So You Think You Can Sing?

Karishma Agrawal, Lu Lyu, Tomer Borenstein, Akshay Kulkarni

November 9, 2015

1 Introduction

In this project we are going to attempt to create a program that takes a song with instrumental and vocal components, separates the singer's vocals from the instrumental portion of the song, modifies the user's vocals to match the pitch and timing of the original singer's vocals, and inserts the modified user's vocals back into the song to make it sound like the user was originally singing the song.

While this may not be completely world changing, we are motivated by the idea that using such a program (perhaps in a Mobile Application form) has significant entertainment value to many people who may not be vocally trained.

While the concepts and technologies required to complete this project have already been developed, no one has combined them into this one single entertaining application thus far, so we hope to be the first group that achieves this goal.

2 Overall Approach

2.1 Input Data

The input to our system is the original song which the user wishes to imitate and the sing-along for the same provided by the user. In later iterations, we believe the user can just read the lyrics and we would be able to give reasonable performance regardless. However in our initial approach we expect the user to sing along which can act as guide to aid the vocal separation. This is essential for NNMF based vocal separation.

2.2 Tasks

- Vocal Separation: Separate the vocal and instrumental components from the original song. Possible technologies to use include Robust PCA and NMF.
- Vocal Alignment: Use Dynamic Time Warping on the separated vocal and user singing/speaking lyrics recording to match up the rhythm.
- Pitch Adaption: First use base frequency extraction to find the pitch information of the original vocal recording as well as the user recording. Then match the user recording pitch to the original vocal pitch.
- Synthesizing The Song

3 Work Done So Far

3.1 RPCA

Robust PCA refers to decomposition of a matrix into a low rank matrix and a sparse matrix. [2]

$$M = L_0 + S_0$$

Here M refers to a large data matrix which is decomposed into the low rank L_0 and sparse S_0.

The idea behind this is that the music would have repetitive patterns because of the nature of the instruments and hence can be captured in the low rank subspace. On the other hand the vocals would be sparse but more can vary much more. To decompose the matrix we can solve

$$\begin{aligned} \text{minimize } & |L||_* + \lambda ||S||_1 \\ \text{subject to } & L + S = M \end{aligned}$$

An additional assumption is that the low rank matrix cannot be sparse. This is done to ensure a unique solution.

For our work, we used the library functions for RPCA [3]. We fed in the *smag* format of the song obtained after fast fourier transform into the RPCA function and reconstructed the separated components which we extracted. Additionally, binary time-frequency masking is applied to get rid of some obvious noise.

$$M_b(m,n) \begin{cases} 1 & \text{if } |S(m,n)| > gain * |L(m,n)| \\ 0 & \text{otherwise} \end{cases}$$

$$\text{for all } m = 1...n_1 \text{ and } n = 1...n_2$$

This mask is applied to the original STFT matrix

$$\begin{cases} X_{singing}(m,n) = M_b(m,n)M(m,n) \\ X_{music}(m,n) = (1 - M_b(m,n))M(m,n) \end{cases}$$

$$\text{for all } m = 1...n_1 \text{ and } n = 1...n_2$$

These correspond to the audio and the instrument part. This method is not error free. Cases where the vocal and music are both sparse can be difficult to separate. [1]

3.2 Base Frequency

We implemented a base frequency extraction function in Matlab. Base frequency is a good indicator for pitch of a recording at a given time. For our major motivation for pitch extraction is to modify the user input audio into a meaningful ‘singing’, we could use pitch and base frequency interchangeably to achieve the goal.

The main idea of the base frequency extraction is to sum up the energy of each base frequency and its harmonious frequencies, then find the peak at each time frame. The base frequency is limited to between 20 Hz 100 Hz, where the human voice usually falls in. While adding up the energy of a certain

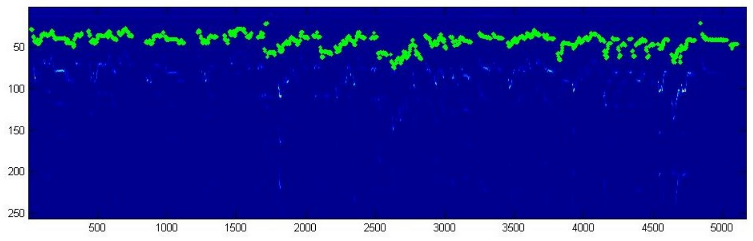

Figure 1: Base Frequency Extraction

frequency, a window function [0.1 0.3 0.8 1 0.8 0.3 0.1] centered at the target frequency is used to smooth the result. This method works mainly for pure vocal recording. Minor noise will not affect the performance, for they will not form a peak anytime. The demo of the function is shown in Figure 1

4 Dynamic Time Warping

Dynamic time warping measures the similarity between two time series and match them up. It calculates the similarity matrix in full or part, then find the best path that connects the diagonals. Visualization of DTW is shown in Figure 2. We experimented matching the singing recording to the original song directly, but no good result was obtained. The plan for this part is to match the separated vocal provided by RPCA and the singing/speaking recording. Using MFCC to extract audio features will possibly help as well. Details about MFCC will be introduced in the next section.

