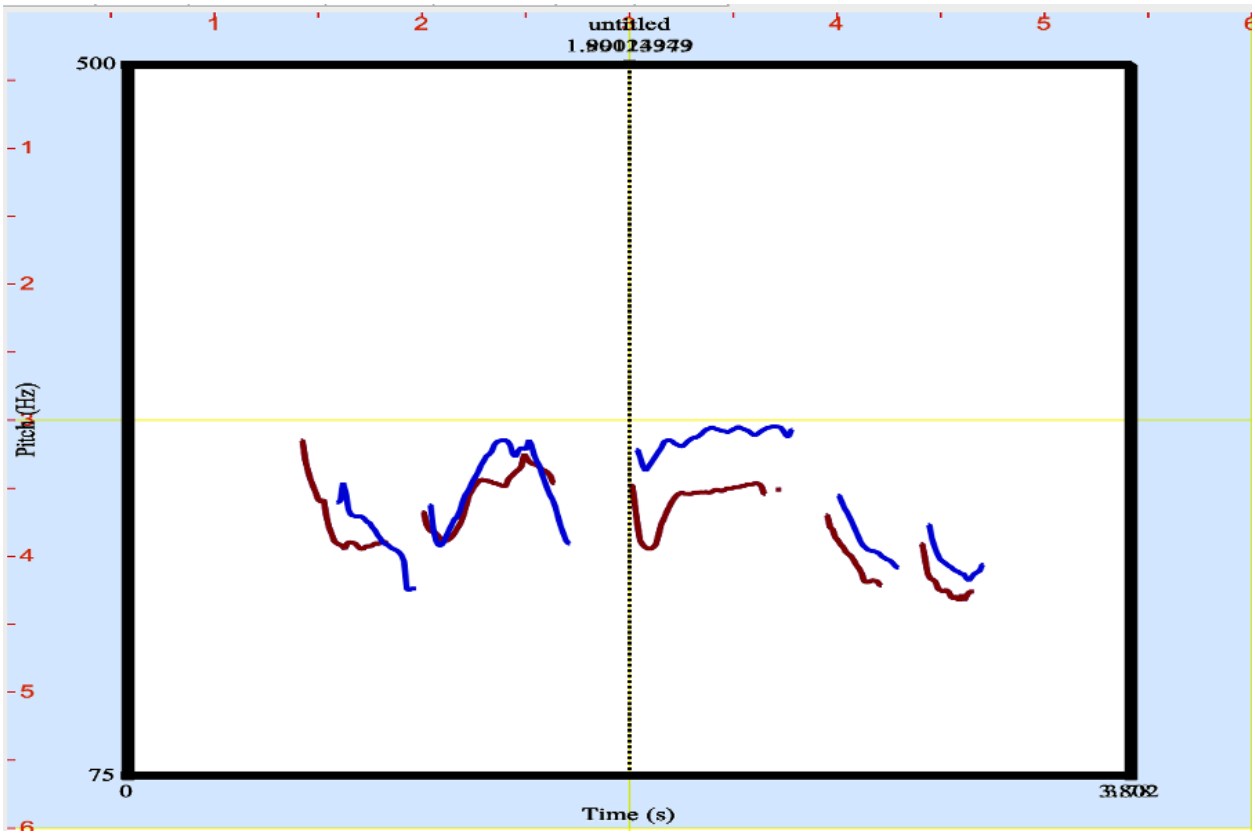


Figure 2: A proposed visual display

undergraduate with a minor in Mathematics, and another Software Engineering undergraduate with a minor in Psychology. Additionally, each member is interested and motivated in understanding human speech.

# 4 Methodology

The methods behind this project are rooted in *Study Intonation*'s work [10]. From the planning stage, we aimed to differentiate our project from its predecessor through the use of segmentation. Our proposed flowchart is shown below in Figure 3

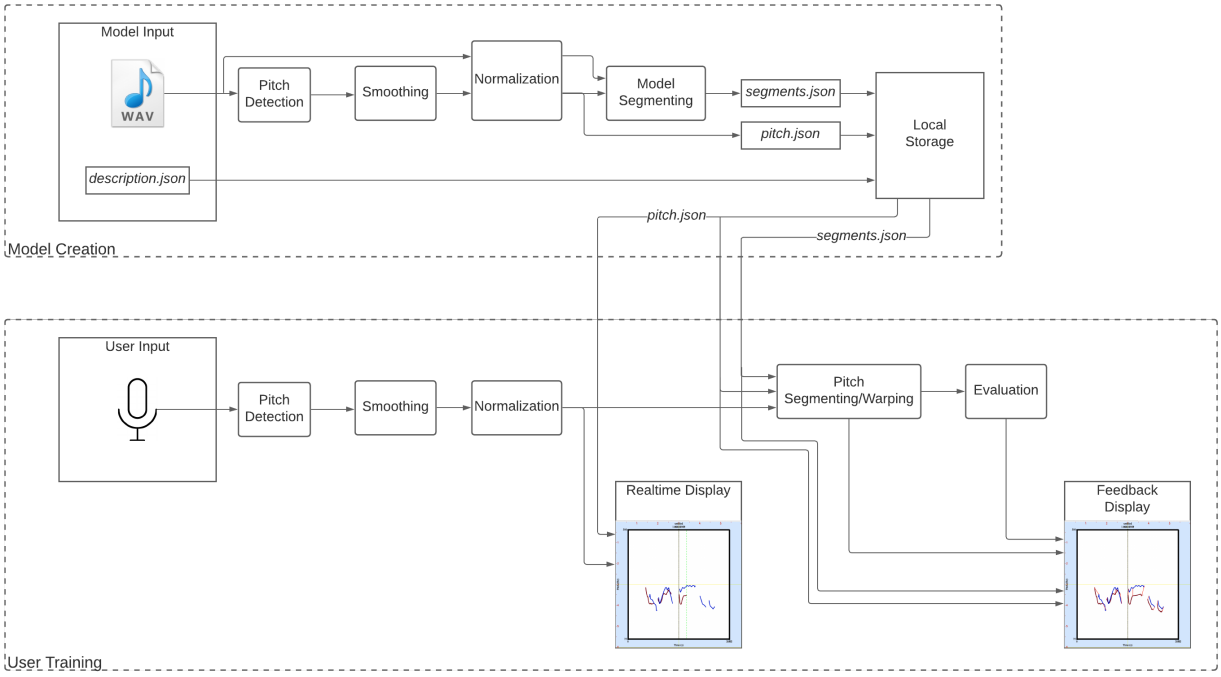


Figure 3: *Tone Nation*'s planned flowchart.

## 4.1 Interface

With the point of emphasis of our application being its usability, the interface should take the form of a web application. This allows for a high degree of accessibility as it can be operated with the only requirements being either a desktop or mobile web browser, internet access, and a microphone. The UI primarily consists of a page to either select or upload a model audio file, a page to listen to this audio and record the user's audio as they mimic the model, and a page for users to compare their recording with that of the model speaker's.

## 4.2 Audio

### 4.2.1 Model Speaker

Our intonation learning application revolves around the user attempting to mimic the sound of an example audio clip. This audio clip contains a short sentence spoken by a model speaker.

These sentences are taken from a spoken corpus titled, "100 Common English Phrases and Sentence Patterns" [11]. This corpus was chosen for several reasons. First, because it was free and

easily accessible. Second, all the sentences in the corpus were about 5 - 10 words long. This length was desirable as we wanted the most natural intonation possible, and shorter sentences prevented "read speech" when saying the sentences out loud. Lastly, the corpus contained a mixture of the four main types of intonation patterns: declarative, interrogative, imperative, and exclamatory. This allowed us to use a balanced mix in our experiments.

