Pipe Organ Stop Identification via Non-Negative Matrix Factorization

Autumn Coe
Department of Electrical and Computer Engineering
University of Rochester
acoe@ur.rochester.edu

ABSTRACT

Pipe organ stop identification presents both a unique problem and a unique advantage in the instrument identification field. Most instrument identification algorithms do not work well when the instruments are playing the same note due to the overlap in identifiable properties (peaks in the spectral or cepstral domain). This is typically a minimal problem since instruments rarely play in unison for any length of time. However, because multiple pipe organ stops are frequently played from the same keyboard note, the problem of identifying unison instruments comes up frequently. The advantage to pipe organs is that they employ a non-expressive mechanism to produce sound causing the timbre of each note to be remarkably stable. This paper attempts to take advantage of this stability to train a note "dictionary" matrix from recorded training audio using non-negative matrix factorization. The unknown stops in testing audio are then identified by evaluating a sparse activation matrix obtained via the LASSO algorithm (Least Absolute Shrinkage and Selection Operator) and the trained dictionary matrix. The results can be viewed as a graph or through a GUI designed to display the predicted stops in time with the audio they were predicted from.

1. INTRODUCTION

1.1 A Brief Description of Pipe Organs

Pipe organs, like the modern digital keyboard, allow the player to select different sounds with which to play by turning on a 'stop'. Unlike the modern keyboard, multiple stops can be turned on at once. This means that for each note that is depressed on the keyboard, a pipe will sound for each stop that is turned on.

There are four main families of pipe organ stops: Principals (the typical organ sound), Strings, Flutes, and Reeds. Although most of these sound like orchestral instruments (both in name and in timbre), they are all pipes (Principals, Strings, Flutes) or beating reeds with resonators (Reeds). Any instrument names used in this paper, unless specifically indicated otherwise, refer to organ stops, not orchestral instruments.

Each stop corresponds to a 'rank' (group) of pipes that is played by allowing pressurized air to flow through the pipe associated with the desired note. Since the pressurized air in pipe organs comes from a reservoir maintained at a constant pressure rather than the human lungs, pipe organs are not dynamic instruments. Each time a pipe is played, it sounds exactly the same no matter how quickly or hard the key playing it is depressed.

Because pipe organs are frequently historic instruments (some are hundreds of years old) and not well regulated, each stop on each pipe organ is unique, although some are very similar. Thus, even though a single pipe organ might have multiple flute stops (Rohrflöte, Spilflöte, ect), they will not sound the same. Likewise, if two different organs have stops with the same name (say each organ has a Rohrflöte for example) they also will not sound exactly the same.

Finally, organ stops sound at different octaves indicated by a number next to the stop name. This number is the average length in feet of the lowest note, and thus the longest pipe, of the rank. An 8' stop has the same octave pitch as a piano. When the A4 key is pressed, the resulting pitch is 440Hz. Each time the number is halved, the pitch goes up an octave. When the A4 key is pressed with a 4' stop on, the resulting pitch is 880Hz.

1.2 Instrument Identification Methods

The current prominent instrument identification methods are hidden markov, source filter, cepstral coefficient, and neural networks. Hidden markov models work well on single instrument identification, but rapidly lose accuracy when polyphony is introduced. The source filter and cepstral coefficient methods work well for polyphony, but do not handel instruments played in unison or octaves very well. Since unison between stops is a common occurrence in pipe organs, these methods were avoided. Neural networks, while quite promising, are time consuming to train and require more annotated training data than I have access to.

I decided to use non-negative matrix factorization (NMF) because it takes advantage of the additive linearity of the fourier transform. Just as a mixture audio is the sum of all the monophonic audios, a mixture spectrogram (as long as it is not in dB) is the sum of all the monophonic spectrograms. If I created a dictionary

with an element corresponding to the spectrogram of each note of each stop, I could use NMF to determine which combination notes are being activated when in a given test audio clip to produce the mixture. Since the audio (and thus the spectrogram) is just the sum of the different sops being played, modeling it as a linear combination of dictionary elements makes sense.

In practice, the stops are too similar for this to work well. The dictionary can be easily created, but the activation matrix resulting from NMF is too muddled. All stops activate for a single note instead of just the stops being played. To combat this, I used the LASSO algorithm to create a sparse activation matrix. This reduces the overall activations to clean up the muddle while still remaining flexible enough to allow the stops that are detected to be activated

2. METHOD

2.1 Non-Zero Matrix Factorization

The dictionary is comparatively easy to make for an organ. Each note of each stop is unique and has no dynamic variation, so the training data does not require multiple iterations of the same note. A drawback is that a separate dictionary must be made for each organ since the stops on each organ are unique.

