Technical Issues in Digitization of Large Online Collections of Phonograph Records

Beinan Li, Catherine Lai, and Ichiro Fujinaga; Music Technology Area, Schulich School of Music, McGill University; Montreal, Quebec/Canada

Abstract

This paper discusses the technical issues involved in digitization of large online collections of LPs and presents the techniques to perform automatic track separation of unsegmented audio streams, the approaches taken to automate metadata extraction from scanned record labels, and the tools used to automatically create derivatives of audio and image files in various optimized formats.

Introduction

Long-playing phonograph records (LPs) were one of the major analogue recording formats distributed commercially throughout most of the twentieth century. Although most of these historic sound recordings have long shelf lives, compelling reasons have led to a shift toward digital preservation. Digitization is time consuming and expensive. Many steps involved in the digital conversion, such as track separation and metadata extraction, require much human intervention. Due to the enormous quantity of existing recordings and the time required to digitize them properly, tools to automate the digitization process so as to reduce the cost of digitizing very large numbers of LPs are highly desirable.

This paper discusses the technical issues involved in digitization of large online collections of LPs. The techniques to perform automatic track separation of unsegmented audio streams, the approaches taken to automate metadata extraction from scanned images of record labels, and the tools used to automate the creation of derivative audio and image files in various optimized formats are subsequently discussed.

Automatic Track Segmentation

The analog-to-digital conversion of an LP results in a continuous audio stream containing unseparated musical pieces. For the distribution of music, however, it is convenient to have separated musical tracks. Segmenting tracks manually is costly and can be avoided by using an automatic track segmentation system. Although there are software tools such as Gramofile [1], LP Ripper [2], and Rip Vinyl [3] that perform track segmentation, they are primarily designed for popular music. They rely on simple heuristics that are inadequate to distinguish between soft musical passages and noisy inter-track sections typical of many old classical recordings.

Related works

Automatic audio segmentation and annotation have gained much attention from researchers in various multimedia-related domains including music information retrieval. Tzanetakis [4] provided an overview of audio segmentation in 2002 while Zhang and Kuo [5] used audio portion of a video signal to facilitate video classification and retrieval.

The most popular approach to automatic audio segmentation is to use machine learning methods and short-time audio feature extraction. A number of techniques have been reported. Support Vector Machine was used for classification of silence, speech, pure speech, music, and environmental sounds [6] as well as segmentation of concert recordings [7]. Segmentation between speakers in multi-speaker environments was achieved using optimal state sequence changes between speakers by Hidden Markov Model [8]. Moreover, the k-nearest neighbor and quadratic Gaussian classifier were used to discriminate between speech and music [9].

ACE-based machine learning approach

The main goal of track segmentation is to locate the boundaries between a music track and its adjacent passages of inter-track silences and then separate them accordingly. To locate the boundaries, we treat the track segmentation task as a binary classification problem. Two types of classes are used for classification: music and inter-track silence. We use jAudio's audio feature extraction library [10] and the ACE framework [11] to perform supervised learning.

jAudio is an open-source audio feature extraction tool written in Java. It provides a convenient way of selecting various audio features and configuring analytical parameters such as sampling rate and window length for classifications. Currently 27 distinct features are implemented.

ACE is an open-source framework that provides a convenient way to experiment with various combinations of audio features and classification methodologies. Given a set of feature vectors, ACE experiments with a variety of classifiers, classifier parameters, classifier ensembles, and dimensionality reduction techniques in order to arrive at a good configuration for the problem at hand. The standardized XML format is used as a communication protocol for jAudio and ACE. For example, ACE reads the feature definition and feature values from the XML files generated by jAudio's feature extraction process and outputs the classification result in another XML file.

Our track segmentation scheme contains the following steps:

1) Feature extraction using jAudio.
2) Training the classification model with ACE.
3) Classification on the test dataset.
4) Post-processing.

Feature extraction

Feature extraction is performed for both the training and test datasets. An input audio file with unseparated musical tracks is first downsampled to 16kHz to reduce signal variability. A sliding-window based feature extraction is then performed using 1-second windows with no overlap. Although many different combinations

of features were tested, it turned out that the root mean square (RMS) alone, a feature also adopted in Gramofile, performed as well as any other features. The feature definition and extracted feature values including the feature types and the analytical parameters are stored in XML files for the following steps.