#### 4.2.2 User Audio

For the application to receive user audio, the JavaScript Web API is needed to receive microphone access. More specifically, recording audio from this microphone input stream is made possible by the API's MediaRecorder object.

### 4.3 Pitch Detection

After receiving audio input either from recorded user input or a model speaker, this raw audio must be processed for its pitch values at each timestamp. To do this, the YIN algorithm should be used to retrieve a time series of fundamental frequencies. This method uses autocorrelation in conjunction with filtering and estimation to reduce pitch detection errors. The YIN algorithm is chosen for its high accuracy at identifying fundamental frequencies within the range of human speech, 40 to 800 Hz, and its successful implementation in the *Study Intonation* [10] app. Its output provides an array of frequencies, $P$, where each frequency represents a subsequent pitch in time.

### 4.4 Smoothing

Smoothing is a necessary step for presenting the audio's pitch contour to the user. This smoothing also acts to discard noise and predict intermediate pitch values which helps in the segmentation and evaluation steps. For this, *Study Intonation*'s [10] implementation will be used again which leads us to use Schoenberg's cubic spline algorithm. The algorithm consists of the standard cubic spline's set of polynomials $S_i(t)$ where the zeroth, first, and second derivatives of adjacent polynomials evaluated at their intersections are equivalent: $S_i(t_{i+1}) = S_{i+1}(t_{i+1})$, $S_i'(t_{i+1}) = S_{i+1}'(t_{i+1})$, and $S_i''(t_{i+1}) = S_{i+1}''(t_{i+1})$.

In order to smooth the resulting line from connected $S_i(t)$, Schoenberg's algorithm asks to minimize the function

$$L = \rho \sum_{i=1}^{n} \omega (p_i - S_i(t_i))^2 + (1 - \rho) \int_{t_1}^{t_n} (S''(t))^2 dt \tag{1}$$

where $\omega_i$ is set to 1, $p_i$ is the detected pitch at time $i$, and $\rho$ is a roughness factor. To determine an adequate $\rho$, we start with defining a metric expressing how accurate a given spline function, dependent on $\rho$, is at predicting pitch values compared to the actual detected pitches. The metric will be the Pearson correlation coefficient

$$r(\rho) = \frac{\sum_{i=1}^{n} |(p_i - \overline{p})(S(t_i, p) - \overline{S(t, p)})|}{\sqrt{\sum_{i=1}^{n} (p_i - \overline{p})^2} \sqrt{\sum_{i=1}^{n} (S(t_i, p) - \overline{S(t, p)})^2}} \tag{2}$$

where $\overline{p}$ and $\overline{S(t,p)}$ are the mean value for detected and predicted pitches, $p_i$ and $S(t_i, \rho)$, respectively, for $i$ within the spline function's domain. Using this correlation function, we can then find a $\rho$ which brings the correlation closest to predetermined value $r_0$ which the *Study Intonation* study chose to be 0.95 "based on the opinion of phonologists." Minimizing the mean squared error

$$MSE = \sum_{j=1}^{m} (r_j(\rho) - r_0)^2 \rightarrow min \tag{3}$$

where $r_j(p)$ gives the correlation for the $j^{th}$ spline polynomial. This can be solved via the Brent algorithm which gives us the desired $\rho$ for Equation 1.

## 4.5 Adjusting Pitch and Time

When comparing the user's contour to the model speaker's contour, it is important to consider the natural differences in their voices to avoid giving inaccurate feedback. This is most easily seen between average speakers of the opposite sex: female speakers tend to have higher voices than males. Normalizing the pitch will help account for individual differences regardless of if the user's voice is naturally higher or lower. To do this, equation 4, which was also implemented in *Study Intonation* [10], will be used. The idea behind this equation is that any pitch $p$ in a given pitch vector $P$ will be compared to the minimum pitch in $P$. This value is then divided by the range of the vector which gives the percentage of the active pitch in the range of given pitches called $P'$. This is a standard interpolation technique to map values from a unique scale onto a common one.

Adjusting time can be done in a similar manner where it is also important to keep user input consistent with the model. This is done to account for the speed at which a speaker is speaking. Since the goal is to help improve the user's pitch contour, the ability to speak at any speed will be valuable. Normalizing time is done in the same way shown in equation 5.

$$P' = \frac{p - Min(P)}{Max(P) - Min(P)} \tag{4}$$

$$T' = \frac{t - Min(T)}{Max(T) - Min(T)} \tag{5}$$

However, there are some concerns with normalizing time. Since speaking in time is not the focus of the application, it would be desirable to allow the user to vary their speed to their comfort level. While normalizing time would allow the user to speak at a consistent differing speed, it would not allow them to vary their speed throughout. For this reason, time normalization between the user and modal audio will not occur on their completed recordings but in segments.

## 4.6 Segmentation

A naive approach to comparing the user's audio to the model's would be to compare pitch values between the two time series at every time $t$. However, this approach does not account for factors that immediately compromise its viability. The two most notable factors being a varying pronun-

ciation speed over a single word or multiple words and a varying pause length between words.

To compare user audio to the model, it is necessary to either map the user audio onto the correct locations of the model audio or map the model audio onto its mimicked locations of the user audio. For our approach, we take the latter method. To achieve this, we first aim to segment the model audio and then map each segment to its counterpart in the user audio.

For model segmenting, we will use a simple amplitude threshold method. This method is chosen for its frequent divisions of audio which still retain recognizable clips of audio. By segmenting based on audio amplitude, we intend to place divisions at grammatical pauses and plosives, also known as stop consonants. Segments will then be created whenever $a$ crosses $a_{thresh}$ where $a$ is the amplitude and $a_{thresh}$ is the threshold.

For user segmenting, dynamic time warping (DTW) will be used. The full process is complex, but it is described in [12]. To summarize the DTW, given two normalized pitch arrays $S$ of size $n$ and $Q$ of size $m$ the process will construct a $n$ x $m$ matrix $M$ which contains the distance, $d(i, j)$, from the pitch at index $i$ in $S$ to the pitch at index $j$ in $Q$. From this matrix, the shortest path from index $(1, 1)$ to $(n, m)$ is found while only allowing moves from ($i$, $j$) to either ($i + 1$, $j$), ($i$, $j + 1$), or ($i + 1$, $j + 1$). This idea is implemented with the following recurrence formula referenced in [12]:

$$
\begin{aligned}
1 < i \leq n, 1 < j \leq m \qquad (6)\\
M(i,j) = d(i,j) + min \begin{cases} M(i-1,j) \\ M(i,j-1) \\ M(i-1,j-1) \end{cases}\\
M(i,1) = d(i,1) + M(i-1,1)\\
M(1,j) = d(1,j) + M(1,j-1)\\
M(1,1) = d(1,1)
\end{aligned}
$$

An example of the matrix $M$ and its time warp mappings can be seen in Figure 4 and Figure 5, respectively. Using DTW, each model segment can be mapped to its replica in the user's normalized pitch array. This allows us to directly compare and contrast segments of audio between the user and model.

