Musical Phrase/ Hum Detection*

Catherine Wu and Ruth Mekonnen

***Abstract*— Our system takes in a reference audio of a musical phrase/hum as a passcode and a query audio. Then, it determines if the reference and query audio are a match to ultimately unlock a phone. Our exploration thus far is focused on the back-end rather than the user interface. We check if the query is a match to the reference by running DTW on various features of the audio and it's transposes, accepting the query as a match if the score from DTW is lower than some threshold. This threshold is determined by plotting the ROC curve and picking a threshold that gives at least 80% true positives and at most 10% false negatives from the training data. We have determined that our system works well if we run sequence DTW with possible transitions of [1,2], [2,1], [1,1] each weighted 1, on chroma features with 24 bins. We tested our system on 100 audio samples with audio from 7 people.**

I. INTRODUCTION

Rather than having a numerical passcode to unlock a phone, we wanted to explore the possibility of using a sung or hummed tune as a passcode instead. The user should be able to record a reference audio of a tune they would want as their passcode on their phone, and be able to unlock the phone by re-humming that tune. We call the audio to compare to as our reference audio, and the audio we are comparing to the reference as the query audio. Our problem statement would then be to determine matches between reference and query audios, accepting query audios that are a close enough match to the reference audio. We have defined a query audio to be a close enough match if the query audio: 1) matches the reference in intervallic structure (deviating by at most a semitone per interval) and 2) matches the reference in rhythm (the relative length of a note should not deviate by more than a factor of two). In order to take into account the ability of an average person to reproduce a melody, we decided that queries that are transposes of the reference audio should be considered matches as long as the intervallic structure is a match. In addition, variations in the overall length of a query audio is acceptable as long as each note is varied by a constant or close to constant amount (i.e. the rhythm is unchanged or only changed slightly). The mechanism should not be voice dependent, that is other people humming the correct tune should also be able to unlock the phone. Short silences at the beginning and end of an audio should not affect the audio either.

II. SYSTEM DESIGN

Given a reference audio and a query audio both of which are hums/musical phrases, we want to determine if they are a match. In order to do so, we have two phases, pre-processing which involves extracting features from the time domain representation of our audio, and dynamic time warping (DTW) to compare the query and the references and generate a match score. The match score is a measure of dissimilarity between the reference and query, so a lower match score indicates a more likely match. Hence, if the match score is below a certain threshold (as will be determined by a training dataset), a query should be accepted as a match. To evaluate our system, we look at the resulting confusion matrix and F1 score. For our purposes, we decided to prioritize a lower false-positive rate so that our system does not wrongfully accept a query audio as a match and unlock a phone.

A. Pre-processing

Before comparing two audio files, features need to be extracted from each audio file. We looked into using log frequency spectrogram, Constant Q Transform (CQT), 12-bin and chroma feature vector matrices and 24-bin chroma feature vector matrices as features.

To get the log frequency spectrogram, we first computed the short-time fourier transform (STFT) using the librosa's implementation. Then, we computed the corresponding log frequency spectrogram (85 bins x number of frames) by collapsing the columns of the STFT that correspond to a note in the range of C1 to C8. However, the STFT has a low frequency resolution at lower frequencies which means that it would be unsuitable for analyzing audio with human voice. To increase frequency resolution at lower frequencies, we estimated instantaneous frequencies from the STFT and used that to compute a log frequency spectrogram.

In addition to the STFT, we tried another method for computing log frequency spectrograms directly from the time domain representation of audio. For this method, we used a librosa function, called Constant Q transform (CQT) which automatically has a higher resolution at lower frequencies than the STFT does [1]. We evaluate our system using both methods (STFT and CQT). In both the CQT and the STFT methods, to cut out background noise and silence, we filtered and cropped the log frequency spectrogram. We assumed that our input audio had low volume noise, so to filter, we checked for cells in the log frequency spectrogram that had significantly lower energy than the maximum energy in the spectrogram, and set the energy of that cell to zero. We also assume that the human vocal range is above a B2 (starting at E2) [2], so we zero out all the energy below a B2 (since it probably noise). Then, to crop, we iterated through the frames in the spectrogram, and deleted a frame if after filtering there was no energy in that frame.

