Automatic LP Digitalization
18-551 Spring 2011

Group 6: Michael Sibley, Alexander Su, Daphne Tsatsoulis
{msibley, ahs1, ptsatsou}@andrew.cmu.edu

Introduction

This project was originated from our interest in doing something with music and finding a way to incorporate digital signal processing to better music. We talked to several professor and parents and students and found that a common problem that could be fixed with digital signal processing is the tedious process of converting a long-playing record into an mp3 file for your computer to utilize. We attacked the problem by analyzing the needs, by asking those with long-playing records what the main issues were with converting their records. The main issues that were brought up was the recording of the entire album taking a lot of time, then having to split the songs. Also when a record is played and recorded by a turntable, there will undoubtedly be artifacts of a dirty record or just playback on a turntable. These annoying pops and clicks need to be subdued and removed. The need to listen to your recording and attempt to identify where distortion of the record occurred was a big time concern for people. The need to sit there and manually remove pops and clicks and other noises from the tracks made it undesirable to devote hours at a time on a Saturday afternoon to convert a long-playing record into an mp3 file. We set out to tackle these problems in a simple efficient solution by automating the entire process. That way, one could just leave the program and record playing overnight and be presented with a completely digitalized long-playing record in the morning.

Background

We looked through previous 18-551 projects to gain some insight as to how we should go about tackling this problem. After searching through the semesters and years of projects, we didn't find any project that attempted to automate a recording, cleaning, and filtering process. We then moved to search the Internet for different software and programs that may already exist that could give us some guidance.

After searching for research projects online that may solve our problem or solve parts of our problem, we concluded that there has not been a process designed to serve as a solution to our problem. We found some software and programs that were sold, that could do the recording of an album and split the album into separate tracks automatically. However, we did not find anything that would automatically master and edit your split tracks. All the programs with noise removal and pop and click removal required human judgment to determine whether or not something needed to be filtered. We checked out the algorithms and methods of these editing programs to try to find a way to reproduce it and automate it without the human sense of judgment.

Our System

We decided our first order of business was to get our program to record the turntable's output and store into a wav file. We proposed that we needed to record it as a wav file because wav files maintain a higher quality of audio. The more data we have in terms of music, the more we can edit it without losing too much data to reconstruct the music. We then wanted to parse through our entire album so that we could methodically find the long pauses in between tracks and separate them into individual songs on the album. After separating the album into individual tracks, we would take the songs one by one and automatically remove pops and clicks and also remove the background noise of a noisy playback. We figured editing the album would be best after separating the tracks because each track would have a different average level of amplitude, making thresholds for amplitude of noise differ from track to track. After the automatic editing, we would then have the edited sound file exported in whatever form the user designates. We wanted to have the entire process completely automated, so in addition to recording, splitting, and editing the tracks, we wanted to label the tracks as well. To employ this we wanted to scan or take a picture of the LP record's case and cover and use Optical Character Recognition (OCR) to determine what the letters on the case were. We would then take this information and tag it onto the exported music file and we'd have a track from a fully digitalized and edited album.

Pop and Click Removal
Due to the nature of vinyl records being an analog medium of recording audio, it is susceptible to inaccuracies in playback. If the record gets any bits of dirt on the surface, this can cause pops and clicks in the output. If the record had been damaged by scratching at some point then clicks could be caused on the album playback. We want to minimize these clicks in the final recording, so we can get as close as possible to studio quality. This can be achieved to some extent by analog means such as cleaning the record but can also be achieved using digital signal processing.