Training the classification model

During the training phase, an *N*-1 (*N* is the total number of training audio files) fold cross-validation is performed with ACE on the features of the training dataset to minimize overfitting. Ground-truth data is gathered by manually labeling the training data using a convenient graphical audio editor developed by the authors as an extension to the open-source software Audacity [12]. ACE is then used to compare various classification schemes and determine the most suitable approach. In our experiment, the selected optimized classifier was the C4.5 Decision Trees with bagging. The classification model is stored for the test phase.

Post-processing

The classification result from the test phase is an array of class labels, *m* for music and *s* for silence, assigned to each of the 1-second window in the audio stream. We often found that a continuous music track or an inter-track silence passage is contaminated with interleaving *m*'s and *s*'s due to sporadic misclassifications. A rule-based post-processing was therefore designed to correct these errors.

Previous methods, such as in [13], use a fixed-window-length sliding-window analysis to perform a smoothing process that re-labels segments according to the majority category in a window. This smoothing process often fails when only a portion of an interleaving structure is captured by the window.

Our strategies involve variable-sized window for smoothing and the use of high-level heuristics. For the smoothing, we introduce the concept of a "chunk," which is a group of consecutive windows that share the same label. The music and inter-track silence chunks are labeled M and S, respectively; thus, for every M there will be two adjacent S's and vise versa. We then assign a level of confidence to each chunk. Chunks that were classified with a high confidence level are taken as boundaries and a smoothing function is applied to the chunks enclosed by the boundaries.

Our heuristics include:

1) The chunks at the beginning and end of an album are usually S.
2) If an M is overly short, the classification of either the chunk itself or an adjacent chunk labeled S is likely to be wrong.
3) If a chunk longer than 3 seconds is labeled as S or a chunk longer than 10 seconds is labeled as M, the classification is likely to be correct. Also, S chunks should be much shorter than M chunks in most music albums.

Based on the heuristics 1) and 2), a special filtering is applied to correct any mislabeled chunks at the very beginning and end of audio streams. By using 2) and 3) and by comparing the relative size of neighboring chunks, incorrectly assigned small chunks are relabeled by smoothing until correct total number of tracks are obtained.

Evaluation method

The error measurement used for the regular window-wise classification, i.e., the percentage of misclassified segments, is not suitable for the track segmentation task evaluation. For example, if a 1-second *m* in the middle of a music track is mislabeled as *s*, then the resulting number of tracks will be wrong even though the classification accuracy would be greater than 99%. Therefore, a special error measurement was developed. The classification errors are categorized into three types in terms of their influences on the track segmentation results. Given a ground-truth data segment "*mmsssmm,*"

1) Error Type I is defined as any mislabels that split a complete music track into segmented pieces that lead to a wrong number of tracks. This error is computed by:

$$e_1 = \left|TF - TT\right| / (TF + TT) \quad (1)$$

where TF denotes the number of tracks found and TT denotes the correct number of tracks. An example of this type of error is "*mmsmsmm.*"

2) Error Type II is defined as any mislabels that mistakenly mark *m* for *s* at transitions. This has a lesser negative impact on the segmentation result than Type I. However, this could still affect the position at which the actual track segmentation is executed. This error is computed by the ratio of the number of mislabeled windows to the total number of windows. An example of this type of error is "*mssssm.*"
3) Error Type III is any mislabels that mistakenly mark *s* for *m* at transitions. This error has the least impact on the result. The error is computed by the ratio of the number of mislabeled windows to the total number of windows. An example of this type of error is "*mmmsmmm.*"

Finally, the error function of the track segmentation algorithm is the weighted sum of all three types of errors in the output:

$$e = w_1 e_1 + w_2 e_2 + w_3 e_3 ,$$
$$w_i > w_{i+1}, w_i \in (0,1), \sum_i w_i = 1 \quad (2)$$

where w_i is the assigned weight for the Error Type *i*.

Experimental results

Our approach to automatic track segmentation was evaluated on digitized recordings of the David Edelberg's Handel LP collection [14]. Five digitized 20-minute classical LP albums were used as training dataset and six other albums from the same collection were used as the test dataset. The music passages included songs with piano, choir, solo instrumental, and orchestral music. The inter-track silence passages of these recordings are usually noisy.

