AutoScore: The Automated Music Transcriber
Project Proposal 18-551, Spring 2011
Group 1

Suyog Sonwalkar, Itthi Chatnuntawech
ssonwalk@andrew.cmu.edu, ichatnun@andrew.cmu.edu

May 1, 2011

Abstract

This project works on developing an automatic music transcription system for a single instrument throughout its entire chromatic range. In this project, we train a transcription system for a keyboard using a non-negative matrix factorization method as referenced in [3]. The preliminary testing was performed in MATLAB, then reimplemented on the TI TMS320C6713B Digital Signal Processor (DSP). The final implementation was done in real-time primarily on the DSP with a Graphical User Interface (GUI) on a Macintosh-based computer. The DSP and Mac were connected through a networking interface that transferred note data in real-time to the Mac.

Problem

Music transcription is the process of converting raw music signals into a musical score. Automated musical transcription can help musicians create sheet music as well as serve as an educational tool for amateurs. Manually transcribing music requires significant skill and time commitment from musicians. Currently, it is difficult for computers to transcribe music as well. This is due to the fact that modern music contains multiple instruments with multiple notes being played simultaneously (polyphony). Many methods have been developed to transcribe music from a single instrument, including bayesian-based methods [1] and even genetic algorithms [2]. Our project implements a recently proposed method that uses a non-negative matrix factorization technique to perform real-time music transcription.

Solution

Our solution uses a recently developed method for real-time music transcription of music as described in [3]. We use a CTK-591 Casio Keyboard to train and test the music transcription system. The system block diagram can be seen in Figure 1.

The system consists of multiple parts. First, the system was trained on musical note samples from the keyboard. This was performed "off-line", meaning it was completed before any testing was done and was not part of the real-time system. Training the system of the note templates consisted of obtaining the short-time Fourier Transform

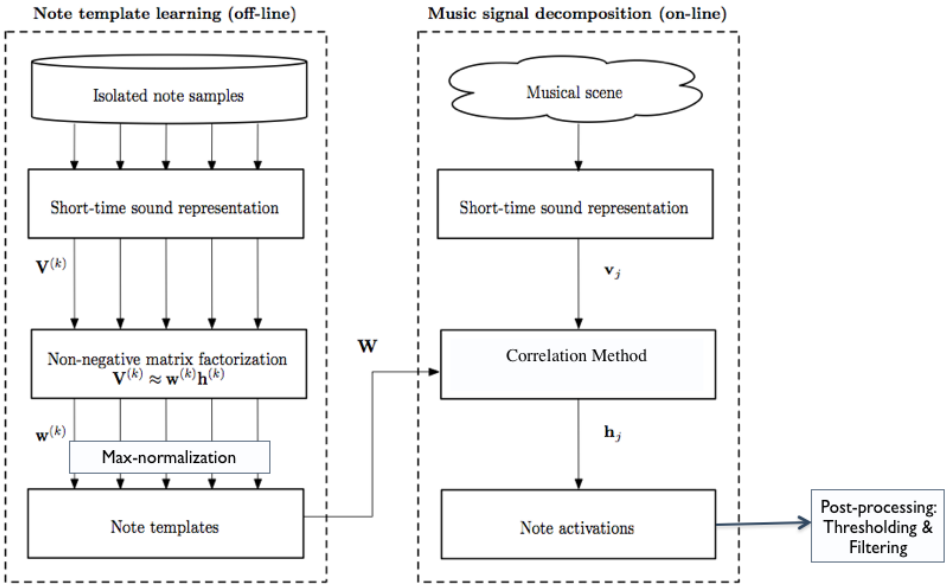

Figure 1: Block diagram of our music transcription system [3].

(STFT) of each of the musical note inputs, then performing Non-negative matrix factorization (NMF) on the spectrogram representation obtained from the STFT. The Non-negative matrix factorization produced note templates $w^{(k)}$ for each of the k music samples. This process was performed for each note on the keyboard and the resulting $w^{(k)}$'s were stacked into a matrix representation W. This completed the training phase of our musical transcription system. The training phase was implemented on a Mac in MATLAB in order to speed up the training process.