Our Solution
In order to remove any extraneous spikes from the original input, we implemented the Phase-Space Thresholding Method introduced by Goring and Nikora in [1]. In this method the original time series elements and their derivatives are plotted against each other in phase-space and then enclosed by an ellipsoid. Any points lying outside the ellipsoid are considered extraneous and are removed via interpolation. The process is then reiterated until the number of outliers does not change. The method is detailed below:

1. Surrogate first and second derivatives of time series are taken.
$$\Delta u_i = \frac{(u_{i+1} - u_{i-1})}{2}$$
$$\Delta^2 u_i = \frac{(\Delta u_{i+1} - \Delta u_{i-1})}{2}$$
2. The standard deviations $\sigma_u, \sigma_{\Delta u}, \sigma_{\Delta^2 u}$ are calculated along with the expected absolute maximum λ.
$$\lambda = \sqrt{2\ln(n)}$$
3. The rotation angle of the principal axis of $\Delta^2 u_i$ versus u_i using the cross correlation is calculated.
$$\theta = \tan^{-1}\left(\frac{\sum u_i \Delta^2 u_i}{\sum u_i^2}\right)$$
4. Calculate the ellipse with the maxima and minima found in step 3. The minor and major axes will be the product of λ and the standard deviation of the variables being plotted. For example, when plotting Δu_i versus u_i the major axis is $\lambda\sigma_u$ and the minor axis is $\lambda\sigma_{\Delta u}$.
5. For each projection, any points outside of the ellipse are replaced. We replace the outlying point by interpolation. We interpolate using a cubic polynomial as implemented by MATLAB.

We used a preexisting MATLAB function to implement the algorithm called Despiking created by Nobuhito Mori [3]. His implementation of the Phase-Space Thresholding Method is the same as the one described above.

Results

The resulting signal contained less spikes and those that remained were noticeably dampened. Most importantly, the clarity and detail of the music was not distorted by this processing. Figure 1 shows a signal before and after the pop-removal.

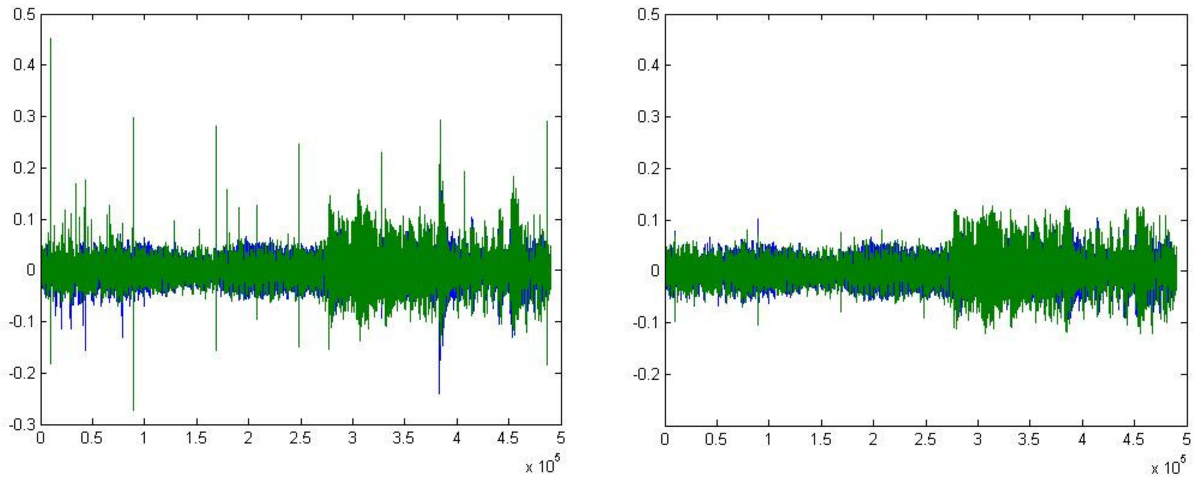


Figure 1: (left) Original Input, (right) After pop-reduction

This process did not work as effectively when pops and clicks were detected in the noisy segments between songs as can be seen in Figure 2. Though this does not affect the quality of the music, it can be annoying between songs.