For our dataset, the optimized classification scheme selected by ACE among a total 36 different schemes was the C4.5 Decision Trees with bagging. Table 1 shows the segmentation accuracies measured before and after post-processing compared with those of Gramofile using our error measurement. The error weights used were 0.6, 0.35 and 0.05, respectively. Note that Gramofile failed to identify any tracks in Album 4.

Table 1. Comparison of track segmentation accuracies between our algorithm (before and after post-processing) and Gramofile.

	Before post-proc (%)	After post-proc (%)	Gramofile (%)
Album 1	82.33	99.99	99.84
Album 2	72.71	99.99	99.77
Album 3	89.97	99.98	99.93
Album 4	69.98	99.98	35.71
Album 5	64.82	99.99	92.20
Album 6	76.87	99.97	99.89

Automatic metadata extraction from LP record labels

Metadata extraction is another time-consuming step of digitization and we attempted to automate the metadata extraction using document analysis and OCR techniques. As shown in Figure 1, a typical LP record label contains horizontal text lines, decorative graphical glyphs, and curved text strings. Our automatic metadata extraction aims to segment the text glyphs from the image and then extract the metadata by using OCR.

Text segmentation

As an unprecedented attempt towards record label segmentation, our approach is rule-based. The algorithm includes the following six steps:

1) Pre-processing
2) Binarization and connected component (CC) analysis
3) Removal of non-text glyphs
4) Separation of horizontal strings and curved strings
5) Separation of the lines of multi-line curved strings
6) Straightening of the curved strings

The algorithm is developed based on our open-source document analysis tool, GAMERA [15].

Pre-processing

A label image is first rotated so that the horizontal lines of text are truly horizontal. Because the scanned high-resolution label images are often in multicolor, a direct binarization applied on images tends to generate many broken text strokes due to the color diffusion present in the original labels. This would cause the CC analysis on the binarized image to fail; therefore, a color quantization using K-means clustering is performed to minimize color diffusion.

Removal of the non-text glyphs

The CCs resulted from the CC analysis comprise of not only text objects but also of graphical glyphs and noise. To eliminate as many non-text CCs as possible, a filter based on the size and location of CCs is used to eliminate obvious non-text glyphs and noise. We remove overly large and small CCs, namely large graphical glyphs and noise. The CCs that belong to the center hole of the label are also removed. The other non-text CCs are gradually removed in subsequent steps.

***Figure 1**. An LP record label with curved strings.*

***Figure 2**. An LP record label with multi-line curved strings.*

Separation of horizontal strings and curved strings

Several attempts were made to separate the curved strings from the non-curved strings. The first approach was to extract horizontally aligned CCs row by row and use the fact that curved strings are usually located near the rim of the disc label. However, this approach was inadequate and failed to isolate the characters around the rim when the curved characters are very close to horizontal text lines or when multi-line curved strings are present (Figure 2). Another approach was based on the observation that the

curved texts are usually arranged in concentric circles with the center being the label center. This approach, however, was also problematic because it is difficult to distinguish between a circle structure extracted from a real curved string and one from a non-curved text block with densely aligned rows.

From the experience that human tends to group things that are visually close together, we devised another discrimination strategy based on grouping by local continuity. A series of rules are used to assist the grouping process:

1) A group is composed of more than one CC.
2) The distance between two CCs from the same group is relatively short compared to the distance between two CCs from two different groups.
3) A group can only contain CCs from a single non-curved line or a single curved line.

A typical CC group as part of a curved string is shown in Figure 3. Two additional rules are used to differentiate between curved and horizontal CC groups:

1) If the group height, denoted by H_g, is greater than the height of the tallest CC member, H_c, in the group by a certain ratio, then this is a curved CC group.
2) If the standard deviation of the baselines of a group of CCs is greater than a threshold, then this is a curved CC group.

