REAL-TIME SYNCHRONISATION OF MULTIMEDIA STREAMS IN A MOBILE DEVICE

Robert Macrae[1]*, *Joachim Neumann*[2], *Xavier Anguera*[2], *Nuria Oliver*[2] *and Simon Dixon*[1]

[1]Centre for Digital Music, Queen Mary University of London, Mile End Road, E1 4NS, London
[2]Telefonica Research, Torre Telefonica Diagonal 00, 08019, Barcelona, Spain
{joachim,xanguera,nuriao}@tid.es, {robert.macrae,simon.dixon}@elec.qmul.ac.uk

ABSTRACT

With the constant improvements in the technical capabilities and bandwidth available to mobile phones, mobile audio and video streaming services are booming. This allows music enthusiasts to watch their favourite music videos instead of only listening to the audio tracks, thus augmenting their listening experience to be multimodal. Usually the highly compressed audio tracks in these videos result in poorer quality music, compared to music the user might already have locally on their phone or playing through another sound source. In this paper we present MuViSync Mobile, a mobile phone application that synchronises real-time high quality music, either stored locally or input through the microphone, with the corresponding streaming music video. We extend previous work on music to music video synchronisation by proposing an alternative algorithm for higher efficiency with similar alignment accuracy, which we tested with music video examples including simulated noise. This algorithm correctly aligns 90% of the audio frames to within 100 ms of the known alignment. We also describe its implementation on an iPhone.

Index Terms— synchronisation, multimedia, multimodal

1. INTRODUCTION

At an increasing pace, mobile devices such as phones, tablets and mp3 players, are becoming powerful computers capable of delivering a full multimedia experience, *i.e.* audio, video, touch-screen interaction, etc. Such devices usually have internal storage capabilities that allow users to carry around their personal music collection in a high quality compressed format, and have high resolution displays that let them view videos either stored locally or streamed over a data connection. The user can therefore, instead of just listening to his favourite music, watch the corresponding music video at the same time. Indeed, many services nowadays allow for music video streaming, for example Youtube[1], LastFM[2] or Yahoo[3]; or music streaming, such as Spotify[4], Pandora[5] or GrooveShark[6]. Due to bandwidth limitations the audio in the streamed videos is usually of lower quality than when streaming audio alone, and neither is comparable to locally stored high quality audio files or music on audio CDs. To illustrate this, Table **??** compares the audio bit rates that these content streaming services typically provide. In general, since music video streaming services have to divide the limited bandwidth of a mobile data connection between the video and the audio content, the latter often suffers.

Music Video Streaming Service	Audio Track Bit Rate (Kbps)
LastFM	64-128
YouTube	64-128
Yahoo! Video	128
Music Streaming Service	Audio Bit Rate (Kbps)
GrooveShark	192
Pandora One	192
Spotify	160-320
Standard Audio formats	Downloaded Bit Rate (Kbps)
AAC from iTunes	256
MP3 from Amazon	256
Audio CD	1411

Table 1. Comparison of typical audio bit rates.

In this paper we solve this problem by proposing an extension of the MuViSync algorithm [**?**] to run on a mobile phone. In [**?**] we showed the feasibility of online synchronisation between audio media and an independent music video by: 1) finding an initial synchronisation between the two media, and 2) making an online alignment between the two to keep them synchronised. In the current paper we propose an alternative algorithm to improve its efficiency and implement the system on a mobile device (in our case, the Apple iPhone 4 and iPad). To the best of our knowledge, MuViSync Mobile is the first application that performs such a task on a mobile phone.

*The first author performed the work while at Telefonica.
[1]http://www.youtube.com
[2]http://www.lastfm.com
[3]http://www.video.yahoo.com
[4]http://spotify.com
[5]http://www.pandora.com
[6]http://listen.grooveshark.com

2. RELATED WORK

In this paper we propose an algorithm for music to music video alignment that is able to run on a portable device such as a mobile phone. We have researched prior work both on music to video (and in particular music video) synchronisation as well as work evaluating the relationship between audio/video streaming in mobile devices.

Our goal is to synchronise the information from two different sources, *i.e.* the music from a file or microphone input and a music video. A few papers propose to synchronise the two sources through an analysis of the visual track in the video stream. They usually extract suitable features from the video to be matched with features from the audio. In [**?**] and [**?**] an offline alignment is performed, which is not applicable to our problem due to the non-linear computational costs and the need to have complete sequences for the alignment.