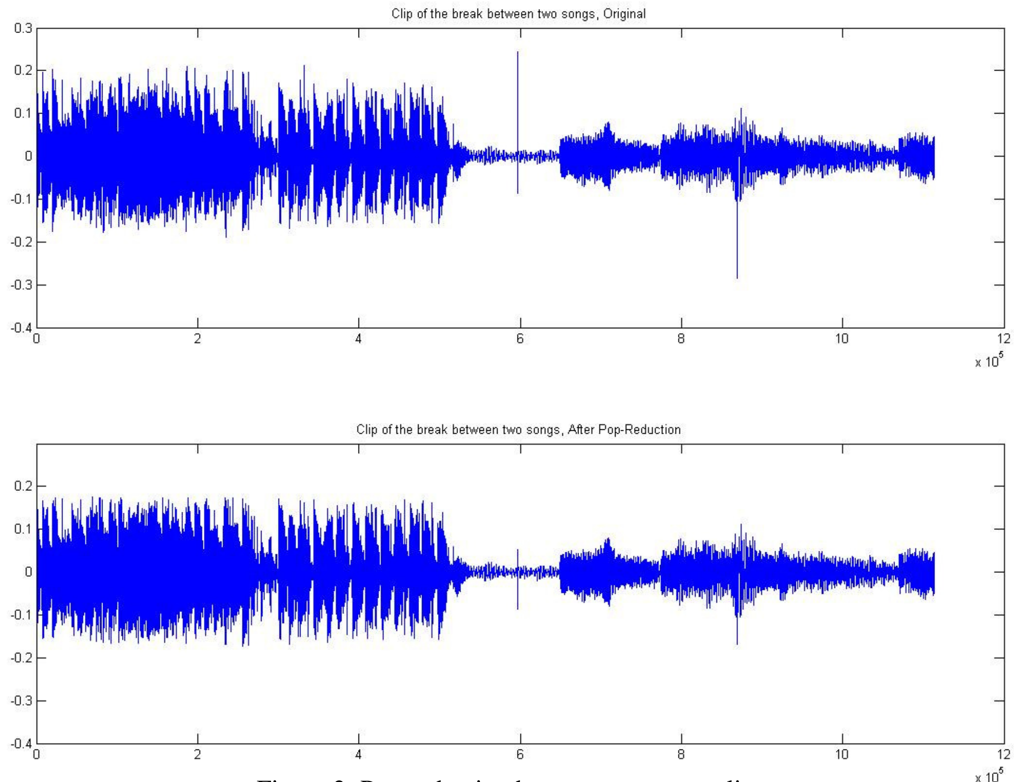


Figure 2: Pop reduction between two song clips

Noise Removal

Records are an analog method of recording audio and are therefore susceptible to noise. The audio on a record is encoded in the surface topology of the record. If there is any dust on the surface of the record, it will cause noise on the output of the record player. Digital to analog conversion is also susceptible to noise from electrical sources.

Our Solution

The algorithm we used to remove noise was a median filter. This is an algorithm commonly used in image processing to remove salt-and-pepper noise. Salt-and-pepper noise is random white and black pixels that occur over the image. A typical median filter involves iterating through the data, taking the median of each point, u_k, and the points to each side, u_{k-1} and u_{k+1}, and setting u_k to the value of the median. This type of filter had an effect on the noise, but did not seem to remove it completely. To increase the effectiveness, we increased the number of points from 3 to 9. The filtered time series equals:

$$u_k = median(u_j : j \in [k-4, k+4])$$

By setting points to the median of the surrounding points, we smooth the signal and remove much of the noise.

We tried several other algorithms before we settled on the median filter. Many audio editing software programs remove noise, by taking a section of noise and reducing those frequencies proportionally in the music. We originally planned on using a similar method, however it is difficult to automatically detect the difference between noise and music. We also attempted using wavelets and a basic low pass filter. Both of these attempts dampened the music too much, and damaged the end result.