The testing phase of our system was performed in real-time with most of the work done on the TI TMS320C6713B Digital Signal Processor (DSP). The DSP obtained new musical input data at short time intervals and calculated the Fourier Transform (FT) of those signals. For the purposes of notation, we label the magnitude of the FTs of these signals v_j. For each v_j, the DSP performs correlation against the template matrix W to compute the musical note activations h_j using the template dictionary W that was trained in the previous step. This is represented by the following equation:

$$h_j \approx W v_j$$

These activations determine whether or not a specific note is being played.

Note that we perform pre-processing on the training notes in order to max-normalize the $w^{(k)}$ of each training sample. In addition, we perform filtering and thresholding on the activations h_j as part of post-processing the data.

In the next section, we cover the mathematical background of our system.

Background

Short-Time Fourier Transform (STFT)

The short-time Fourier Transform, or STFT, is a Fourier-related transform that is used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time [4].

In the discrete-time STFT, the signal is broken into chunks by a window function $w[n]$. Each chunk is then Fourier transformed. This can represented by the following equation:

$$STFT\{x[n]\} = X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n}$$

Here m represents the shift of the window in time, while ω represents the frequency.

The spectrogram is represented as the magnitude of the STFT [4].

$$Spectrogram\{x(t)\} = |X(m,\omega)|^2$$

An example of a Short Time Fourier Transform can be seen in Figure 2.

The window function used when computing the STFT was a hamming window, as defined by the following equation:

$$w[n] = 0.54 - 0.46 * cos(2\pi\frac{n}{N}), 0 \leq n \leq N$$

In our implementation, the window length was equal 4096.

An example of the hamming window function can be seen in Figure 3.

Non-negative Matrix Factorization (NMF) Non-negative Matrix Factorization (NMF) is a process that aims to factorize an n x m non-negative matrix V into an n x r non-negative matrix W and an r x m non-negative matrix H. Here r is a positive integer less than n and m [3,6]. r is called the rank of factorization. This will produce an approximation of V such that:

$$V \approx WH$$

The problem in solving for the NMF of a matrix is to find a goodness of fit measure called the cost-function. The standard cost function uses a Euclidean Distance measure. This makes the problem of solving for the NMF a minimization problem of the function:

$$\frac{1}{2}||V - WH||^2$$

The method used to solve for this equation has been extensively studied. To compute the W and H matrices, the iterative multiplicative updates algorithm, introduced in [5], is used. In [8], Lee and Seung provide proofs as to why the algorithm works. The updates for the Euclidean Distance metric are as follows:

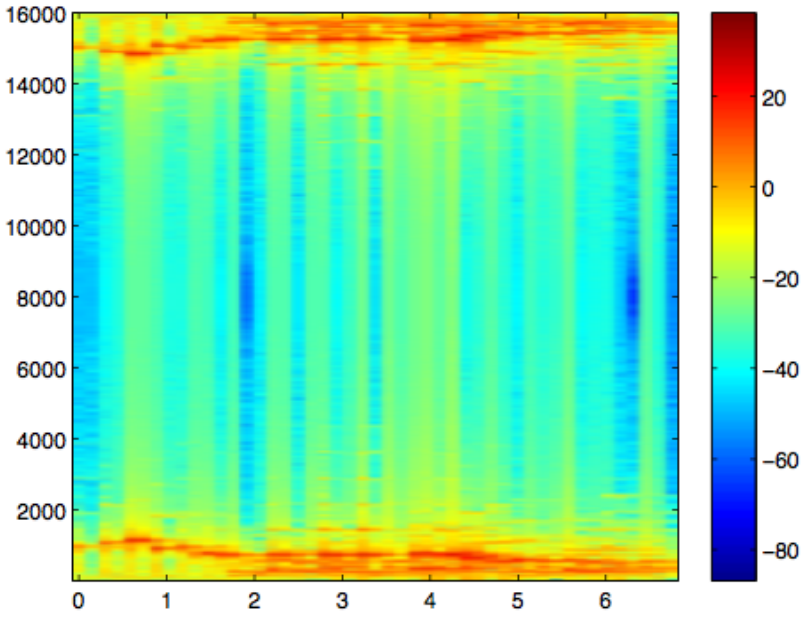

Figure 2: Short-Time Fourier Transform Example. The x-axis represents time domain (seconds), the y-axis represents frequency (Hertz).