In our case the music video contains an audio track which is expected to be similar to the audio stream; we therefore use audio-to-audio alignment algorithms, rather than trying to devise similar features to relate the video and the audio. Many techniques have been proposed over the years for this task, both for offline and for online processing, using techniques such as beat-tracking [**?**, **?**], Hidden Markov Models [**?**] or Dynamic Time Warping (DTW) [**?**]. Using beat-tracking, in [**?**] the tempo of a woman dancing within a video clip is altered to match that of the audio. Also, in [**?**], beat-tracking is used to align music with the drummer in a band in order to keep the music to the same tempo to what is being played. Similarly, Hidden Markov Models have been used for synchronisation [**?**]. These can be run online, but usually need prior training. Finally, techniques derived from DTW [**?**] can be used to align any two audio segments. The classical DTW implementation is suitable for offline alignment, although it has quadratic complexity in computational and memory costs, which makes it not scalable for long sequences or usable in real-time. For this reason various modifications have been proposed over the years to improve its efficiency, such as Sakoe and Chiba's bounds [**?**], Itakura's slope constraints [**?**] and Salvador and Chan's multi-resolution "FastDTW" [**?**], as well as variations in the local constraints imposed on the dynamic path finding algorithm [**?**]. These algorithms are not applicable for our purpose since they are offline in nature. Other modifications to the classical DTW, like [**?**], [**?**] and [**?**] allow for an online alignment. In [**?**] the application of slope constraints together with a progressive DTW method is used for online synchronisation of two audio files. Due to the nature of the path generation algorithm, this method can result in either a discontinuous real-time alignment or suffers from latency incurred by waiting for the path to be firmly established. [**?**] proposes an alternative progressive local DTW algorithm to effectively align two audio sequences. This algorithm constitutes the starting point for the present work and is reviewed in Section **??**.

In order to meet the constrains of a mobile phone platform, we propose a novel algorithm that simplifies the approach in [**?**], making it much faster while preserving the accuracy.

Very little research has looked into the impact of streamed music videos in the context of the user's listening experience. In [**?**] the authors analyse the suitability of mobile devices for watching streaming video. They analysed users' responses to devices with streaming capabilities and found two situations to be particularly appealing. One is when a person is alone and is looking for a way to use his free time, for example, listening to music or watching a video. The other is in social contexts, for example singing karaoke with friends. Both situations match well with the capabilities of the system proposed in this paper, either synchronising with the user's local music library or to music in the local environment.

Given the availability of audio and video in a mobile device, in [**?**] the authors evaluated the relationship between audio and video quality perception on mobile devices for different kinds of media. They explicitly tested music videos and concluded that the audio was playing a much more important role with respect to user perception than the image. Similarly, [**?**] shows, through a user study for multimedia content including music videos, that changes in video streaming bit rates do not greatly affect the user's perceived quality, while small changes in the audio quality had a large effect on the user's perception.

3. SYNCHRONISATION ALGORITHMS

This section examines the methods for synchronising music and corresponding music videos proposed in order to drive MuViSync Mobile video playback. This process is based on audio to audio alignment as we focus on the audio soundtrack of the video and use this to adjust the video playback which fits the original music audio as best as possible. The specifics of MuViSync's application leads to a number of requirements for this audio-audio alignment method:

- **Real-time**: For live audio synchronisation, the algorithm has to run in real-time with incomplete data. The alignment also has to be calculated ahead of the video playback, where a user may even skip forward.
- **Efficient**: The processing power available to make the alignment in a mobile phone is limited but cannot detract from the video playback experience.
- **Unknown start and end points**: Since the audio and video streams do not necessarily start simultaneously, an initial alignment has to be determined.
- **Structural differences**: The two audio sequences are not guaranteed to have the same structure as one sequence could be incomplete (missing segments) or contain added segments.

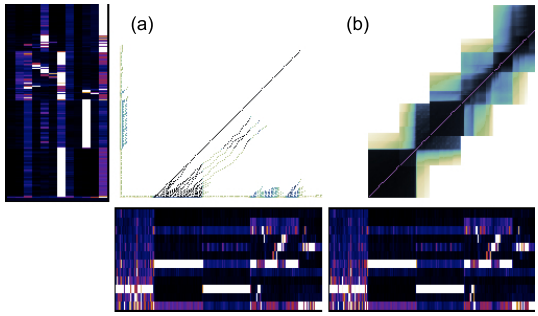

Fig. 1. Original MuViSync algorithms showing (a) The *Initial Path Discovery Algorithm*, (b) The *Sequential-DTW Based Approach* and the input chroma feature sequences.