However, we found that even after filtering and cropping the CQT and the STFT-log frequency spectrogram alone, they did not perform well enough with the training data to

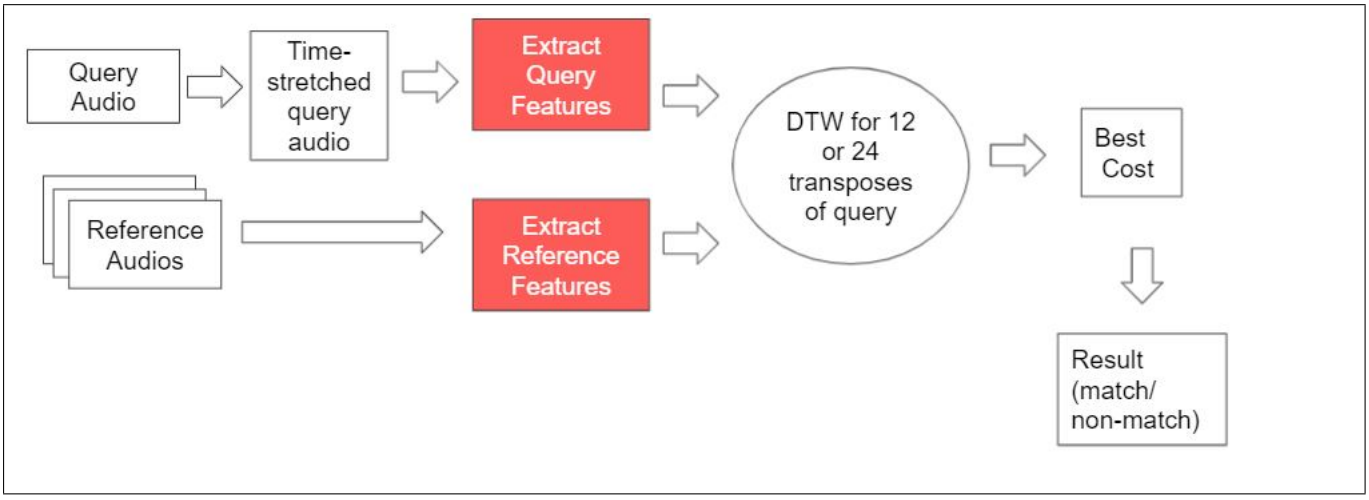

Fig. 1. Block Diagram for comparing reference audios to query audio

use it on the testing data. In order to get a more accurate feature reperesentation of our audio as well as decrease the dimensionality of our data for a more efficient comparison between audios later on, we had collapsed the bins further into chroma feature vector matrices, where each chroma feature vector contained the 12 notes in an octave. So the final chroma feature vector matrix of a particular audio was 12 x number of frames. However, we also found that by collapsing the audio to 12-bin chroma, the ascending to a certain note looked the same as descending to that note. (For example, if you start at C4 and ascend to E4, it looks the same as if you started at C4 and descended to E3). So we expanded our window to 24 notes (two octaves) for the Chroma feature vector matrix which didn't fully solve the problem but had a good tradeoff of performance in preserving the ascension and descension of notes and reducing the dimensionality of the audio data.

B. Comparing Query Audio to Reference Audio

In this phase, our program first takes in the time domain representation of the query audio and reference audio then it computes the 12-bin or 24-bin chroma feature vector matrix for both audios. Then, we use dynamic time warping on the query chroma and the reference chroma to determine a match score. We chose to do DTW instead of subsequence DTW because we defined a match to be when all the sequence of notes in the reference are also in the query (we are not matching part of the reference audio to the query audio, rather we are matching all of the query audio to all of the reference audio).

However, we noticed that when we didn't time stretch the audio of the query to match the reference, DTW was not able to recognize a path with the transitions and weights that we had specified. So we decided to time stretch the uncropped version of the query audio to match the reference audio in length. This got rid of the errors for most of the query audio.

For the DTW, we used cos distance between the query and reference audios as our cost matrix. We want to accept audio if relative pitch is preserved (allowing for a slight difference of one semitone above and below the expected note at a certain point), so we made sure to account for that in our DTW. For a single query chroma feature vector matrix we perform DTW 12 or 24 times (depending on the dimensionality of the chroma feature vector matrix). Each time we do DTW, we shift the chroma feature vector matrix up by 1 semitone and compare this transposed chroma of the query to the chroma of the reference. We do this to ensure that our program is looking for a preservation of relative pitch rather than absolute perfect pitch. Each of these 12 or 24 DTW calculations for a query produces an optimal cost, a measure of the lowest cost of a path through our cost matrix. For the final best cost of the query (which we will use as our match score), we take the minimum of all these optimal costs. If the best cost is below a certain threshold (discussion of how this is determined below), then we predict that the query audio matches the reference audio.

C. Determining Threshold and Evaluating Our System

First, we partitioned our data into training (54 audio files) and testing (45 audio files). Then we used the training dataset to calculate a threshold that would minimize false positive rate but maximize true positive rate and used that threshold on the testing data.