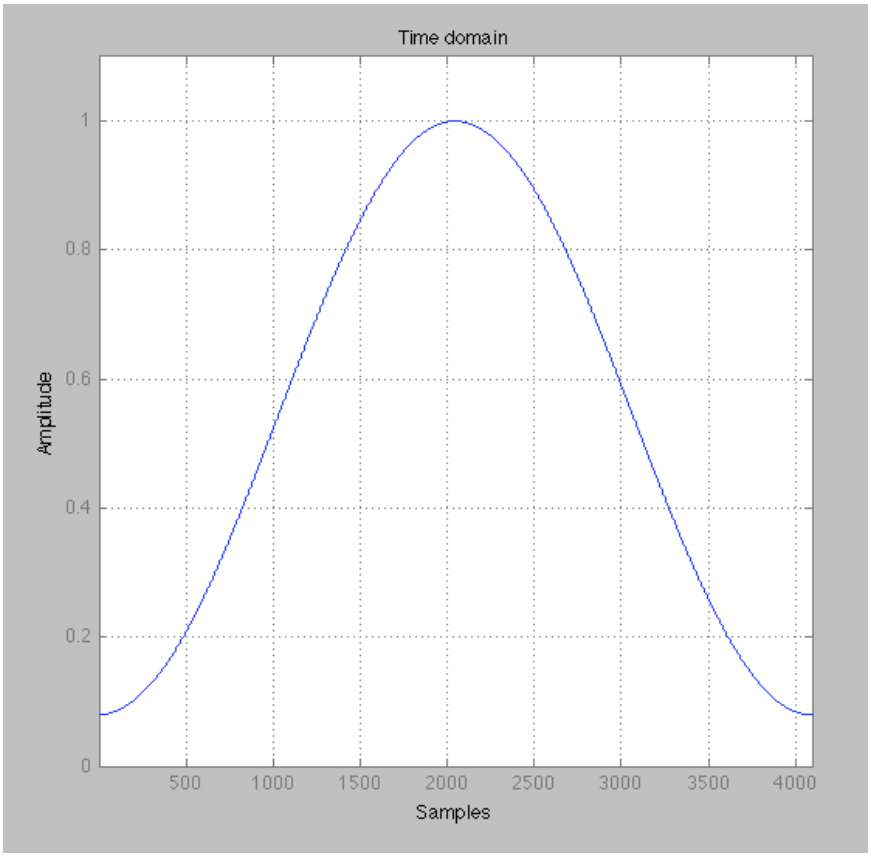

Figure 3: 4096-point Hamming Window

$$W \leftarrow W \otimes \frac{VH^T}{WHH^T}$$

$$H \leftarrow H \otimes \frac{W^TV}{W^TWH}$$

Where $\otimes$ is the element-wise multiplication of matrices and the division is the element-wise division of the matrices.

The rank used in our implementation is $r = 1$. This is due to the fact that we are using vectors for each training template. These vectors are later stacked into a matrix dictionary W.

Correlation Method

In the real-time testing phase, it is necessary to compute the correlation of the magnitude of the Fourier representation v_j. This method was chosen because it is extremely efficient and simple to implement on the DSP. The correlation method can be represented by the following equation:

$$h_j \approx Wv_j$$

Alternative methods to use for performing a similar computation would involve using a distance metic to determine the correspondence between the template vectors in W and the magnitude of the Fourier representation v_j. These methods will be described in the future work section.

What we implemented

Database

For our project, we created our own database of training samples from the CTK-591 Casio Keyboard. We created a database of 61 musical note samples from each of the keys of the Casio Keyboard. These were used in the training phase of our solution.

Testing

We test our data by performing error calculations on musical samples played on the Casio Keyboard. We compute an error between our transcriptions and the actual note played. The total error is a combination of the substitution error ϵ_{subs}, the missed error ϵ_{miss}, and the timing error ϵ_{time}. The substitution error is the error that occurs when the transcription classifies a note as another note, including octave errors. The missing error is the error that occurs when the transcription does not classify any note when a note is actually playing. The timing error is the error that occurs when the transcription does not identify small timing issues. An example of a timing error is when one note is played multiple times in a short interval, but is classified as only being played once.