Typical offline audio-audio alignment techniques, such as DTW, make use of the entire sequence information to guarantee the alignment is optimal with respect to the chosen path constraints and features. Our aim is to match, as closely as possible, the accuracy performance of a typical offline audio-audio alignment technique, whilst achieving the above stated goals. First we shall review the algorithms used from [**?**]. Then we shall describe a new synchronisation algorithm that we designed specifically for MuViSync Mobile.

3.1. Feature Extraction

For our audio-audio alignment, we need to extract contextual information from the audio signals, that can be used to relate and synchronise the two signals in a robust and computationally efficient manner. For this purpose we use chroma features, a 12 dimensional representation of musical pitch content that is often used for musical information retrieval tasks [**?**]. Given an input music file and a music video file, we take the two sequences of audio S_1 and S_2 and divide them into 186 ms overlapping frames with a hop size of 23 ms, filtered with a hamming window, and then transformed into the frequency domain using a standard Fast Fourier Transform. The resulting spectrum is then mapped into 12 dimensions, corresponding to the pitch classes of Western music. The resulting values are then normalised for each frame to avoid differences in volume affecting the alignment. This results in two feature sequences $U = (u_1, u_2, ..., u_M)$ and $V = (v_1, v_2, ..., v_N)$ for each of the two audio sequences being aligned.

3.2. MuViSync Algorithms

3.2.1. Initial Path Discovery

In previous work [**?**], an algorithm was devised to make a quick estimation of the start position of the two audio feature sequences, U and V. This makes use of multiple greedy searches through a (partially calculated) similarity matrix $\mathcal{S}(m, n)$ of costs between the two feature sequences S_1 and S_2 to find a path $P_g = (p_{g1}, p_{g2}, ..., p_L)$ of length L. The local cost is calculated as the inner product subtracted from 1, $d_{U,V}(m, n) = 1 - \frac{<u_m, v_n>}{\|u_m\|\|v_n\|}$, to ensure lower path costs are optimal. We call these searches greedy as they do not calculate the accumulated cost, as is normal in DTW, but instead only consider the cost of the current step. The cost $D(m, n)$ at any location (m, n) , can be calculated as $D(m, n) = d_{U,V}(m, n) + min[D(m-1, n-2), D(m-1, n-1), D(m-2, n-1)]$. With this approach, the path is constrained by a minimum and maximum slope of $\frac{1}{2}$ and 2 respectively. Initially, we perform an initial greedy search path for every possible position where either the audio or the video are at the initial frame *i.e.* (U_1, V_n) or (U_m, V_1). Then a path selection procedure is applied in order to prune unsuitable initial paths: after each path is progressed a step, all the paths whose overall cost $D(P_g)$ above the average cost (of all the paths currently in consideration) are discarded and when two paths collide, the path with the highest cost is discarded. When one path remains, this is selected as the optimal starting point. This *Initial Path Discovery* can be seen in Figure **??** (a).

3.2.2. Sequential-DTW Based Approach

Once the initial path is discovered, the two feature sequences need to be kept synchronised. Previously [**?**], MuViSync made use of an adaptation of DTW, for real-time, that we will hence forth reference as the *Sequential-DTW Based Approach* or *SDBA*. We use this term as the synchronisation is calculated by breaking the process into a series of small DTW steps that would be self guided. Combined with the *Initial Path Discovery* mentioned above, this algorithm is capable of keeping music and music videos synchronised within 100 milliseconds of each other in over 90% of the alignment points examined. It also performs the alignment of two pieces of 3 minutes in length in approximately 1 second CPU time on a standard desktop computer. In the *SDBA*, a single greedy search $P_g = (p_{g1}, p_{g2}, ..., p_L)$ is computed, similar to the paths found in the *Initial Path Discovery* in Sec. **??**. Once this line has reached a distance of 5 seconds on either of the feature sequences, a standard DTW is calculated between the start and end point (P_{g1} to P_{gL}) of this path. The overall cost for the DTW path at any location (m, n) can be computed as $D(m, n) = d_{U,V}(m, n) + min[D(m-1, n), D(m-1, n-1), D(m, n-1)]$ to give a path $P_{dtw} = (p_{dtw1}, p_{dtw2}, ..., p_{dtwL})$ that contains the optimal alignment between both signals for that time segment. From half way along this DTW path, a subsequent greedy path starts in $p_{g1} = p_{dtw\frac{L}{2}}$. Subsequent DTW sub-paths are computed until the end of either source is reached. The initial halves of each DTW sub-path are joined to create the final alignment path as seen in Figure 1(b).