To make the dictionary, I recorded a chromatic scale from lowest note to highest note for each stop. I converted it to a spectrogram using a hamming window of 100ms (4410 samples) and a 50% overlap. I then used the recorded scale and NMF to train a dictionary with the same number of elements as number of keys (58 on the keyboards and 30 on the pedalboard). To make sure the dictionary elements were harmonically coherent, I used the harmonic comb initiation of the dictionary matrix W as described in [1] (see fig. 1)and the euclidean distance multiplicative updates for W and H from [2].

$$W_{ia} \leftarrow W_{ia} \frac{(VH^T)_{ia}}{(WHH^T)_{ia}} \quad (1)$$

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^T V)_{a\mu}}{(W^T WH)_{a\mu}} \quad (2)$$

Where W is the dictionary matrix, H is the activation matrix, and V is the matrix being factored (the spectrogram of the input audio).

The euclidean distance method was required to prevent division by zero since many of the W elements are initialized (and remain) at zero in the harmonic comb initialization method. The advantage to this method of forming the dictionary is being able to keep the input as an entire scale instead of having to split it into individual notes. Once dictionaries for each stop were trained, I combined them into a master dictionary by concatenating the matrices.

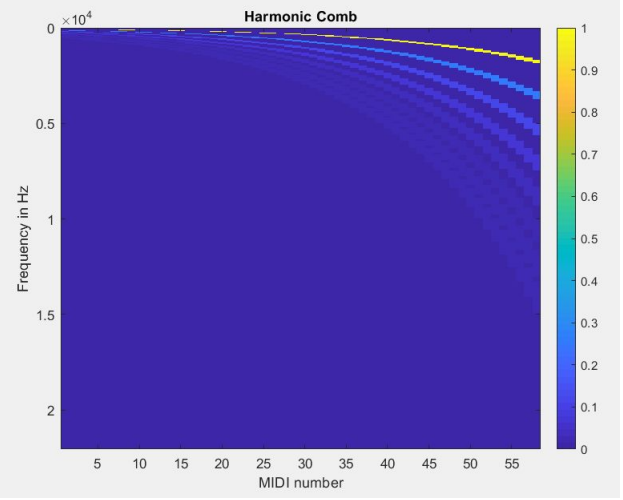

Figure 1. Harmonic comb initiation of the dictionary matrix W

2.2 LASSO Algorithm

To calculate the activation matrix, I used the built in LASSO function in matlab that computes the algorithm as follows:

$$\min_{\beta_0,\beta}\left(\frac{1}{2N}\sum_{i=1}^{N}\left(y_i-\beta_0-x_i^T\beta\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|\right) \quad (3)$$

Where N is the number of observations, y_i is the response at observation i, x_i is a data vector of length p at observation i, λ is a nonnegative regularization parameter, β_0 and β are a scalar and a vector of length p respectively [4].

The inputs were the trained dictionary as x and each frame of the mixture spectrogram as y. The activation vector for each frame was the first column of the output matrix of the LASSO function, min β_0 and β. To make the activation matrix, each activation vector was placed in its corresponding frame column.

The LASSO function in matlab is pretty slow before running it for each 100ms frame of an audio clip, so I sped things up by limiting the number of iterations in the LASSO algorithm to 100. I found this did a good job of finding the activations without taking forever. I also removed the window overlap when calculating the mixture spectrogram to reduce the total number of frames by 50%.

2.3 Stop Determination

To determine if a stop is being played, the rows of the activation matrix are divided into sections representing each stop. The maximum activation value of each frame is taken for each section and thresholded to determine if the corresponding stop is active. The threshold is set at 50 from observation of the data. See fig. 2 for an example.