## 4.7 Comparing Time/Pitch Arrays

Once both the user's and model's audios are segmented, their matched segments will be compared, so that the user can be given constructive feedback on how their contour compared to the model's. To do this, the correlation and MSE of the two arrays will be evaluated. These equations were used in *Study Intonation* [10] and are a common procedure used in statistics to compare data sets.

$$
r = \frac{\sum_{i=1}^{k}(S_a(\tau_i) - \overline{S_a(\tau)})(S_b(\tau_i) - \overline{S_b(\tau)})}{\sqrt{\sum_{i=1}^{k}(S_a(\tau_i) - \overline{S_a(\tau)})^2 \sum_{i=1}^{k}(S_b(\tau_i) - \overline{S_b(\tau)})^2}} \qquad (7)
$$

In equation 7, $S_a$ and $S_b$ are pitch/time arrays. These two arrays are a pair of matched pitch segments. The input $\tau_i$ represents the $i^{th}$ time index of the array. This means that $S_a(\tau_i)$ represents

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **8** | 51.5 | 35 | 26 | 25.8 | 25.8 | 27.5 | 31 | 33.8 | 36.3 | 39.3 | 41.3 | 42.3 | 42.8 | **42.8** |
| **7** | 47.5 | 32.5 | 25 | 24.5 | 24.3 | 26 | 28.5 | 31.3 | 34.3 | 37.3 | 39.3 | 40.3 | **40.8** | 40.8 |
| **6** | 44.5 | 31 | 25 | 24.3 | 23.8 | 24.5 | 27 | 30.3 | 33.3 | 36.3 | 38.3 | **39.3** | 39.8 | 39.8 |
| **6** | 42.5 | 30.5 | 24 | 23.5 | 23.3 | 24 | 27 | 30.3 | 33.3 | 36.3 | **38.3** | 39.3 | 39.8 | 39.8 |
| **6** | 40.5 | 30 | 23 | 22.8 | 22.8 | **24** | **27** | **30.3** | **33.3** | **36.3** | 38.3 | 39.3 | 39.8 | 39.8 |
| **8** | 38.5 | 29.5 | 22 | 22.3 | **22.8** | 25 | 29 | 33.3 | 37.3 | 41.3 | 44.3 | 46.3 | 47.8 | 48.8 |
| **10** | 34.5 | 27 | 21 | **21.3** | 21.8 | 24 | 28 | 32.3 | 36.3 | 40.3 | 43.3 | 45.3 | 46.8 | 47.8 |
| **10.5** | 28.5 | 22.5 | **18** | 18.3 | 18.8 | 21 | 25 | 29.3 | 33.3 | 37.3 | 40.3 | 42.3 | 43.8 | 44.8 |
| **11** | 22 | 17.5 | **14.5** | 14.8 | 15.3 | 17.5 | 21.5 | 25.8 | 29.8 | 33.8 | 36.8 | 38.8 | 40.3 | 41.3 |
| **11** | 15 | 12 | **10.5** | 10.8 | 11.3 | 13.5 | 17.5 | 21.8 | 25.8 | 29.8 | 32.8 | 34.8 | 36.3 | 37.3 |
| **9** | 8 | **6.5** | 6.5 | 6.75 | 7.25 | 9.5 | 13.5 | 17.8 | 21.8 | 25.8 | 28.8 | 30.8 | 32.3 | 33.3 |
| **7** | **3** | 4.5 | 4.5 | 4.75 | 5.25 | 7.5 | 11.5 | 15.8 | 19.8 | 23.8 | 26.8 | 28.8 | 30.3 | 31.3 |
| | **4** | **5.5** | **7** | **6.75** | **6.5** | **4.75** | **3** | **2.7** | **3** | **3** | **4** | **5** | **5.5** | **6** |

Figure 4: A DTW matrix with example data from two time series aligned on the vertical and horizontal axes.

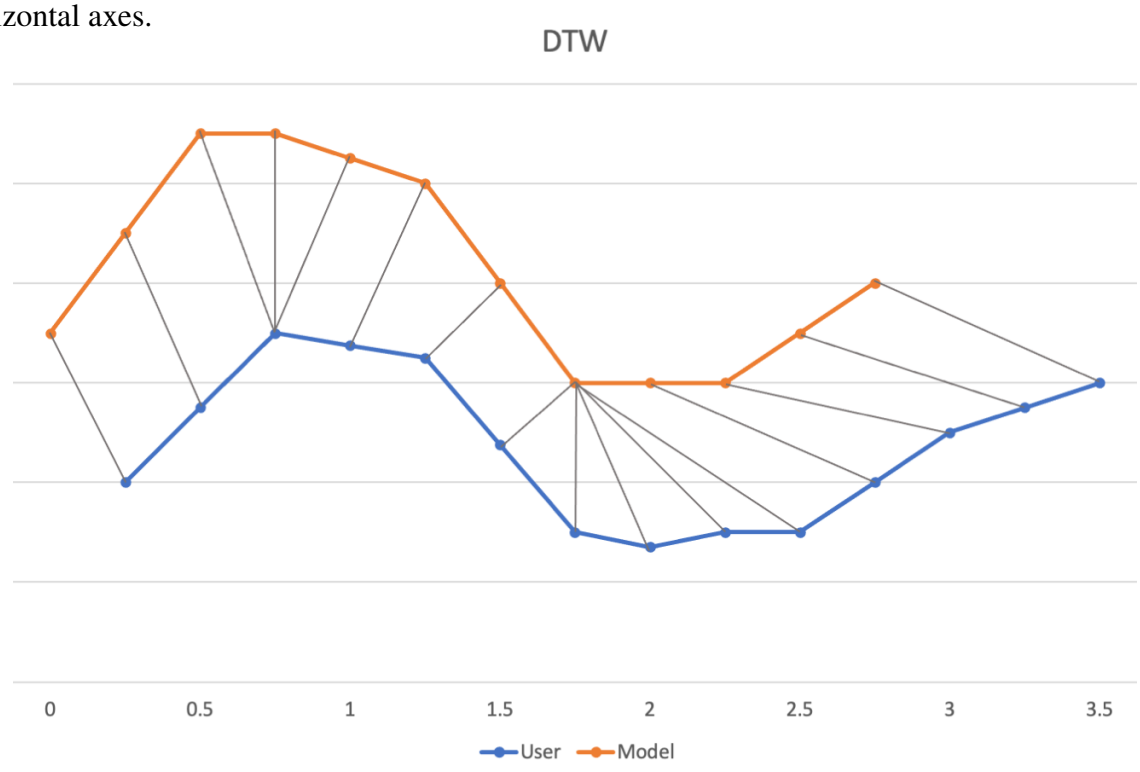


Figure 5: The line plot of data from Figure 4 showing the associations made by the DTW.

the pitch of array $S_a$ at time $\tau_i$. The output is the correlation coefficient. This can be thought of as a scale from -1 to 1 where the closer to 0 the coefficient is the weaker the correlation between the two arrays is and the greater the magnitude the stronger the correlation. If the value is negative, there is a negative correlation. This means the user's contour is moving in the opposite direction it should. Conversely, a positive correlation means that the contour is moving in the same direction.

Similarly, mean squared error (MSE) will also be used to compare the two pitch arrays. Equation 8 will be used for this and is also common in statistics. Since the arrays are broken into segments, taking the MSE of each will identify the most inaccurate segments. These segments will have higher MSE. Thus allowing more precise feedback to be given.

$$MSE = \frac{1}{k}\sum_{i=1}^{k}(S_a(\tau_i) - S_b(\tau_i))^2 \tag{8}$$

## 4.8 Evaluation and Feedback

Once the correlation coefficient and MSE have been calculated, feedback can be generated for the user. In essence, a high correlation coefficient means that the user's pitch is moving in the same direction as the model's, whereas a low correlation coefficient would mean the opposite. Similarly, a high MSE means that the pitch height for that segment is inaccurate relative to the other segments, whereas a low MSE would mean that the pitch's height is accurate relative to its other segments.