Hardware

We used a Mac with MATLAB installed in order to train the musical template dictionary. In order to display the output data of our musical transcription system, we created a GUI on Mac OS X. The GUI is displayed in Figure 4.

We performed the real-time transcription calculations on the TI TMS320C6713B Digital Signal Processor (DSP). We communicated between the Mac and the DSP using a TCP sockets networking interface.

DSK Implementation

The implementation of the real-time algorithm on the DSP consisted of using a 44.1 KHz sampling rate on the line input from the keyboard. The Fourier Transform computation was performed at every 4096 samples (0.1 seconds). The magnitude of the FT was obtained, then correlated with the template matrix W.

The template matrix W was sent dynamically from the Mac after a network connection was established. The output from the correlation was returned to the Mac using the same network connection.

Real-time Speed Issues

The DSP code performed its calculations in real-time and sent data to the Mac periodically with little lag. The DSP code did have significantly more timing errors than the MATLAB tested code. These issues could be addressed in future work which involves performing the calculations on interleaved windows.

Demo

A live demo displaying our system transcribing notes from the keyboard was performed on April 26, 2011. An image of an example demo is provided in Figure 5.

The notes were input into the DSP from the line in, transcribed, then displayed on the Mac on a virtual keyboard. We also allowed others to test our system by playing their own notes. In addition, when the tone of the keyboard was changed, the algorithm still performed an acceptable transcription even though the system was not trained as such (an example of which is a trumpet tone).