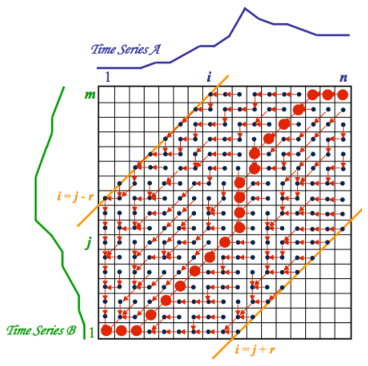

Figure 2: Dynamic Time Warping

5 Results

See attachments for the audio result.

6 Work Currently in Progress

6.1 Mel-Frequency Cepstrum

Recall that the goal of this project is, for a given song, to take the original singer's vocals out of the song, and replace them with the user's vocals in a manner which makes the user's vocals sound good. Essentially, once we have separated the singer's vocals from the background music via Robust Principle Component Analysis we would want to warp the speed and pitch of the user's vocals to match the original singer's so they fit well within the song, which can be done with Dynamic Time Warping and Pitch Modification.

Before we can proceed with dynamic time warping and pitch modification, however, we need to find a way to "match" the user's vocals to the original singer's; that is, we need to be able to determine where the user and the original singer sing the same lyrics. To do that we need to be able to extract and match features from the two sets of vocals. Since the end goal of this project is to make anyone sound good when singing, we cannot make any assumptions about the user's singing being the correct pitch, and therefore need to extract features that are pitch-invariant.

The Mel-frequency Cepstrum (MFC) [4] is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. It is often used to extract features from sound in the context of speech recognition. MFC is also pitch invariant, which means that it will be ideal for our purposes.

The process of extracting the Mel-frequency Cepstral Coefficients (MFCC) is as follows:

- Frame the vocal signal into short frames.
- For each frame, calculate the periodogram estimate of the power spectrum to identify which frequencies are present in the frame.
- Apply the mel filterbank to the power spectra, and sum the energy in each filter.
- Take the logarithm of all filterbank energies.
- Take the Discrete Cosine Transform (DCT) of the log filterbank energies.
- Discard all DCT coefficients except 2-13.

Once we follow this process or use an existing Matlab implementation we should be able to find matching points in time where the original singer and the user are singing the same lyrics. One thing to watch out for, however, is that the MFCC are very sensitive to noise, so we may want to normalize their values in order to make them more robust, especially since we expect the original singer's separated vocals to have some amount of noise in them.

6.2 Non Negative Matrix Factorization

NNMF technique is to decompose a matrix say A into two matrices W and H with the property that all the 3 matrices have non-negative elements.

$$[W, H] = nnmf(A, k) \tag{1}$$
$$A - \ the\ given\ NXM\ matrix\ (the\ song\ in\ our\ case) \tag{2}$$
$$W - \ weight\ matrix\ NXk \tag{3}$$
$$H - \ base\ matrix\ kXM \tag{4}$$
$$k - \ the\ matrix\ rank \tag{5}$$

Until now we were using the Robust PCA to separate the vocals from the music. The results we observed weren't that good. So we thought of using the NNMF to learn the bases. Initially we tried to learn the bases from individual pieces of music and vocals and we used these weights to reconstruct a different piece. The reconstruction from this was pretty bad. We also tried to get a better result by increasing the dimensionality (increase the 'k' value in the equation) but the results were not good.

Now for further implementation we have decided to change the approach. We see that the musical part in general has a more structure to it compared to the vocal part. We plan to learn the bases by using the NNMF technique only on the musical part of the song (the part of the song without any vocals). We will use these bases to reconstruct the song. Using this approach we should be able to get better separation of the vocals and the music in a song. [5]

7 Timeline

- voice separation – done
- voice alignment – 1 week
- pitch adaption – 2 weeks
- synthesizing the song – 2 days

References

[1] Alex Berrian. Music voice separation. 2014.

[2] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, June 2011.

[3] P.-S. Huang, S. D. Chen, P. Smaragdis, and M. Hasegawa-Johnson. Singing-voice separation from monaural recordings using robust principal component analysis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 57–60, 2012.

[4] James Lyons. Mel frequency cepstral coefficient (mfcc) tutorial. 2013.

[5] T. Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(3):1066–1074, March 2007.