Taken together, a user's input can generate a combination of four different possibilities: a high correlation coefficient with a low MSE (correct direction, correct height), a low correlation coefficient with a low MSE (incorrect direction, correct height), a high correlation coefficient with a high MSE (correct direction, incorrect height), and a low correlation coefficient with a high MSE (incorrect direction, incorrect height). Using this model, corrective feedback will be generated for each segment, and areas of discrepancies regarding the pitch direction and pitch height relative to the model speaker's will be pointed out.

# 5 Implementation

While trying to stick to our desired methodology, some problems were encountered along the way. Here we will discuss what went well and what did not when attempting to implement the system. The final implementation's flowchart is visible in Figure 6 and the implementation can be found at `https://kmkilburg28.github.io/IntonationLearner/`.

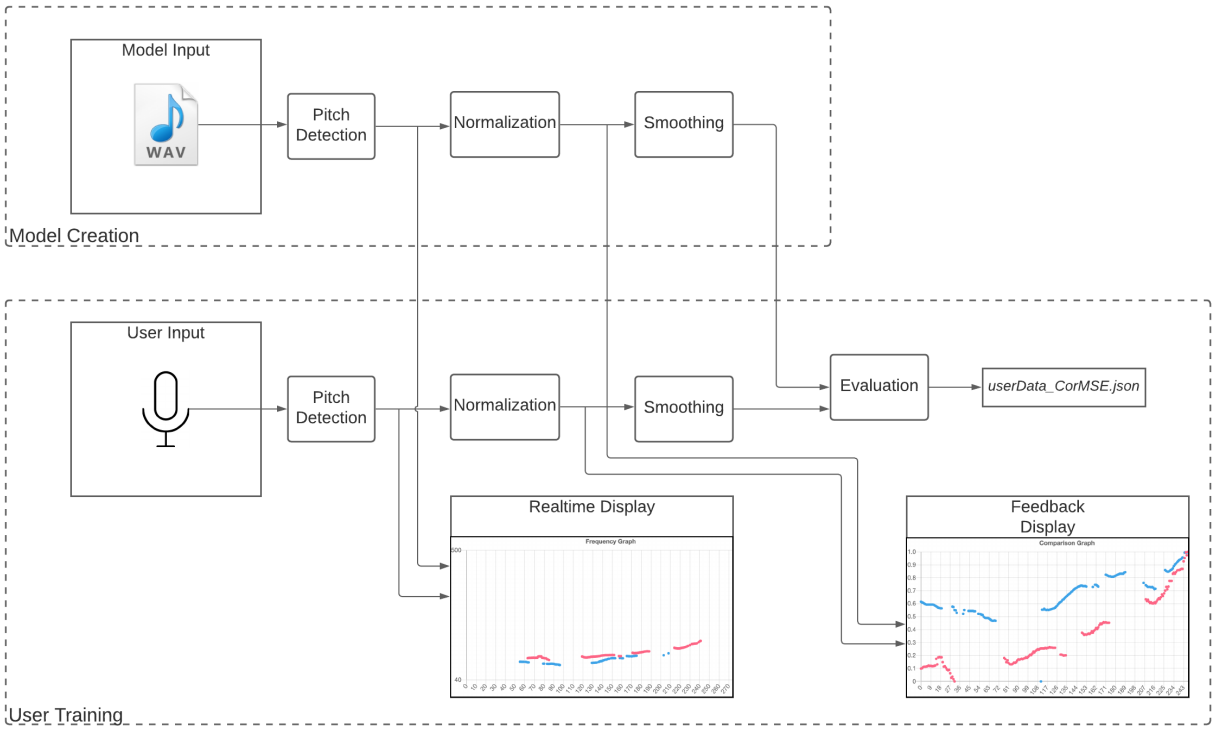


Figure 6: *Tone Nation*'s final flowchart.

## 5.1 Interface

*Tone Nation*'s interface was successfully implemented through web pages. The main goal of the interface was to possess three pages: a home screen for uploading and selecting model output, a training screen with real-time pitch detection, and an evaluation page for giving the user feedback. All of these were successfully implemented except for the home screen. Due to time constraints and limited client-side storage, user-uploaded model audio files were unable to be implemented. The real-time pitch interface can be seen in Figure 7

However, it should be noted that three additional web pages were added to organize and control our experiments; one page was made for each experiment group. These groups are discussed further in the Results section. The interface of experiment 2.1 is shown below in figure 8.

There are two more important notes to be made concerning *Tone Nation*'s interface. The first is about the application's device support. The intended goal was to support both desktop and mobile device usage; however, differences in the JavaScript Web API made this difficult and our team

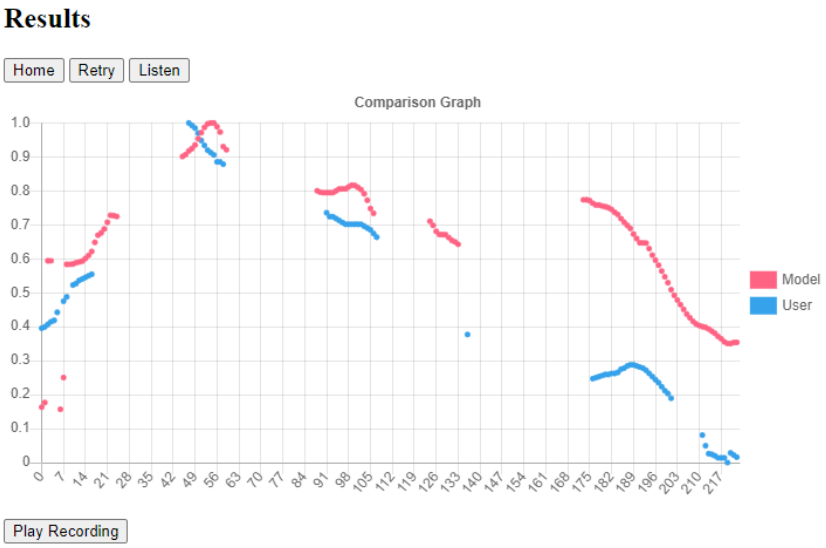


Figure 7: *Tone Nation*'s real-time pitch display. The user's pitch is shown in blue, and the model's pitch is shown in red.