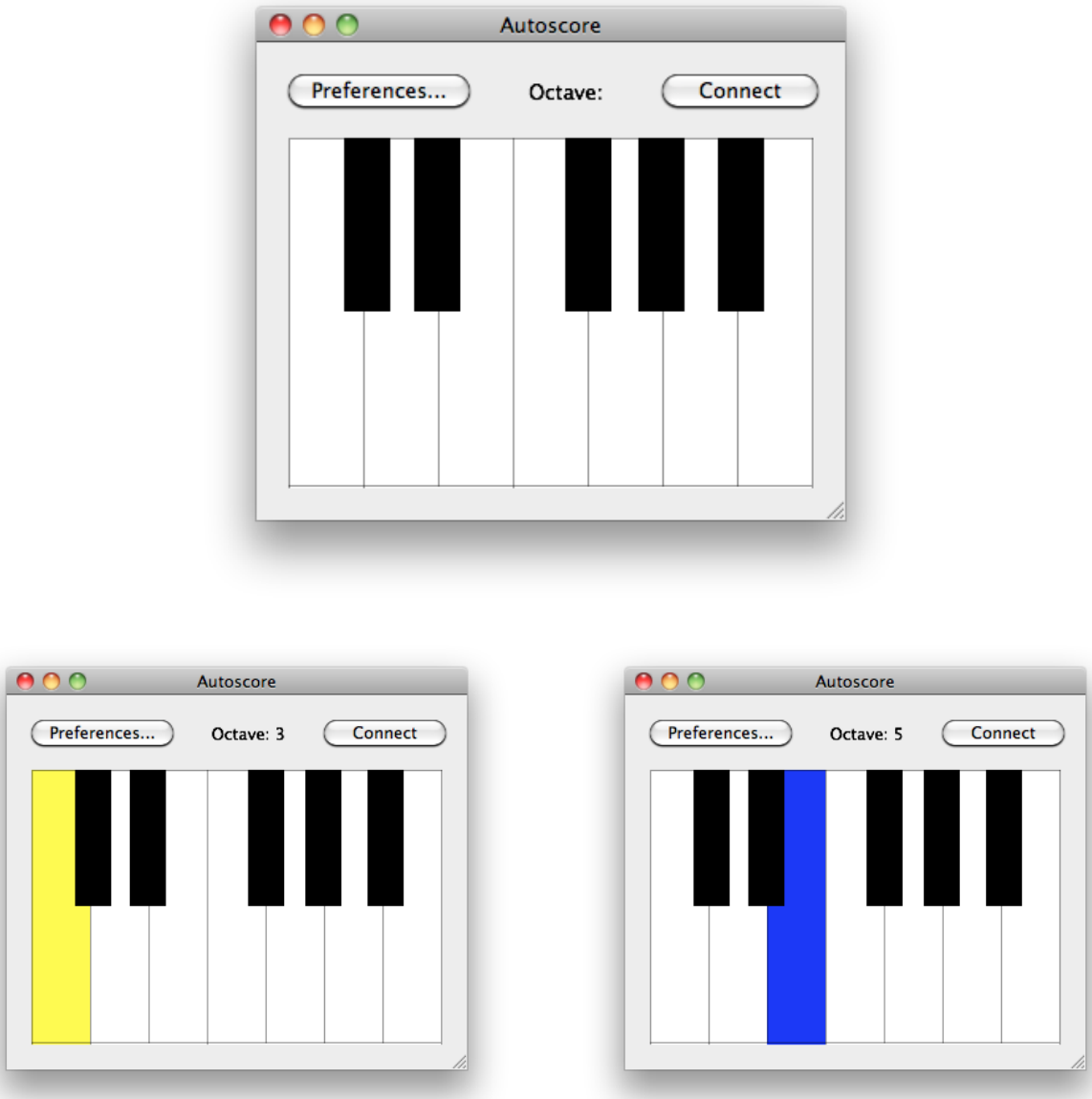

Figure 4: Mac OS X GUI (top), with notes playing (bottom)

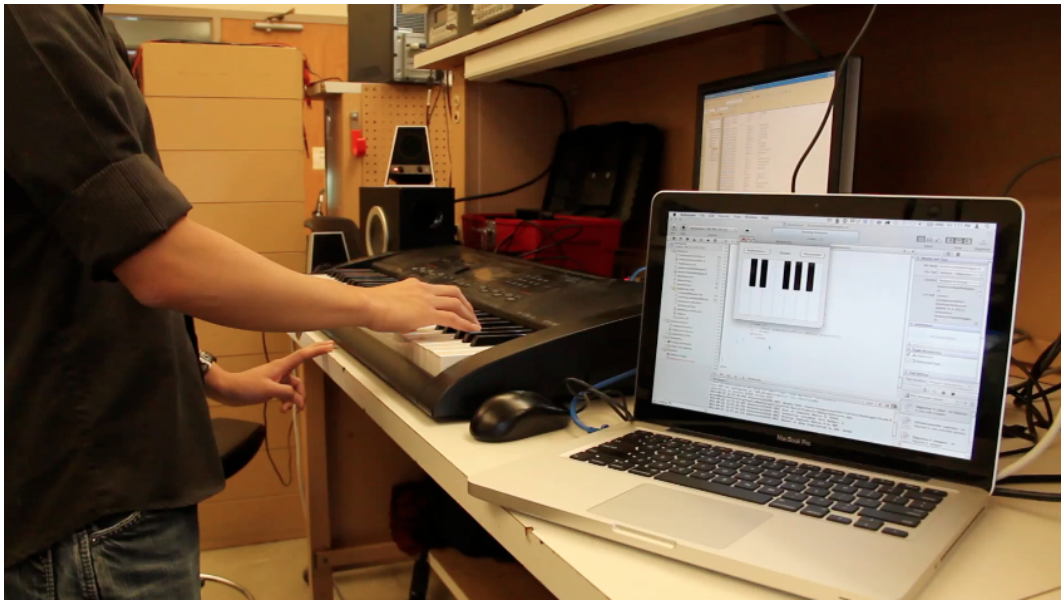

Figure 5: Example Demo, Keyboard (left), Mac GUI (right), DSK is in background

Results

Figure 6 displays the template dictionary matrix W that was trained on the individual notes of the keyboard. The training was performed in MATLAB on a Mac. The X-axis shows the (k)th note of the keyboard (out of 61) while the Y-axis displays the $w^{(k)}$ template vector for the corresponding note (k).

MATLAB Testing Results

We performed a test of the system in MATLAB over a sample song of Mary Had a Little Lamb in C major. The results are shown in Figure 7. The X-axis shows the (j)th time window while the Y-axis represents the 61 notes. The song was sampled at 44.1 kHz while the time window used was 4096 samples. The red bars in the figure represent notes that were activated at a given time frame. The sample song is provided in the given CD.

MATLAB Error Rates

The error rates calculated for the Mary Had a Little Lamb song are provided below. The explanations of each error rate was provided in the Testing Implementation section above.