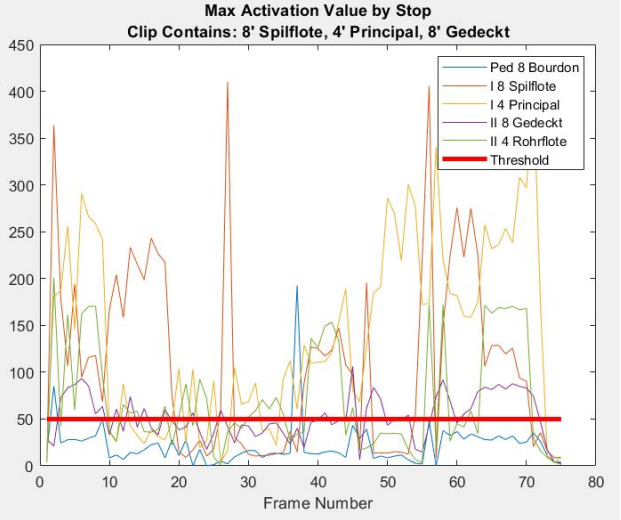

Figure 2. Max activation value by stop

3. CONCLUSIONS

3.1 Testing and Accuracy

The proposed method was tested on recordings from the Wahl practice organ at the Eastman School of Music. Note dictionaries were trained for the following stops: Pedal 8' Bourdon (a flute stop), First Manual 8' Spilflöte (a flute stop), First Manual 4' Principal (a principal stop), Second Manual 8' Gedeckt (a flute stop), and Second Manual 4' Rohrflöte (a flute stop). The dictionaries were tested on 11 recordings ranging from 3 to 12 seconds. Each recording contained a combination of 3 stops.

While the data is very noisy, the results were surprisingly accurate. The overall precision was 0.729 with a recall of 0.502 and an F1 score of 0.594. This is lower than the best F1 score of 0.661 for multi source instrument identification in [3], but remarkably close.

Breaking down the F1 score by stop reveals significant variation. This variation correlates to the number of appearances of each stop in the testing data as shown in table 1. A larger data set for testing is clearly needed to determine the true accuracy of this method.

Stop Name	Precision	Recall	F1 Score	Number of audio
8' Bourdon	0.809	0.391	0.527	6
8' Spilflöte	0.892	0.524	0.660	9
4' Principal	0.660	0.437	0.526	5
8' Gedeckt	0.929	0.524	0.670	10
4' Rohrflöte	0.447	0.674	0.537	4

Table 1. Precision, Recall and F1 score by stop

3.2 Presentation

While the graph in fig. 2 is clear, it is tedious to read, especially while trying to line it up to music. To make this easier, I have made a GUI presentation via a matlab app to show what stops are on while the music is being played. A screenshot of the GUI at frame 66 of fig. 2 is shown in fig. 3.

This GUI is designed to be easily understandable by organists in order to aid in education. An important part of learning to play the pipe organ is learning how to select appropriate stops for a piece. This is learned primarily by listening to and examining what more experienced organists do. An algorithm to identify stops and an easily understandable presentation of the results would be a great aid in pipe organ pedagogy.

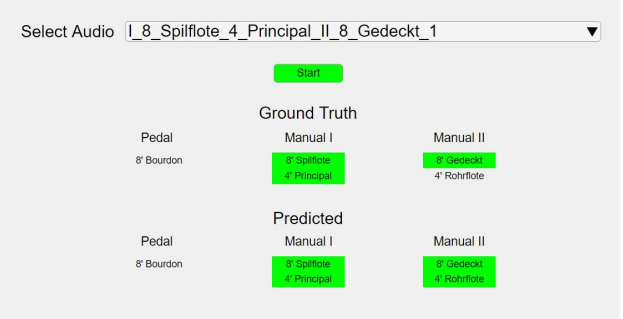

Figure 3. Display GUI at frame 65 of fig. 2

3.3 Future Work

I believe the accuracy of stop identification could be greatly improved by using a recurrent neural network (RNN) instead of the proposed NMF based method. A major identifying characteristic of a stop is the onset of each note. NMF, since it focuses on the sustain portion of notes instead of their onset, does not capture this well. A RNN, since it has memory, would do a better job of identifying this onset characteristic.

4. REFERENCES

[1] S. Ewert and M. Muller, "Using score-informed constraints for NMF-based source separation," in 2012, . DOI: 10.1109/ICASSP.2012.6287834.

[2] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in Proceedings of the Neural Information Processing Systems (NIPS), Denver, CO, USA, 2000, pp. 556–562.

[3] L.-F. Yu, L. Su, and Y.-H. Yang, "Sparse cepstral codes and power scale for instrument identification," in Proc. 2014 IEEE Int. Conf. Acoust., Speech Signal Process., 2014, pp. 7460–7464.

[4] Mathworks.com. (2018). Lasso or elastic net regularization for linear models - MATLAB lasso. [online] Available at: https://www.mathworks.com/help/stats/lasso.html [Accessed 4 Dec. 2018].