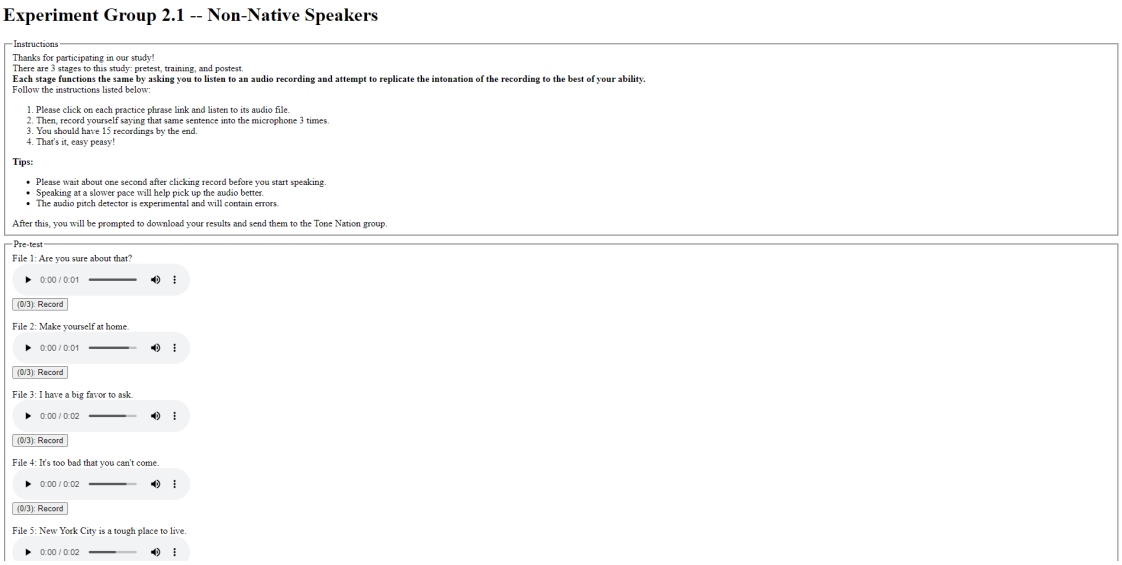


Figure 8: This is *Tone Nation*'s interface for experiment 2.1. It instructs the user how to perform the experiment and contains embedded audio controllers.

could only guarantee support for desktop usage. We believe that mobile support is still possible, but a device-native application would likely perform better. The second note is about our application's styling. By putting a larger focus on the computational aspects of our application, CSS styling was given little attention and could be improved dramatically by the addition of a designer

to the team.

## 5.2 Audio

No difficulty was found in using static model audio files sourced from the public corpus. However, recording user audio was problematic. This issue stems from the JavaScript Web API's audio functionality being underdeveloped for the use of real-time data collection. This is because the provided MediaRecorder object does a good job at recording audio input from start to finish. However, extracting data prior to stopping the recording creates issues.

When update requests are made for new audio data, an audio blob is given. This blob is formatted such that it can be appended to an existing file consisting of audio blobs which were previously requested. Therefore, the data of such blobs include metadata about the audio recording as well as being encoded itself. In order to use such data, a file reader is required on the concatenation of all previously requested audio blobs. However, as the file size grows, this concatenation and usage of the file reader becomes increasingly slow. This is a large bottleneck for the real-time display.

## 5.3 Pitch Detection

The YIN algorithm was successful in determining fundamental frequency. We used Adriano Di Giovanni's YIN implementation as our team had issues fully implementing the algorithm ourselves [13]. In addition, we ran into difficulties concerning the algorithm's usage.

The first of these difficulties was detecting lower range voices. Originally, we had used a window size of 1024 to detect frequencies within. This was used to help improve the speed at which pitches could be detected. The problem with the window size was that it was only possible to detect frequencies above 85.96 Hz. It is not uncommon for male voices to drop below this range, so it was necessary to increase the window size to 2048. This allowed frequencies all the way down to 43.02 Hz to be detected at the price of a significant reduction in speed.

The second difficulty was low frequency detection. This problem became noticeable once the raw input length was increased to 2048. Pitches were detected much less frequently because there were simply fewer frequency detection functions being called. This was because no overlap between frequency detection calls was being used. With this being the case, only a single frequency could be detected across this larger set of raw values and many frequencies beginning in one window's set and ending in the next would evade detection. Using an overlap of 1792 greatly helped pitch detection by searching for a new pitch every 256 data points. However, this further resulted in a large decrease in detection speed. While this slowed the results page load time, its effect on the real-time display was too great. To get around this, we shrank the overlap for real-time processing to 1536 along with making use of separate threads to detect pitches in parallel between windows.

The third difficulty was the noise produced by the YIN algorithm. While smoothing the frequency values will occur in a later step to reduce noise, this smoothing would still be skewed under heavy noise. A large factor in the amount of noise found by the YIN algorithm is its absolute threshold in step 4 of the algorithm [14]. By increasing this threshold, more noise is detected, but, by decreasing it, fewer pitches overall are detected. The ideal threshold depends on the audio quality, so we

settled for a value of 0.15. Furthermore, we implemented a couple of crude filtering techniques such as discarding pitches detected more than 100 Hz away from a previous pitch if the previous occurred less than 10 time steps ago. Additionally, we discarded all pitches below 50 Hz and above 500 Hz. These measures greatly increased the fit of the cubic spline in the following section.

## 5.4 Smoothing

For help with smoothing splines, we looked towards D.S.G. Pollock's paper on smoothing splines. There, he explains each part of Equation 1 [15] and how to minimize it. Additionally, he takes these explanations further by using them to derive pseudocode, albeit with a few bugs. Using this we successfully implemented cubic smoothing splines. However, due to a lack of time, we were unable to implement *Study Intonation*'s method of deriving a roughness factor, $\rho$. Instead, we simply choose the roughness factor to be 0.01 due to its visible appearance of fitting closely to commonly detected pitches and having a limited resistance to noise.

## 5.5 Adjusting Pitch and Time

The normalizing approach proposed in the methodology section was used and worked relatively well. With regards to normalizing pitch, the only time this gave inaccuracies was when the microphone was bumped while recording or picked up loud background noises out of the users speaking range. This would change the normalization drastically as normalizing is purely based on the minimum and maximum pitch.

To avoid this one could normalize using the average pitch or use the maximum and minimum values which occurred multiple times. This would help to normalize based on what was normally happening when recording instead of using the extremes which were recorded. This was not tried as normalizing in the way proposed did not cause issues when testing and if encountered could easily be remedied by rerecording.

With regards to normalizing time, inaccuracies can occur if there is a relatively long period of time between the user clicking record and when they start speaking or if there is a relatively long period of time between when the user is done speaking and clicking stop recording. While the fact that the pitch arrays are trimmed such that they start when the first pitch is detected and stop when the last pitch is detected helps mitigate this problem, if background noise is detected during times that would otherwise be trimmed this can skew how the normalized times line up. This can cause drastic changes in the results. But this also was of little concern as it can be fixed by rerecording or if necessary pressing the record and stop recording button more precisely.

Another concern we had with normalizing time was that it did not account for varying speaking speeds. We felt this was important because the focus of this application is on correct pitch not correct speed. Dynamic time warping was a proposed alternative to normalizing time and was tried initially. However, the next section discusses the issues with DTW and why it was not used in favor of normalized time.

## 5.6 Segmentation

When segmenting the pitch arrays, breaking the array into subarrays at points where there was a sufficient amount of silence worked well for segmenting the model. Consistent segments were

able to be generated for ten sample models. While this method worked well for the model it was not used for the user because the user's recording environment is unknown and could potentially cause issues when trying to map user segments to model segments. To resolve this concern, we originally proposed to use dynamic time warping. In practice, however, DTW had several issues. Most notably was the fact that when mapping, oftentimes get "stuck" and map several points in the opposite array to a single point. It would remain stuck until the difference in pitch can be reduced. This results in unwanted behavior in changes in direction or when there is a large difference between the heights of the two arrays. A visualization can be seen in figure 9. As seen in the figure, the model pitch array gets stuck mapping several values to just a few values in the user pitch array. Shortly after, the user then gets stuck mapping several of its values to just a few values in the model pitch array. In short, the pitch arrays are being warped in a way that is undesirable. As seen in figure 10, the warped arrays don't have much resemblance to each other or their corresponding original pitch array despite the original arrays being similar. In general, the warped pitch arrays were altered to the point where they could not be used to accurately evaluate the original pitch arrays.