Results

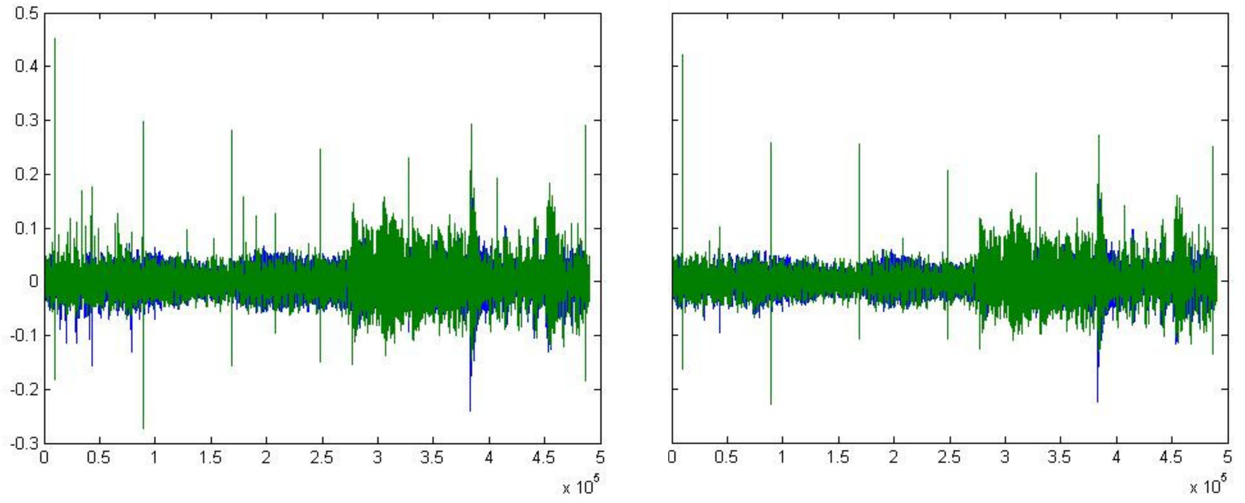


Figure 3: (left) Original input, (right) After noise-removal

The median filter was very effective for removing noise from the songs without dampening or damaging them. Figure 3 shows some of the pre- and post-processed music. Figure 4 shows music that has undergone both pop-click removal and noise removal.

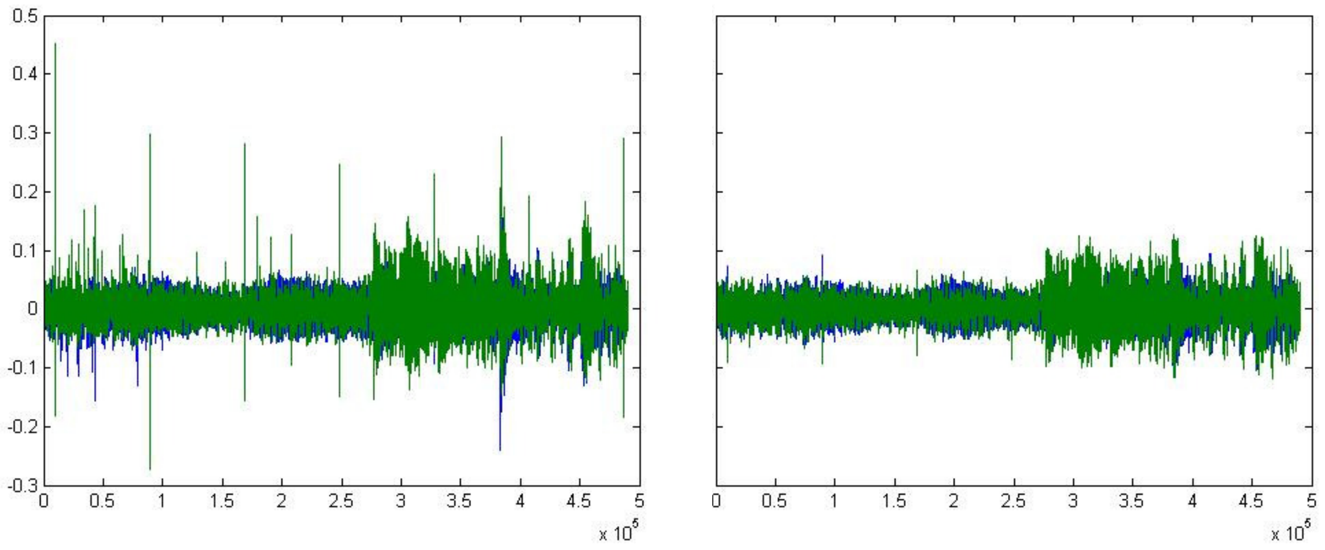


Figure 4: (left) Original input, (right) After pop and noise removal

Optical Character Recognition

In this part of the project it was our intention to show that it would be possible to take an image of the backside of a record, and convert the text on the image into labels for the record's songs without user input. Due to the scope of our project, we limited the input images to simply colored images such as the ones in Figure 5.