To determine a threshold, we arbitrarily chose a set of 3 reference audios (which had matches in the dataset) into a folder and computed the chroma of each reference audio and the chroma of each query in our training dataset. For each query audio, we computed a best cost with each of the 3 reference audios and stored the best costs for that query audio in an array of best scores. So for each query we had best cost values with each of the 3 reference audios. In order to have a single match score per query, we tried two methods, taking the *maximum* and taking the *average* of all the best scores of a query with all the references. Once, we had a single best score value for each query audio, we used a Receiver Operating Characteristic (ROC) curve to compute false positive rate and true positive rate at several thresholds. Then, we chose a threshold corresponding to a

true positive rate of above 80% and a false positive rate of below 10% because we are focused on minimizing false positives. We will use this threshold to classify our testing data into matches and non-matches.

To determine the accuracy of our system we repeated a similar process, except with the testing data so that we had a single match score for each of the query audios when compared to all 3 references. If the match score was below the threshold, then we predicted the query was a match. At this point we had an array of 1s (indicating matches) and 0s (indicating non-matches) as our predicted labels and we also generated ground truth labels. We manually labeled whether a query audio was classified as a match or a non-match. Then we iterated through both arrays of labels and counted the number of false positives, false negatives, true positives and true negatives. From these values, we also computed accuracy, f1-score, false positive rate and false negative rate.

III. DATA

Our dataset consists of 100 audio samples of various hummed tunes from 7 different people, although the majority of the audio came from our voices (Ruth and Catherine). The majority of tunes being hummed consists of: the first four notes of a major scale (labelled as "Four Notes"), Old McDonalds, and Twinkle Twinkle Little Star. These tunes are hummed with different enunciations (such as "la", "oo", "ee", etc) and are transposed around several different keys. In addition, we have several other nursery rhymes and random notes as queries. All of our audio is within 2 to 8 seconds long, and has low to medium noise.

IV. RESULTS

We had two sets of experiment where we varied what we used as a reference. In our first set of experiments we used Four Notes audio as our references and in the second set, we used Old McDonalds. In each case, we arbitrarily selected three audios that matched the tune we wanted to use as a reference, and compared each query in our training set to each reference. We then recorded both the averaged costs between a query and each reference as well as the maximum cost between query and references, and used the ROC curve to determine which to use as a metric to accept or reject. We then plotted the ROC curves using average and maximum costs and found that maximum cost was marginally better for most cases, having a very slightly higher true positive rate at low false positive rates. Using the maximum cost, we picked a threshold that would give us at least 80% true positives and at most 10% false positives.

To calculate the score to use as a metric, we implemented the system described in the previous section, with some variances to determine the most optimal strategy.

First, we compared the performance using 12 chroma features to using 24 chroma features. We also tried varying the weights of transitions, testing weights of 2 for transitions [1,2] and [2,1] as opposed to weights of 1. We reasoned slightly penalizing steps that weren't exact matches in time could potentially further reduce false positives. Furthermore, we tried calculating the chroma features using CQT rather than from our calculated log frequency spectrogram. In addition to chroma features, we tried using the energy from CQT directly.

For our first experiment, we used three of our Four Notes audio as references. Using 24 chroma features as opposed to 12 didn't have a significant effect on the result percentages. However, we found that changing the transition weights from 1 to 2 significantly decreased the performance of the system. When plotting the ROC for the training data, the curve was below the diagonal, signaling a very inaccurate system. In addition, we found that using maximum costs to determine a threshold gave us less accurate results than using average costs on our testing data. When using 12 chroma features and weights of 1 with maximum cost, we determine a suitable threshold to be 149 from the ROC curve of our training set. Using maximum best cost to determine a threshold yields us an accuracy of 87%, a F1 score of 77%, a false positive rate of 17%, and a false negative rate of 0% on the testing set. In contrast, using average best cost gives us a threshold of 127 and an accuracy of 93%, a F1 score of 87%, a false positive rate of 9%, and a false negative rate of 0% on the testing set. Thus, we determined that the better ROC curve we observed when using maximum cost was an instance of overfitting, and we decided to use average cost to determine threshold for the rest of our analysis especially since the maximum ROC curve was only marginally better anyways. Also, increasing the chroma features to 24 did not change our results, with the thresholds determined (using average) from the training set to be 123, and exactly the same percentages for accuracy, F1, false positive rate and false negative rates as mentioned.

TABLE I
CONFUSION MATRIX FOR FOUR NOTES USING 24 CHROMA FEATURES AND WEIGHTS OF 1

-	Predicted Positive	Predicted Negative
Actual Positive	10	0
Actual Negative	2	32

For our next set of experiments, we used three of our Old McDonalds audio as references. In this case, we saw a more noticeable difference between using 12 versus 24 chroma features. When using 12 chroma features and weights of 1, we determined a threshold of 207. This gave us an accuracy of 84%, a F1 score of 72%, a false positive rate of 19% and a false negative rate of 0%. Hence, this approach does not satisfy our evaluation metric. When using 24 chroma features and weights of 1 however, we find a threshold of 211, which gives us an accuracy of 97%, a F1 score of 95%, a false positive rate of 3% and a false negative rate of 0%.