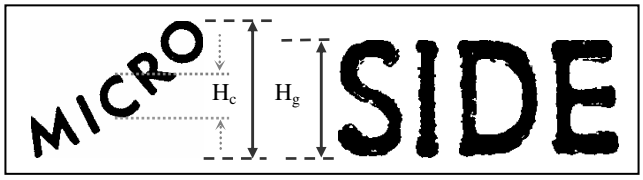

***Figure 3**. A typical local character group that belongs to a curved string (left) and a horizontal string (right). For a curved string, the group height (H_g) is significantly higher than the height of the tallest member CC of the group (H_c).*

Separation of the lines of multi-line curved strings

As mentioned before, the curved text strings on a record label are usually organized as concentric circles with the center being the label center, the location of which was determined during the CC analysis. We calculate the polar coordinates of all the CCs of the curved CC groups with the label center as the origin. Then by grouping the CCs that share a similar radius, we find the CCs that belong to the same concentric circle, i.e., the same line of curved strings. The azimuths of the CCs can be used to determine the parameters for the straightening process later.

Straightening the curved strings

Curved text strings need to be straightened into horizontal line for a successful OCR. The straightening involves two steps. First the CCs are rotated to their upright positions according to their azimuths. Then the inter-character spaces are calculated and applied to the rotated CCs. Note that a curved string can be arranged with the bottom of a character pointing to or away from the center depending on whether the curve is part of an upper circle or a lower circle. This was taken into account in the rotation step to avoid an inverted line.

Let α denote the azimuth of a CC. The rotation angle θ is calculated as follows, with counter-clockwise being the positive direction:

$$\theta = \begin{cases} -\alpha & \alpha \in [0,\ \pi/2) \\ \pi - \alpha & \alpha \in (\pi/2,\ \pi] \\ \alpha - \pi & \alpha \in (\pi,\ 3\pi/2) \\ 2\pi - \alpha & \alpha \in (3\pi/2,\ 2\pi] \end{cases} \quad (3)$$

The rotation angles on the positions $\pi/2$ and $3\pi/2$ are determined by whether the CCs at these positions belong to a group of an upper circle or a lower circle.

The inter-character space between two adjacent CCs in a straightened curved string, CC_n and CC_{n+1}, is calculated by:

$$D_{n,n+1} = \frac{r_n + r_{n+1}}{2} \cdot \left| \alpha_{n+1} - \alpha_n \right| \quad (4)$$

where (r_n, α_n) and (r_{n+1}, α_{n+1}) are respectively the original polar coordinates of CC_n and CC_{n+1} before being straightened.

OCR

OCR was performed by using commercial software ABBYY FineReader 8.0 [16]. The test dataset included 4 LP record label images from our collection scanned at 1200dpi. The character recognition performance is shown in Table 2, where accuracy is the ratio of correctly recognized characters and the total number of characters in a category. Note that after straightening, the recognition rate for the curved characters increased significantly.

Table 2. OCR accuracies (H: horizontal characters, C: curved characters, C1: non-straightened curved characters, C2: straightened curved characters, #: number of characters, Acc.: OCR character recognition rate)

	# of H	# of C	Acc. of H (%)	Acc. of C1 (%)	Acc. of C2 (%)
Label 1	204	20	99.02	0	100
Label 2	322	29	100	10.34	82.76
Label 3	259	162	100	0	99.38
Label 4	598	180	99.00	10.00	93.89

Automatic Derivative Creation

For the creation of media derivatives to be used for the web delivery, we use cross-platform, open-source software tools that are command-line based and can perform batch processing easily. We use ImageMagick [17] for image format conversion and libsndfile [18], SoX [19], and LAME [20] for audio format conversion. The conversion workflow is written in Unix Shell scripts and monitored remotely through SSH. A processing log was created and maintained in the digitization database to monitor the workflow.

Conclusions and future work

Several technical issues have been described concerning the digitization of large online collections of phonograph records. Our machine learning-based automatic track segmentation approach achieved better results than its peer tools. Our rule-based approach

to automatic metadata extraction from record labels was also effective.

Currently the feature used for our automatic track segmentation includes only RMS. It would be interesting to experiment with different features for the same task using a larger and varied collection of LPs. Also, an improvement should be made to our current label segmentation scheme, which assumes that there is only one foreground and one background color present in the image. The next major step in this endeavor is to automatically extract metadata and full text from album covers and liner notes using sophisticated document analysis techniques.

Acknowledgement

This research is funded in part by the "Richard M. Tomlinson Digital Library Innovation and Access Award," David Edelberg Foundation, CIRMMT, CFI, and FQRSC.