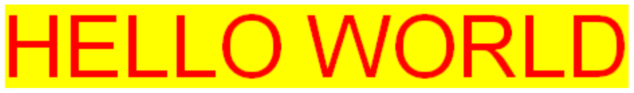

HELLO WORLD

Figure 5: Sample OCR input

Preprocessing

The input to the system is a .bmp file containing an image of song titles separated by line in Arial font of size 48. We converted all input images to gray scale and separated the song titles by detecting horizontal lines of only one color. Once the song titles had been separated from each other, one title was processed at a time. We detected individual letters in the image by detecting their edge and clipping out the box surrounding this edge as shown in Figure 6. This method was not always successful. For example, when the letters 'EX' were located next to each other, they were detected as one letter, not two. This could be avoided by feeding the entire song title into the MACE filter (described below).

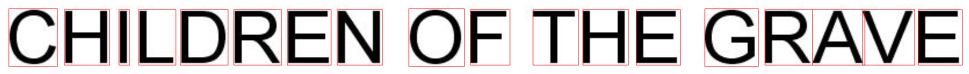

CHILDREN OF THE GRAVE

Figure 6: Letter detection

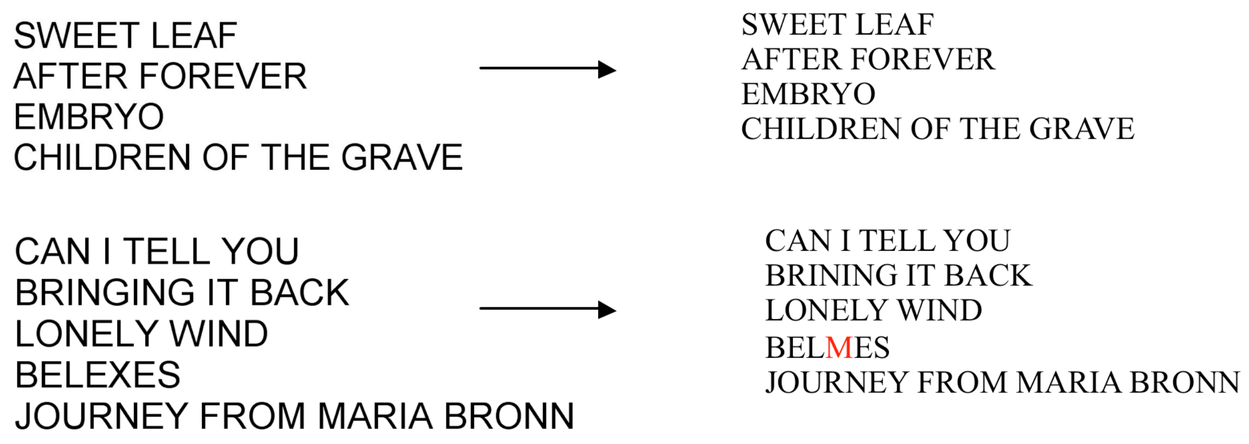
Classification
In order to classify detected letters we used minimum average correlation energy (MACE) filters [2]. A resulting MACE filter takes the form

$$\mathbf{h} = \mathbf{D}^{-1}X(\mathbf{X}^{T}\mathbf{D}^{-1}X)^{-1}\mathbf{c}$$

where X is the training input, the matrix $\mathbf{D}$contains the average power spectrum of the training images along its diagonal, and $\mathbf{c}$ is a unit vector of length equal to the number of training images.

We trained on three sets of individual characters in different fonts. We then tested using random phrases and song titles.