Error	**Fraction**	**Percent**
Timing Error	2/25	0.08
Substitution Error	0/25	0.00
Missing Error	1/25	0.04
Total Error	3/25	0.12
Success Rate	22/25	0.88

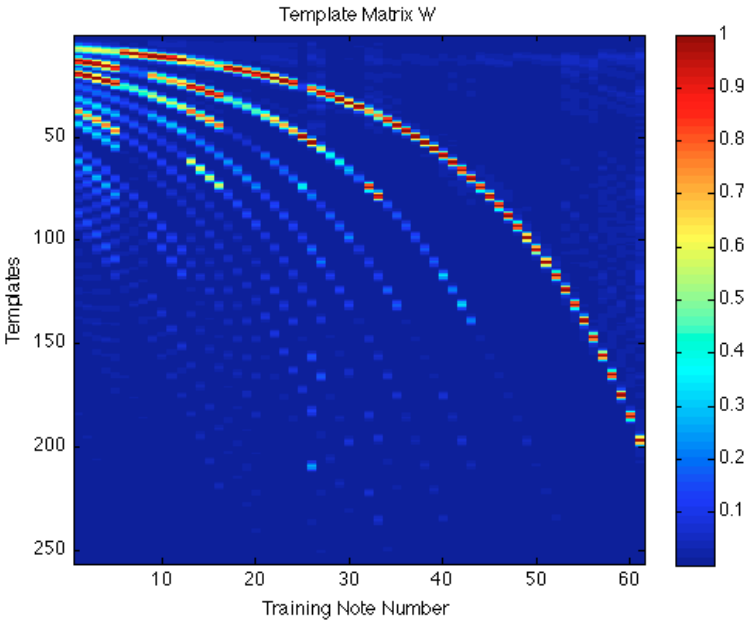

Figure 6: Template Dictionary Matrix W

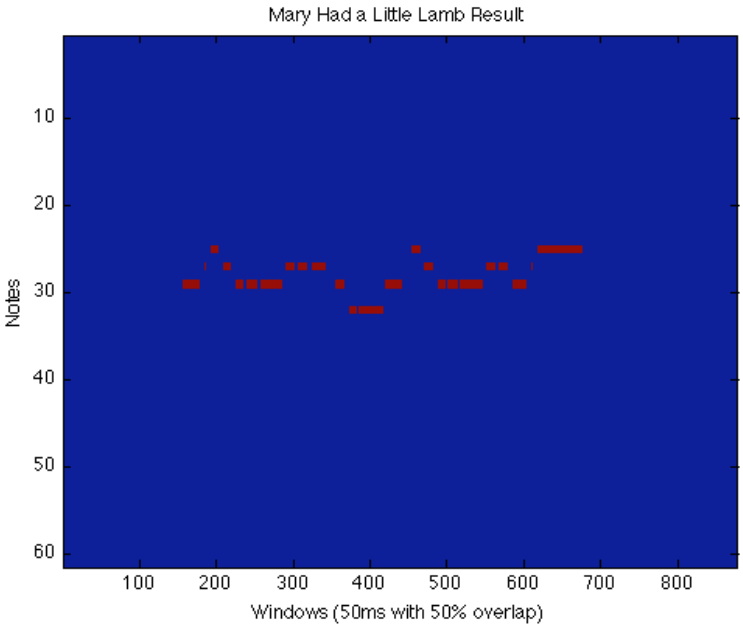

Figure 7: Template Dictionary Matrix W

DSP Error Rates

The DSP error rate was calculated for the entire chromatic scale of 61 notes. Note that timing errors were not included in this calculation, as there were significant timing errors with the DSP implementation. An improvement will be discussed in the future work section.