References

[1] A. Bezemer, "GramoFile Home Page." (2005). Accessed March 5, 2006. Available at http://www.opensourcepartners.nl/~costar/gramofile/.

[2] CFB Software," LP Ripper - Split Vinyl LP WAV Files into Tracks." (2006). Accessed March 5, 2006. Available at http://www.cfbsoftware.com/lpripper/lpripper.htm.

[3] Wieser Software Ltd, "LP to CD transfer made easy with RIP Vinyl from Wieser Software Ltd." (2006). Accessed March 5, 2006. Available at http://www.ripvinyl.com/.

[4] G. Tzanetakis, "Manipulation, Analysis And Retrieval Systems For Audio Signals." Ph.D. thesis, Princeton University, (2002).

[5] T. Zhang, and C. J. Kuo, Content-Based Audio Classification and Retrieval for Audiovisual Data Parsing (Springer, MA, 2000).

[6] L. Lu, S. Z. Li, and H. J. Zhang, Content-Based Audio Segmentation Using Support Vector Machines, Proc. ICME, pg. 956. (2003).

[7] R. Ferguson, "Automatic Segmentation in Concert Recordings," M.A. thesis, McGill University, (2003).

[8] D. Kimber, L.Wilcox, F. Chen, and T. Moran, Speaker Segmentation for Browsing Recorded Audio. Proc. CHI, pg. 212. (1995).

[9] K. El-Maleh, M. Klein, G. Petrucci, P. Kabal, Speech/Music Discrimination for Multimedia Applications, Proc. ICASSP, vol. 4, pg. 2445. (2000).

[10] D. McEnnis, C. McKay, I. Fujinaga, and P. Depalle, Feature Extraction: An Extensible Library Approach. Proc. ISMIR, pg. 600. (2005).

[11] C. McKay, R. Fiebrink, D. McEnnis, B. Li, and I. Fujinaga, ACE: A Framework for Optimizing Music Classification. Proc. ISMIR, pg. 42. (2005).

[12] Audacity Development Team, "Audacity: Free Audio Editor and Recorder." (2006). Accessed March 5, 2006. Available at http://audacity.sourceforge.net/.

[13] W. Chou, and L. Gu, Robust singing detection in speech/music discriminator design. Proc. ICASSP, vol. 2, pg. 865. (2001).

[14] Marvin Duchow Music Library, "David Edelberg Handel LPs." (2006). Accessed March 5, 2006. Available at http://coltrane.music.mcgill.ca/handel/lp/search.php.

[15] M. Droettboom, K. MacMillan, and I. Fujinaga, The Gamera framework for building custom recognition systems, Proc. SDIUT, pg. 275. (2003).

[16] ABBYY Software House, "Optical Character Recognition and PDF Conversion Software." (2006). Accessed March 5, 2006. Available at http://www.abbyy.com/finereader8/?param=44890

[17] ImageMagick Studio LLC, "ImageMagick: Convert, Edit, and Compose Images." (2006). Accessed March 5, 2006. Available at http://www.imagemagick.org/script/index.php.

[18] E. C. Lopo, "libsndfile." (2006). Accessed March 5, 2006. Available at http://www.mega-nerd.com/libsndfile/.

[19] Sourceforge.net, "SoX - Sound eXchange | HomePage." (2006). Accessed March 5, 2006. Available at http://sox.sourceforge.net/Main/HomePage.

[20] Sourceforge.net, "LAME Ain't an MP3 Encoder." (2005). Accessed March 5, 2006. Available at http://lame.sourceforge.net/.

Author Biography

Beinan Li is currently a Ph.D. student in Music Technology in the Schulich School of Music at McGill University, Canada. He received his bachelor's degree in Communication Engineering in 2001 and his master's degree in Pattern Recognition and Intelligent Systems in 2004 at Huazhong University of Science and Technology, China.

Catherine Lai received her BAs in music and applied math with an emphasis in computer science and M.I.M.S. in information management and systems from the University of California at Berkeley. She is currently a Ph.D. student in Music Technology at McGill University. The focus of her research is on music information retrieval. She is interested in exploring web-based music information retrieval constituting areas such as data/information presentation and user interface design.

Ichiro Fujinaga is Assistant Professor of Music Technology at McGill University. His research interests include optical music recognition, music perception, machine learning, and music information retrieval.