One alternative approach that was tried was modifying the distance matrix to account for the distance in time, not just height, but this contradicts the goal of creating a system that can adapt to the users speaking pace and normalizing has similar results with less computation. Another alternative that was tried was using the difference in change in pitch instead of the difference in height in the distance matrix. This, however, requires the user to speak with relative accuracy which cannot be assumed. With the issues caused by DTW, we resorted to not segmenting the user's audio for comparison and instead compared the whole pitch array.

An alternative that was not tried would be to segment the user's pitch array using silence the same as the models and then match segments with similar relative length in chronological order. This was not tried because this does not account for varying speaking speeds and if the user has too much background noise segments would be unable to be identified.

With the issues encountered when implementing segmentation, we decided to stripe it from the final implementation. This is not ideal as it plays an important role when providing constructive feedback. But with the phrases used in testing being relatively short, this does not have as large of an effect if longer phrases were being used.

## 5.7 Comparing Arrays

When comparing pitch arrays the correlation coefficient reflected the direction of the array as intended. One concern with the final implementation is the fact that without segmenting, it is difficult to identify the direction of a whole phrase. This is because when speaking a phrase, pitch often goes up and down thus having no one distinct direction to the phrase. If we were able to break the phrase down into segments, it would be easier to identify the direction of said segments. Short bursts of sound when speaking often have direction while the whole phrase varies. Since we used the whole phrase this could potentially cause inaccuracies.

While the correlation coefficient seemingly worked as intended the mean squared error was not as easy to interpret. This is because after normalizing pitch and time, the MSE was consistently on the lower end of the spectrum. Normalizing the pitch and time made it difficult to identify the overall

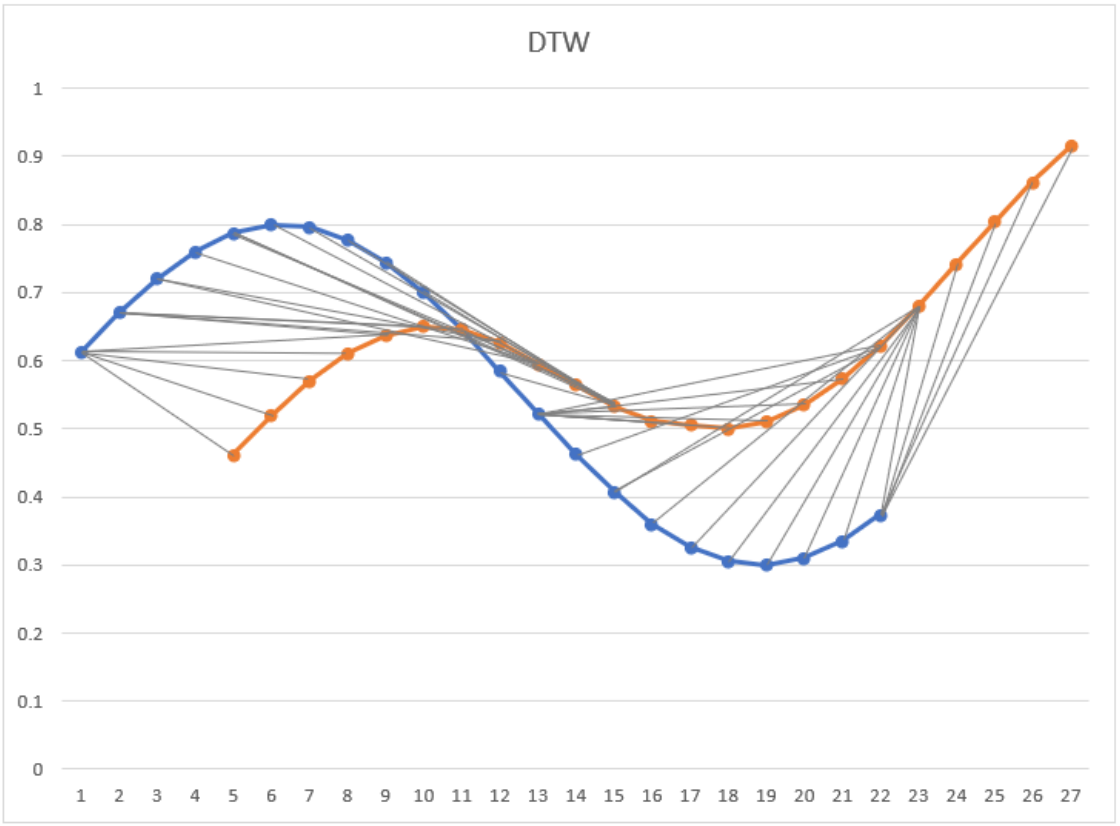


Figure 9: Line plot showing the associations made by the DTW on two modified sine waves.

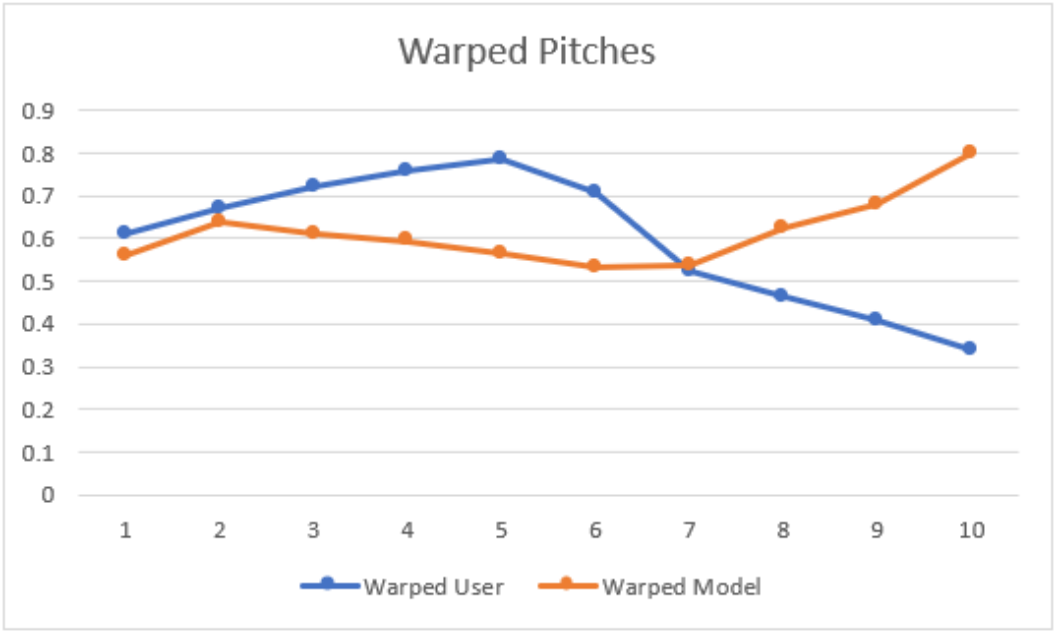


Figure 10: Warped pitch arrays from figure 9.

error in height. Since the pitches were adjusted to be on a scale from zero to one, squaring the difference made the MSE smaller. While the MSE is on a scale from zero to one, it is not a linear scale. If a model were to be designed to interpret the MSE, it would be important to keep this in mind. In hindsight, root mean squared error (RMSE) and mean absolute error (MAE) should have been considered. These options would have provided a linear metric to base our results on. The main difference in these is that RMSE yields a higher value when large errors are present and MAE is simply the average of the errors. It is difficult to say without testing how to interpret RMSE and MAE and perhaps future implementations should experiment and test with RMSE and MAE [16].

