OLD MACDONALD HAD A HARMONY: AN ANIMAL-VOICED HARMONIZER 2019

Yun Ning Hung
Georgia Institute of Technology
yhung33@gatech.edu

Punahamoa Walker
Georgia Institute of Technology
pwalker40@gatech.edu

Lisa Anne Zahray
Georgia Institute of Technology
lzahray3@gatech.edu

ABSTRACT

We have developed a harmonizer which creates a four part harmony of animal vocalizations based on a monophonic input recording. The system is broken down into four parts: music parameter detection, audio transcription, harmony generation with voice leading, and audio processing of the animal vocalization. The input audio is transcribed using a state-of-the-art pitch tracking algorithm. The harmony is generated using a Hidden Markov Model trained on Bach chorales. We then use a rule-based method to penalize voice leading transitions, and find the optimal path using the Viterbi algorithm. Finally, we pitch shift animal sounds to generate the final audio file. We analyze our system by evaluating each subsystem.

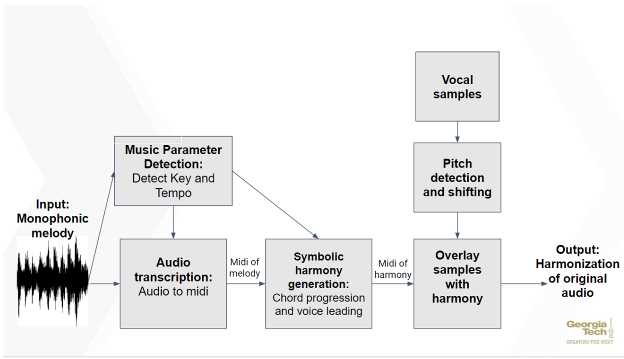
Input: Monophonic melody
Music Parameter Detection: Detect Key and Tempo
Audio transcription: Audio to midi
Midi of melody
Symbolic harmony generation: Chord progression and voice leading
Midi of harmony
Vocal samples
Pitch detection and shifting
Overlay samples with harmony
Output: Harmonization of original audio
Georgia Tech
CREATING THE NEXT

Figure 1. The work flow of our harmonic generation system.

1. INTRODUCTION

A harmonizer traditionally combines pitch-shifted audio samples to produce a harmony with the original audio. This paper outlines the implementation of our animal-voiced harmonizer. It generates a four-part harmony, voiced by animals, in response to a melody audio file.

The motivation behind this project is that while traditional harmonizers may appeal to musicians, we believe that incorporating animal sounds will broaden the appeal of this system to also include non-musicians. This project does not aim to advance techniques in any one particular area, but rather to be a novel implementation of existing techniques and subsystems.

This paper will first outline related work, then discuss the implementation of each subsystem, and finally provide evaluation and discussion.

2. RELATED WORK

One of the first attempts at harmony generation in the style of Bach chorales was a rule-based system built by Ebcioglu in 1988 [4], which uses approximately 350 rules. Later approaches were more data-driven. For example, Allan and Williams [1] used a Hidden Markov Model (HMM). They used melody pitches as observations, and the intervals of each voice relative to the melody pitch as hidden states, with the addition of harmonic labels for chord function. They quantized time to each beat, adding ornamentation such as passing tones later through a second HMM. Their system does not directly take the key of the piece into account. Liang et al. used recurrent neural networks with long short-term memory (LSTM) [8] in their system BachBot, first transposing each piece into C major/A minor and quantizing time to sixteenth notes. Hadjeres et al. [6] presented DeepBach, which uses a dependency network with pseudo-Gibbs sampling to generate Bach chorales. Advantages of their system include flexibility over which voice(s) to keep fixed and which to reharmonize, and not being constrained to sequential generation.

For monophonic pitch tracking, autocorrelation is the most widely used method. There are several autocorrelation-based algorithms such as YIN [3] and probabilistic YIN [9]. The algorithm SWIPE [2] takes a different approach. It estimates pitch as the fundamental frequency of the sawtooth waveform whose spectrum best matches the spectrum of the input signal, and has high accuracy on speech and musical instrument pitch tracking. However, due to the development of deep neural networks, state-of-the-art pitch tracking algorithms are achieved by CREPE [7], which uses data-driven methods and operates directly on the time-domain waveform. Since CREPE performs better than other algorithms in monophonic pitch tracking [7], we adopt the CREPE algorithm for our pitch tracker and add some post-processing to transcribe the melody midi file.

3. SYSTEM IMPLEMENTATION

Our system consists of four sections. First, we detect the key and tempo of the input monophonic audio. Using this information, the second section transcribes the audio to a midi file. The third section uses this melody to generate a four-part harmony midi file. The final section voices the harmony by pitch-shifting and time-stretching animal sounds, producing the final audio file. An overview of our system is shown in Figure 1. Each block will now be described in greater detail.