Results
The MACE filter was very successful in correctly identifying the input. Issues arose when characters were not correctly separated, but this could be avoided by simply feeding the entire song title into the filter rather than parsing out each character. There was some misclassification, though it did not occur often and it was not consistent. Errors were highly dependent upon the letters surrounding the misclassified letter, because they were often separated from each other badly. The most highly pairs of confused letters were ‘L’ and ‘E’ and ‘I’ and ‘T’. Figure 7 displays some of the input and output pairs.



SWEET LEAF
AFTER FOREVER
EMBRYO
CHILDREN OF THE GRAVE

→

SWEET LEAF
AFTER FOREVER
EMBRYO
CHILDREN OF THE GRAVE

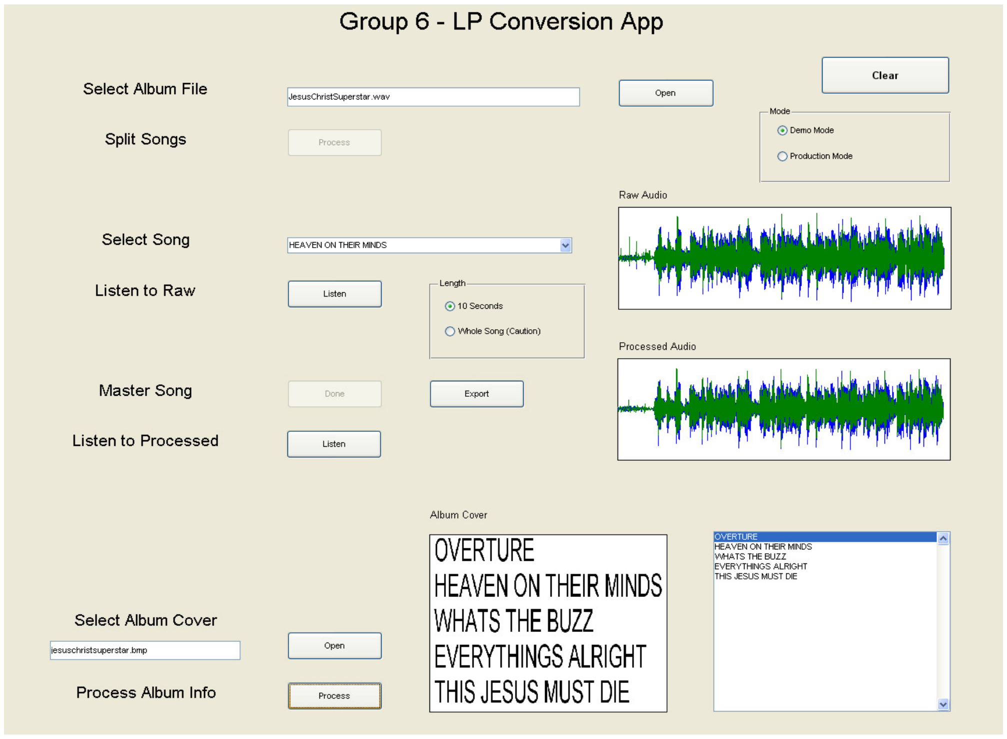
CAN I TELL YOU
BRINGING IT BACK
LONELY WIND
BELEXES
JOURNEY FROM MARIA BRONN

→

CAN I TELL YOU
BRINING IT BACK
LONELY WIND
BELMES
JOURNEY FROM MARIA BRONN

Figure 7: Final input and output from OCR

Demo



JesusChristSuperstar.wav

Mode: [✓ Demo Mode] Production Mode

HEAVEN ON THEIR MINDS

Length: [✓ 10 Seconds] Whole Song (Caution)

jesuschristsuperstar.bmp

OVERTURE
HEAVEN ON THEIR MINDS
WHATS THE BUZZ
EVERYTHINGS ALRIGHT
THIS JESUS MUST DIE

OVERTURE
HEAVEN ON THEIR MINDS
WHATS THE BUZZ
EVERYTHINGS ALRIGHT
THIS JESUS MUST DIE

Figure 8: Screen shot of the GUI in use

Since the stimulus of our project was to create an automatic system to convert the vinyl records into digital form, the ultimate goal would be software with which one would simply put the record on the player, press go and start playing the record. The user would also have to input a picture of the song information on the album cover. In the case of the demo however, we wanted to demonstrate each step in the process, so the demo was divided up step by step. We wrote separated the audio processes and the image processes, so that they could be run in either order.