## 5.8 Evaluation and Feedback

With segmentation removed from the project, feedback consists of visuals, the correlation coefficient, and the MSE. While we had hoped to generate corrective feedback using the variables, we fell short of time and were not able to construct a model which we felt could give accurate corrective feedback based on the variables calculated. Ideally, such a model would have tiered thresholds for both the correlation coefficient and the MSE which would draw the user's attention to where errors were made and what needs to be done to correct those errors. Again, without segmentation, the feedback would be more general and unable to point out precise points which errors were made. Another reason why such a model was not calculated was the issues with MSE stated above. It is still possible but the thresholds for smaller values would require more precision than the correlation coefficient.

# 6 Results

This section reports *Tone Nation*'s viability as an intonation training app. Even though the eventual goal of this app is to use it with the intonation of any language, this study only tests it with the intonation of American English. Our study design consisted of Experiment 1 and Experiment 2. The goal of Experiment 1 was to understand the accuracy of *Tone Nation*'s algorithm by testing it against native speakers of American English; the goal of Experiment 2 was to test the efficacy of *Tone Nation* as an intonation learning app with learners of American English. The procedure and results of both experiments are reported below.

**Experiment 1**

*Procedure*

The app was tested with native speakers of American English (n=8) using model recordings of speakers who are also native speakers of American English. All of the participants were presented with 5 sentences and asked to say each sentence 3 times into the microphone. The correlation coefficient score and MSE score for each sentence was calculated for all 3 attempts for each participant. As both the model speaker and participants from this group are of the same language background, it is expected that their attempts should yield a consistently high correlation coefficient and low MSE score across all 5 sentences.

*Results*

Overall, the group average for Experiment 1 shows that while MSE scores were low as expected, the correlation coefficient mean was also significantly lower than expected (see Table 1). In other words, in terms of detecting the pitch height, the MSE score (M = 0.11) indicates that the algorithm was accurate in detecting the accuracy of a native speaker's pitch height whereas the algorithm for detecting the native speaker's pitch direction was weak (M = 0.28). Of note, the standard deviation for the correlation coefficient values in both the pre-test and post-test are high (SD = 0.37), which might be indicative of an inconsistency or over-sensitivity in pitch direction input. This could have contributed to the overall low scores for the pitch directions despite both groups of speakers coming from the same linguistic background. To corroborate this, individual results yielded a negative correlation coefficient score in some instances, suggesting that pitch direction might have even been moving in the opposite direction.

**Table 1.** Mean and SD correlation coefficient and MSE scores for Experiment 1

**Experiment 2**

*Procedure*

Experiment 2 tests the app with non-native speakers of English (n = 8), and consists of a pre-test, a training session, and a post-test. 4 of the participants were recruited to be in the experiment group (EXP) and the other 4 in the control group (CTL). During the pre-test, both groups were asked to listen to 5 model sentences once and to say that sentence into the microphone 3 times. The data was collected as a baseline reference for each participant. The 5 model sentences used were the same ones used in Experiment 1 to control for variability. Then, both groups went through an intonation training session. The CTL group was asked to practice the intonation of the 5 sentences, but via audio only. This audio-only condition serves as a control as it is similar to what language learners typically experience in the traditional classroom learning environment. The EXP group was asked to practice with *Tone Nation*, which afforded learners a real-time, audio-visual condition

Table 1: Correlation coefficient and MSE scores for Experiment 1
**Native speakers**

| Participants | CO Mean | CO SD | MSE Mean | MSE SD |
|---|---|---|---|---|
| NS1 | 0.26 | 0.25 | 0.10 | 0.05 |
| NS2 | 0.23 | 0.38 | 0.09 | 0.07 |
| NS3 | 0.49 | 0.29 | 0.07 | 0.06 |
| NS4 | 0.23 | 0.38 | 0.09 | 0.07 |
| NS5 | 0.44 | 0.37 | 0.10 | 0.06 |
| NS6 | 0.15 | 0.40 | 0.13 | 0.07 |
| NS7 | 0.13 | 0.42 | 0.18 | 0.10 |
| NS8 | 0.31 | 0.50 | 0.13 | 0.08 |
| Group Avg | 0.28 | 0.37 | 0.11 | 0.07 |

(see Figure 11).

After the training session, both groups went through a post-test, where they were asked to say the same 5 sentences they had been practicing with into the microphones 3 times. Finally, their mean pre-test and post-test coefficient correlation and MSE scores were calculated to check for improvements in their intonation. It is expected that both groups should show improvements, with the hope that the improvements from the EXP group will be higher. The metrics for measuring these improvements include a comparison of the pre-test and post-test results of both the CTL and EXP groups. Specifically, we expect the collected data to fit all of the following requirements:

- The average performance for the EXP group in the post-test should be stronger than in the pre-test.
- By the end of training, the average performance of the EXP group should have improved more than the average performance of the CTL group.
- For new models, average speaker performance is higher than the average performance recorded for the previous model. (Does this still apply?)

*Results*
Overall, the group average for both the CTL and EXP group in Experiment 2 showed minimal increases and decreases for both the coefficient correlation and the MSE (see Table 2), indicating that there were no significant performance improvements for either group.

For the CTL group, these findings are consistent with the literature in that audio-only conditions are slow to produce results. The data reveals that as a group, the participants started out with a low correlation coefficient mean (M = 0.30) which decreased slightly in the post-test (M = 0.25), indicating a slight regression in pitch direction performance. This group also started out with a low MSE score (M = 0.11) which increased slightly in the post-test (M = 0.13). This indicates

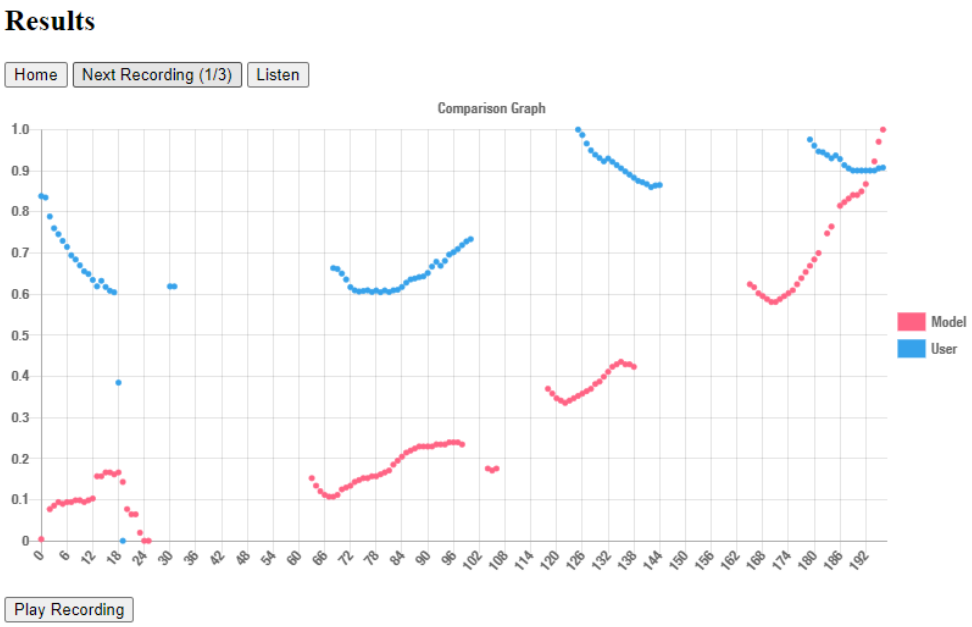


Figure 11: Intonation Training with Tone Nation.