We also tested our system using CQT to calculate the chroma features, specifically using 24 chroma features and weights of 1 since this yielded the best results for the Old McDonald tune. This method was slightly faster than manually calculating the log frequency from the STFT, although less accurate. We determined this by timing each

TABLE II
CONFUSION MATRIX FOR OLD MCDONALDS USING 24 CHROMA FEATURES AND WEIGHTS OF 1

-	Predicted Positive	Predicted Negative
Actual Positive	9	0
Actual Negative	1	35

benchmark run on the same computer; on average the non CQT method took 14 minutes while CQT took 10 minutes. However, we recognize this is a very naive approximation of runtime. Meanwhile, our threshold for CQT is 270, and we have an accuracy of 89%, a F1 score of 78%, a false positive rate of 14%, and a false negative rate of 0%. The false positive rate for this method exceeds the acceptable standard of our evaluation metric.

TABLE III
CONFUSION MATRIX FOR OLD MCDONALDS USING 24 CHROMA FEATURES, WEIGHTS OF 1, AND CQT

-	Predicted Positive	Predicted Negative
Actual Positive	9	0
Actual Negative	5	31

Finally, we tried running DTW on the directly on the cropped and filtered log frequency spectrogram energies as well as CQT energies. We reasoned that comparing the log frequency spectrogram and CQT before collapsing the columns to form chroma would help us preserve the ascension and descension of notes for even large jumps in the pitch. In both cases, we were not even able to determine a suitable threshold from the ROC curve, as no threshold was able to satisfy our condition of 80% true positives and 10% false negatives. Therefore, we determined these approaches were not sufficient for our application.

V. DISCUSSION

One concern we had with our method of pre processing the data was that our system would not work well for very noisy data. This is because we found just the fundamental frequency by determining the index where there first was energy after cropping and filtering, since we assumed all instances of noise would be zeroed out. However, if we are unable to successfully crop and filter because the noise was extremely high, we could possibly incorrectly identify regions of unfiltered noise as the fundamental frequency. As a next step, we could work to make our system still work with noisy audio.

From the collected data, we determined that the most optimal strategy would be to use 24 chroma features and keep the transition weight for [1,2] to be 1, as this was the method that gave us the best results for Old McDonalds.

Although 24 versus 12 chroma features didn't appear to make any difference for when the Four Notes was a reference, we recognize this is because our dataset does not include audio where we descend to the correct note rather than ascend to it. On the contrary, this issue does affect the accuracy of Old McDonalds, since Old McDonalds and Twinkle Twinkle Little Star share all of the same notes but have different melodic contours. As per our problem statement, these two melodies should not be equivalent, but have the exact same chroma feature graph if we only use 12 chroma features, hence the high false positive rate. We fix this issue for this instance by increasing our chroma features to 24. This method does not solve this issue for all instances, for example, ascending from a C4 to a C5 will still be identical to descending from a C4 to a C3. However, making notes within every two octaves unique rather than every octave greatly reduces the chance of wrapping around to the correct note in the opposite direction, since at least one of the matching query or reference tunes would need to leap by at minimum an entire octave to introduce this error. We could try to generalize this problem as a next step, somehow incorporating ascension/descension information instead of simply increasing chroma features.

For the weighting, increasing the transitions [1,2] and [2,1] to have a weight of 2 did increase the score for queries of unacceptable (by our problem statement) rhythms. However, it also dramatically increased the score for many matching query audios, since many of the matching queries did not maintain the exact tempo of the reference. Since we did state that tempo varying by a factor of 2 should still be accepted, keeping the weights as 1 made sense.

One last observation we noticed was that the threshold for Old McDonald was higher than that of Four Notes. As expected, the threshold for more complicated audio with more notes needs to be higher compared to simpler audio. More importantly, we don't have a general threshold we can apply to any arbitrary tune; instead, we need to determine this threshold by comparing queries in the training set to reference audios. Since the ultimate goal of our system is to work as a method to unlock a phone, we note that it would be beneficial to have a more robust way of determining a threshold without having to run the system on a set of data that is known to be matches and non-matches. In the context of unlocking a phone, the user probably would only be able to input several reference audios but not necessarily get an exhaustive dataset of non-matches, and thus wouldn't be able to determine a very good threshold.

REFERENCES

[1] Schörkhuber, C. and Klapuri, A., Constant-Q transform toolbox for music processing, in 7th Sound and Music Computing Conference, Barcelona, Spain, 2010, pp. 3-64.

[2] Kob, Malte, et al. Analysing and understanding the singing voice: recent progress and open questions. Current bioinformatics 6.3, 2011, pp. 362-374.