3.1 Music Parameter Detection

To improve the audio transcription process, key and tempo are first extracted from the audio. The tempo of the input is estimated using a short time Fourier transform novelty function to determine the median inter-onset interval. For the key detection algorithm, we compute the pitch chroma for the input audio and use the Krumhansl templates for selecting the key candidate.

3.2 Audio Transcription

With knowledge of the key and tempo, our system then runs the pitch tracker CREPE [7], a deep convolutional neural network based algorithm, to get the pitch contour probability. For the post-processing part, our goal is to process the pitch contour into a midi file which can be used in the harmony generation system. The frequency contour is converted into music notes, and we then run a median filter to smooth out any vibrato. Finally, we quantize the pitch contour into eighth note as the smallest unit to generate the output midi file.

3.3 Harmony Generation

From the melody, we use a Hidden Markov Model to generate the chords, and a rule-based system for voice leading. We use music21's [1] corpus of Bach chorales. For simplicity, we ignore chorales that change time signature or key signature throughout the piece. All chorales are transposed into the key of C major/C minor. As in [1], we quantize time to the beat level. Since our voice leading is separate, we represent the data using pitch classes, rather than absolute pitch. Each observation is a 24-element one-hot vector representing the 12 pitch classes of the melody, along with whether or not it is on a downbeat. States are all pitch classes that exist on a beat, taking the first non-rest note for each voice within the beat. We use two separate models for major and minor keys. We ignore pickup measures for start probabilities, instead treating the first beat of the first full measure as the start state. We record transition probabilities between the 30 most-common chords. This number was chosen as an approximate border point where the number of instances of less-common chords became too low to provide meaningful probabilities. We perform additive smoothing by adding .001 to all start probabilities, transition probabilities, and emission probabilities for which the melody and chord combination is valid.

[1] https://web.mit.edu/music21/doc/index.html

Rule	**Penalty**	**Avg. penalty test data (major, minor)**	**Avg. penalty HMM (major, minor)**
Parallel fifths/octaves	2	0.061, 0.069	0.000, 0.000
Doubling third of major triad	1	0.096, 0.064	0.000, 0.000
Leap of tritone	1	0.014, 0.020	0.001, 0.000
Leap in alto or tenor line	0.1 per semitone	0.600, 0.551	0.109, 0.113
Doubling leading tone in key	2	0.034, 0.010	0.000, 0.000

Table 1. Voice leading scoring system and results. Average penalties are per transition

Voice leading has numerous rules and guidelines, some of which have exceptions in certain contexts. Our system has unique constraints of an unknown range for the input soprano line and a limit on how much each animal sound can be reasonably pitch shifted. We therefore implement a custom rule-based system to satisfy our constraints as well as some of the main rules of voice leading. First, we limit the alto, tenor, and bass to a range of one octave, starting at midi pitches 60, 53, and 46, respectively. We shift the melody line by octaves such that the most possible notes are contained between the pitches 67 and 78, inclusive. We then enumerate all possible voicings for each chord, including all potential options for doubling. These voicings are used as nodes in a trellis for the Viterbi algorithm. Each transition is scored according to the rules in Table 1. The Viterbi algorithm is used to select the least-cost path through the trellis, thus selecting the voice leading option with the least penalty.

3.4 Audio Synthesis

After being provided with midi data from the harmony generation section, the system then needs to synthesize audio. The first step was to choose vocalizations of animals which would be suitable for this application and develop a database. We decided on 3 criteria for selecting audio samples to minimize the amount of pitch shifting and time stretching. The samples were selected manually according to these criteria.

The first criterion is that the animal vocalizations need to be a suitable length. We have found that an audio sample with a length of around 0.7-1 seconds can be time stretched while remaining natural-sounding within a tempo range of 70-140 beats per minute. The second criterion is that the natural frequency of the chosen animals should be comparable to their human vocal part counterpart. Finally, the pitch of the audio sample should remain relatively constant. We experimented and decided that it would be better to pitch shift the whole audio sample by the same amount rather divide the audio in blocks and pitch shift each block to the correct pitch. We felt that it was less important for the vocalizations to be perfectly in tune but rather maintain the characteristic cadence of the each animal.

In order to pitch shift our samples, we implemented pitch synchronous overlap and added (PSOLA) [5]. The audio sample is resampled according to the ratio of the desired pitch and the average pitch of the sample. This achieves the desired pitch but will also change the length of the sample. To achieve the desired length, the sample is divided into overlapping blocks that are twice the length of the period and overlap by 50%. Each of these blocks then had a periodic Hann window applied to them. At a regular interval, blocks are either repeated or deleted. These blocks are then resynthesized into a single dimensional vector and combined with the other melodic lines.

4. EVALUATION

Due to the unique nature of this system, we evaluate the individual parts, rather than evaluating the whole system.