On the audio side of things, the user selects a wav file that is the recording of the entire album and runs the song splitting process. This enables a dropdown list of the individual tracks on the album. The user can then select each track and process the audio to remove pops and clicks. The GUI provides an option to listen to the track before and after processing as well as provides a graph of the waveform. It is easy to identify clicks and pops visually on a waveform, so we provided this as another way for the user to view the results of our algorithms.

On the image side of things, the user selects a bitmap file that is a picture of the track information on the album. The GUI displays this image so the user can verify that they have the correct image. The user can then process the image to run the OCR and obtain the track titles. This is then output as a list and also integrated into the dropdown list of songs, if the user has already done the audio processing.

Finally, the software needs to provide the user with a song file that they can add to their music collection. The GUI provides an export file that renames the track file to the name obtained by the OCR. This is then copied into the output folder and the folder is opened for the user.

Due to the actual processing of songs taking significant time, we also implemented a demo mode and a production mode. In production mode, the software does each step real-time and actually processes the audio as input. In demo mode however, the software uses preprocessed audio files so that we could demonstrate the results in a reasonable amount of time.

Future Work

Our song splitting was not of the highest quality, because the algorithm we used required a lot of memory to process an album. At the moment, we cannot find a computer with enough random access memory to actually process our song splitting algorithm on an entire full-length album. In future works, a full album is just too much information to process at a time, so our suggestion is to take the high quality wave file, and reduce its sampling rate. This will reduce the number of points dramatically making it much less dense and requiring less memory to process. A break in between songs that happens at 4:39 in the wave file will also happen at 4:39 in a lower sampling rate file as well. We analyze the low sampling rate file to find out at what time the break in between the tracks is, and save that information. We then return to the high quality album, and cut the tracks at the desired spots obtained from this process.

Our algorithm at the moment runs on MATLAB. In the future we'd like to be able to make it run on the DSK. A problem with the DSK and this complex problem is that there is not enough on chip memory to process an entire song. So for future work to implement more processing on the DSK, we'd need to write an algorithm that does the processing in small chunks, so that we can transfer a small chunk to the DSK, process it, and grab it, and start on another small chunk. If coded well enough, it will actually be more robust than the current code we have now.

In terms of the GUI, there are additional developments that should be done to make it into a finished product. The recording of the album into the computer currently needs to be done with Audacity software, independent of the GUI. In the future, this functionality could be integrated into the GUI so that tit could be done automatically. For the final product, the GUI should be simplified. The user does not need to see each step of the process, but would want to see some sort of status bar to indicate how far along the process is. Also, the GUI would need a more robust file handling. Currently, it only works if all of the files are placed in the same folder as the code. In a production version, it should be able to handle files in any location on the computer.

References

[1] D. G. Goring and V. I. Nikora, "Despiking Acoustic Doppler Velocimeter Data", *Journal of Hydraulic Engineering*, ASCE, Vol. 128, No. 1, pp. 117-126. Discussion: Vol. 129, No. 6, pp. 484-489, (2002).

[2] A. Mahalanobis, B.V.K. Vijaya Kumar, and D. Casasent, "Minimum average correlation energy filters," *Appl. Opt*. 26, pp. 3633-3630 (1987).

[3] N. Mori, 'Despiking', <http://www.mathworks.com/matlabcentral/fileexchange/15361-despiking>, 2007.

Tasks

Michael Sibley
- Song Splitting
- Moving Average Implementation
- Audacity pop removal implementation
- GUI

Alexander Su
- Audacity pop removal algorithm
- Median filter algorithm
- Testing

Daphne Tsatsoulis
- Song Splitting
- Wavelet algorithm
- Median filter algorithm and implementation
- Phase-Space pop removal algorithm
- OCR algorithms and implementation