Error	Fraction	Percent
Substitution Error	8/61	0.13
Missing Error	8/61	0.13
Total Error	16/61	0.26
Success Rate	45/61	0.74

Timeline

Date	Tasks	Responsibility
Week 6-8 (2/14-3/6)	Obtained the training data set (Keyboard note samples) Started MATLAB training code (Compute STFT and NMF on training samples)	Suyog Itthi
Week 9 (3/7-3/13)	Finished up MATLAB training code Started implementing DSP code	Itthi Suyog
Week 10 (3/14-3/20)	Implemented Mac code and Networking	Suyog
Week 11 (3/21-3-27)	Finished up DSP code	Itthi and Suyog
Week 12 (3-28-4/3)	Combined the systems and finished coding (Combine Mac and DSP code)	Itthi and Suyog
Week 13 (4/4-4/10)	Tested on synthetic data	Itthi and Suyog
Week 14 (4/11-4/17)	Reimplemented MATLAB code Retrained Training Notes Finish up GUI	Itthi Itthi and Suyog Suyog
Week 15 (4/18-4/24)	Optimization of code and system Evaluation on test data Clean up and properly comment code	Suyog Itthi Itthi and Suyog

Previous Work in 18-551

Previous projects in the course have performed limited transcription, either in the case of not using stringed instruments (such as G8-S05') or only detecting single-tones in a limited octave range (such as G9-S00').

Novelty

Our project performed transcription on a keyboard in real-time with notes playing throughout its full range. In addition, we utilized the DSP to perform most of our real-time calculations. In comparison, the paper referenced in [3] implemented their real-time solution in MATLAB on a 2.4 GHz PC.

Discussion & Future Work

Improvement of Accuracies

For future work, we would like to improve our accuracies on the lower octaves by spreading out the template matrix. This can be achieved by downsampling, which spreads out the frequency content in the Fourier domain. In addition, we can improve the resolution of the template vectors $w^{(k)}$ after downsampling by zero-padding in time [7].

Alternative to Correlation & Polyphonic Music

In addition, we can modify our algorithm by using a non-negative matrix decomposition method to determine the h_k vectors [3] (rather than the correlation method currently used).

This is done in [3] by using an idea similar to NMF to solve for the activations h_j, given a fixed W. This can be represented by the following equation:

$$v_j \approx W h_j$$

In [3], the Beta-Divergence Distance Metric is used as a cost function to solve for h_j, which is defined as follows:

$$d_\beta(x|y) = \frac{1}{\beta(\beta-1)}(x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1})$$

This distance metric, used in [3] for computing the activations h_j, produces the following update equation:

$$h \leftarrow h \otimes \frac{(W \otimes (ve^T))^T (Wh)^{\cdot\beta-2}}{W^T (Wh)^{\cdot\beta-1}}$$

Where e is defined as a vector of ones and the powers are element-wise powers.

We can use this new h_j vector to potentially improve our accuracies while testing. In addition, [3] mentions that this new h_j vector can work with polyphonic music as well. This could allow our implementation to work with multiple notes at the same time.

Improve DSP Implementation

Our DSP implementation does not currently use interleaved or hamming windowed functions (as our MATLAB implementation does). We can potentially improve the accuracies of our DSK implementation by using 50% overlapping hamming windows for time window of the input signal.

References

[1] - Peeling, Paul H., "Probabilistic Modelling and Bayesian Inference Techniques for Music Transcription," University of Cambridge, 2007.

[2] - Reis, G.; Fonseca, N.; Ferndandez, F.; , "Genetic Algorithm Approach to Polyphonic Music Transcription," Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on , vol., no., pp.1-6, 3-5 Oct. 2007

[3] - Dessein, A.; Cont, A.; Lemaitre, G; , "Real-time Polyphonic Music Transcription with Non-Negative Matrix Factorization and Beta-Divergence," International Society for Music Information Retrieval Conference, 2010.

[4] - "Short-time Fourier Transform." Wikipedia, the Free Encyclopedia. Web. 12 Feb. 2011. http://en.wikipedia.org/wiki/Short_Time_Fourier_Transform.

[5] - Lee, D.; Seung S., "Learning the parts of objects by non-negative matrix factorization," Nature, 1999.

[6] - Berry, M.; Browne, M.; Langville, A.; Pauca, V.; Plemmons, R.; "Algorithms and Applications for Approximate Nonnegative Matrix Factorization," Elsevier Preprint, 2006.

[7] - Oppenheim, A.V., Schafer, R.W., Yoder, M.T., and Padgett W.T., "Discrete-Time Signal Processing", Prentice Hall, 2009.

[8] - Lee, D.; Seung, S.; Algorithms for non-negative matrix factorization, Advances in Neural Information Processing Systems, April 2001. 556-562.