Table 2: Pre-test and post-test coefficient correlation and MSE scores for Experiment 2
**Non-native speakers**

| | **CO Mean** | **CO SD** | **MSE Mean** | **MSE SD** |
|---|---|---|---|---|
| CTL Pre-test | 0.30 | 0.30 | 0.11 | 0.05 |
| CTL Post-test | 0.25 | 0.37 | 0.13 | 0.07 |
| EXP Pre-test | 0.50 | 0.33 | 0.09 | 0.06 |
| EXP Post-test | 0.52 | 0.30 | 0.09 | 0.06 |

a small regression in terms of pitch height performance. Of note, the standard deviation for the correlation coefficient values in both the pre-test and post-test are high (SD = 0.3, 0.37), which might be indicative of an inconsistency or over-sensitivity in pitch direction input. This could have contributed to the overall low scores for the correlation coefficient.

The EXP group showed a slight increase in the correlation coefficient mean from the pre-test (M = 0.50) to the post-test (M = 0.52), suggesting a slight increase in pitch direction performance. The group also started out with a higher MSE score (M = 0.33) in the pre-test than in the post-test (M = 0.30), which indicates a small improvement in terms of pitch height performance. Similar to the previous groups, the standard deviation for the correlation coefficient values in both the pre-test and post-test are high (SD = 0.33, 0.30), which might be indicative of an inconsistency or over-sensitivity in pitch direction input. This could have contributed to the overall low scores for the correlation coefficient.

# 7 Discussion

The goal of this experiment was to two-fold: (1) to gain feedback on the accuracy of the algorithm based on the native speaker group so that we can continue to improve the algorithm, and (2) to test the app on a group of language learners to test the efficacy of such an app. For the first objective, we were able to gain valuable feedback and identify the main problem area, namely, the correlation coefficient for the pitch direction. The inaccuracy could be a result of several factors. In an informal survey, several participants reported completing the test in a loud environment. Background noise and even other noises would have significantly impacted the data. Furthermore, several participants reported that because they were native speakers, they did not feel the need to mimic the model speakers' intonation exactly. As a result, they said the sentence as they normally would, which could differ quite a bit from the model speaker's intonation. All in all, these are important factors to take into account for the next iteration of the algorithm.

For the second objective, we were able to compare the pre-test and post-test scores between the CTL group and the EXP group. The data shows that the EXP group did result in a slight increase in performance in pitch direction and pitch height, whereas the CTL group showed a decrease in both pitch direction and pitch height. While the sample size is small and the data is too statistically insignificant to make any sort of meaningful conclusions, an informal survey with the EXP group did show that the participants enjoyed the visual component of the application and they reported that the pitch contours were helpful in helping them see what their voice was doing. The biggest limitation of this section was that we were expecting to see a more significant improvement within the EXP group. However, the same problems mentioned above with the native speaker group also applied to this group. Additionally, because of time constraints, we were not able to implement the corrective feedback portion of the app, which is an important component in language learning. Taken together, this could explain why our overall results are not as expected.

# 8 Conclusion and Future Work

Overall, there is much work to be done to the algorithm before it can be used as a tool in a learning context. First, the algorithm for this application needs to be improved upon, particularly in its pitch detection capabilities. The next iteration of the app will need to focus on background noise filtering and to include the possibility for the different types of intonation variations. Secondly, while visual feedback is a form of corrective feedback, a more robust corrective feedback model needs to be implemented within the app to help learners notice what they are doing well and what they need to work on.

Nevertheless, the app does show much promise. To our knowledge, it is the only intonation learning app right now that contains a real-time pitch detector. Immediate feedback has been shown to significantly aid a learner in understanding the mechanics of their voice, and more empirical research on how this could lend to pronunciation learning is much needed. Additionally, as mentioned above, the participants who used the application reported mainly positive feedback. Despite the occasional inaccuracies in the visual displays and some initial difficulties in navigating the application, they enjoyed playing around with it and testing out different pitch patterns.

As a whole, CAPT is a relatively new field and the development of an effective application requires

a multidisciplinary approach. This study combines linguistics and computational perception is a good first step in that direction. In future iterations of this application, it will also be crucial to improve upon the UX/UI of the design, not just for aesthetic purposes, but also for cognitive education purposes.

To our knowledge, there are no existing applications that aim to do exactly what this application does, which is to integrate the different fields of computational perception, linguistics, cognitive psychology, pedagogy, and visual design (amongst others). This intersection provides grounds for much research, particularly when it comes to the burgeoning field of human-computer interaction as technology continues to develop.

# References


[1] A. Neri, C. Cucchiarini, H. Strik, and L. Boves, "The pedagogy-technology interface in computer assisted pronunciation training," *Computer assisted language learning*, vol. 15, no. 5, pp. 441–467, 2002.

[2] A. James and J. Leather, *Second-language speech: structure and process*, vol. 13. Walter de Gruyter, 2011.

[3] K. De Bot, "Visual feedback of intonation i: Effectiveness and induced practice behavior," *Language and Speech*, vol. 26, no. 4, pp. 331–350, 1983.

[4] R. W. Schmidt, "The role of consciousness in second language learning1," *Applied linguistics*, vol. 11, no. 2, pp. 129–158, 1990.

[5] D. M. Hardison, "Generalization of computer assisted prosody training: Quantitative and qualitative findings," *Language Learning & Technology*, vol. 8, no. 1, pp. 34–52, 2004.

[6] Y. Hirata, "Computer assisted pronunciation training for native english speakers learning japanese pitch and durational contrasts," *Computer Assisted Language Learning*, vol. 17, no. 3-4, pp. 357–376, 2004.

[7] A. Germain and P. Martin, "Présentation d'un logiciel de visualisation pour l'apprentissage de l'oral en langue seconde," *Alsic*, vol. 3, no. 1, pp. 61–76, 2000.

[8] G. Molholt, "Computer-assisted instruction in pronunciation for chinese speakers of american english," *Tesol Quarterly*, vol. 22, no. 1, pp. 91–111, 1988.

[9] J. Nouza, "Training speech through visual feedback patterns," in *Fifth International Conference on Spoken Language Processing*, 1998.

[10] Y. Lezhenin, A. Lamtev, V. Dyachkov, E. Boitsova, K. Vylegzhanina, and N. Bogach, "Study intonation: Mobile environment for prosody teaching," in *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*, pp. 1–2, IEEE, 2017.

[11] B. E. Speaking, "(audio lessons) 100 common english phrases and sentence patterns with dialogue.."

[12] M. Zhou and M. H. Wong, "Efficient online subsequence searching in data streams under dynamic time warping distance," in *2008 IEEE 24th International Conference on Data Engineering*, pp. 686–695, 2008.

[13] A. D. Giovanni, "yinjs."

[14] A. de Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *J Acoust Soc Am*, vol. 111, pp. 1917–1930, 2002.

[15] D. Pollock, "Smoothing with cubic splines," tech. rep., Queen Mary University of London, School of Economics and Finance, 1993.

[16] T. Chai and R. Draxler, "Root mean square error (rmse) or mean absolute error (mae)?," *Geosci. Model Dev.*, vol. 7, 01 2014.