4.1 Audio Transcription and Music Parameter Detection

As we were unable to find a dataset that contains both monophonic melody line along with its midi transcription and tempo, we randomly downloaded 20 pieces of synthesized audio and midi from Musescore forum. [2] We compare our pitch transcription result with the ground truth labels, achieving a note accuracy of 0.762 and an overall accuracy of 0.853, which are calculated by using python library: mir_eval [10]. We also used this database to evaluate tempo detection, manually annotating the BPM. This provided an RMS error of 22.6 BPM. However, when we remove the outliers present at half or double the ground truth, our RMS error is 4.46 BPM.

4.2 Harmony Generation

Our dataset consists of 174 major Bach chorales (139 training, 35 test) and 172 minor Bach chorales (137 training, 35 test). We run our HMM on each song in the test sets, and record two different metrics. First, we find the percentage of correctly guessed chords. While this gives some indications about model performance, it does not provide insights into how well transitions and chord diversity are modeled. We additionally calculate the average posterior probabilities of the ground truth states. This is the probability the model assigns to each actual state, given the melody. States that exist in the test data, but not our 30-state model, are assigned probability 0. We also provide these metrics for the training set as a comparison. The results can be seen in Table 2.

The evaluation metrics between test and training sets have similar values, indicating good generalization of the model. However, our model tends to overly favor more common chords, resulting in less chord diversity than the Bach chorales. The major songs show better results than the minor songs. One observation is that for minor songs, the HMM favors certain chords in uneven proportions to their ground truth frequency. This may be due to structural aspects of minor chorales that are not taken into account by the HMM. For example, minor chorales often end on a major I chord, resulting in a high self-transition probability for this chord of 63%. Therefore, if the model transitions to this chord in the middle of the piece, it is unlikely to transition away from it. One potential modification to account for this could be to use backwards, rather than forward, transitions.

	Test Major	**Training Major**	**Test Minor**	**Training Minor**
Avg. posterior prob. of ground truth states	16.86%	16.20%	12.88%	12.38%
Percentage of correctly predicted chords	43.02%	42.93%	37.68%	38.38%

Table 2. Results of chord evaluation on 30-state HMM

Due to our simplified approach to voice leading, which does not aim to fully emulate Bach's style, we evaluate success based on the rules used in our system. Table 1 shows the average penalty per transition for each rule, comparing the ground truth data of the test set to the results of our algorithm. Our system avoids breaking almost any rules, but may be too strict, particularly when punishing large leaps in the alto and tenor lines.

4.3 Audio Synthesis

Despite using animal performers rather than trained singers, we want to make a system which is pleasing to listen to, albeit slightly ridiculous. We found that we had to implement other restrictions to our system. We found that using four different animals introduces too many different timbres which is distracting from the actual music. We use the same individual animal per vocal line for the same reason. In addition to this, any long notes were converted to repeated notes so that the samples were not stretched to unnatural sounding lengths. Working within these confines allowed us to produce fairly pleasing sounding audio.

5. CONCLUSION

Our system converts a monophonic audio recording into a four-part chorale sung by animals. We have achieved reasonably accurate transcription, as well as developed a system which imitates the style of Bach's chorales. Using these, we have produces audio sung by different animals, and addressed some of the unique challenges presented by our selected performers' vocal timbres. We believe that we have achieved our goal of producing a harmonizer with musical merit, as well being unusual and amusing enough to appeal to non-musicians.

[2] https://musescore.com/dashboard

6. REFERENCES

[1] Moray Allan and Christopher K. I. Williams. Harmonising chorales by probabilistic inference. *Advances in neural information processing systems*, 17:25–32, 2005.

[2] Arturo Camacho and John G Harris. A sawtooth waveform inspired pitch estimator for speech and music. *The Journal of the Acoustical Society of America*, 124(3):1638–1652, 2008.

[3] Alain De Cheveigné and Hideki Kawahara. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

[4] Kemal Ebcioglu. An expert system for harmonizing chorales in the style of j. s. bach. *Journal of Logic Programming*, 8:145–185, 1990.

[5] M. Stella F. Charpentier. Diphone synthesis using an overlap-add technique for speech waveforms concatenation. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1986.

[6] Gaëtan Hadjeres, François Pachet, and Frank Nielson. Deepbach: a steerable model for bach chorales generation. In *International Conference on Machine Learning*, 2017.

[7] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. Crepe: A convolutional representation for pitch estimation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165. IEEE, 2018.

[8] Feynman Liang, Mark Gotham, Matthew Johnson, and Jami Shotton. Automatic stylistic composition of bach chorales with deep lstm. In *International Society for Music Information Retrieval Conference*, 2017.

[9] Matthias Mauch and Simon Dixon. pyin: A fundamental frequency estimator using probabilistic threshold distributions. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663. IEEE, 2014.

[10] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. mir_eval: A transparent implementation of common mir metrics. In *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.