3.3. MuViSync Mobile Algorithm

3.3.1. Diagonally-Constrained Greedy Search

In this paper we introduce a modification of the SDBA algorithm for increased efficiency in mobile devices. We call

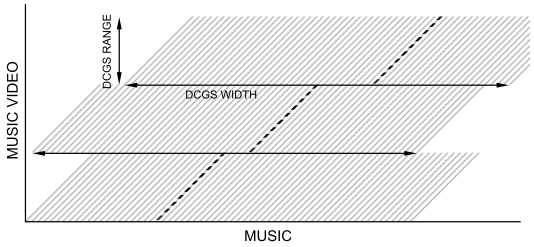

Fig. 2. The *Diagonally-Constrained Greedy Search*. The light grey lines represent the multiple paths that are calculated. The dashed lines represent the lowest cost paths for each segment that then get selected for the final *DCGS* path.

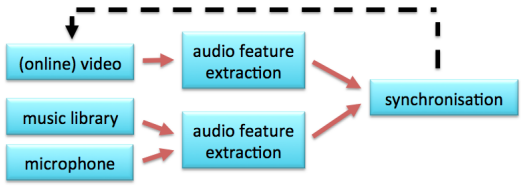

Fig. 3. MuViSync Mobile extracts audio from either the device's microphone or the device's music library. Simultaneously, the application streams video content from YouTube, plays music, extracts features and synchronises the video playback in real-time, ensuring the video is synchronised. The algorithm maintains the synchronisation by altering the video as people are more preceptive to jumps in the audio.

this algorithm the *Diagonally-Constrained Greedy Search* (DCGS), which was motivated by an analysis of typical music and music video data, which revealed that most of the time the tempo is very similar between music and corresponding music videos. The *DCGS* is also based on the *Initial Path Discovery* step (see section **??**) but restricts the multiple greedy paths to a diagonal constraint so that the cost can be calculated at any location (m, n) as $D(m, n) = d_{U,V}(m, n)+D(m-1, n-1)$. By doing this, we are assuming the sources maintain roughly the same tempo, within short time periods, so that the algorithm is simply required to find the diagonal that has the least cost. This time period is referred to as the DCGS range which we set to 2 seconds (as this was required in order to avoid loosing sufficient accuracy when noise is applied). We calculate all the possible diagonal paths within a pre-defined search range, which we refer to as the *DCGS* width (which we vary in our experiments). The path $P_g = (p_{g1}, p_{g2}, ..., p_L)$, with the lowest cost $D(P_g)$ is appended to the *DCGS* path and the process is repeated from the final point P_{gL}, as illustrated in Figure **??**.

4. MOBILE IMPLEMENTATION

Applications that benefit from a synchronisation between the audio track of a video and an audio stream are not limited to web based or desktop computer based scenarios. While on the go, mobile devices can be used in two example scenarios. One is to improve the users multimedia experience by providing synchronised videos when music is playing in the device's environment. Another is to replace the audio of streamed online video content on a handset by using high-quality songs from the devices local music library. For this purpose we have developed MuViSync Mobile. In comparison to our desktop computer implementation [**?**], the mobile implementation needs to comply with tight restrictions of CPU power and limited system memory of current (as of 2010) smartphones. As a platform for the implementation of the proposed system, we chose Apple's iOS SDK 4.1 and the iPhone 4 / iPad hardware. However, our aim was to assure that the implementation of the MuViSync Mobile library remained cross-platform so all algorithmic source code is written in ANSI C++. The only part of the two mobile applications that is not platform-independent is the hardware dependent audio I/O and the graphical user interface provided by the iOS SDK. In order to optimise our algorithm for a mobile implementation we used a profiler tool to identify which processing blocks are most computationally expensive in the system. Aside from the alignment algorithm, we identified the feature extraction as a performance bottleneck. We therefore optimised it by applying a set of engineering changes to the standard extraction approach, consisting of (a) joining the required FFT for the two audio streams into a single complex FFT transformation; (b) dynamically switching between the platform-independent implementation of the FFT and (when available) the accelerate framework that is provided by iOS which offers an optimised FFT implementation, utilising vector instructions of the iPhone's ARM7 CPU and (c) manually optimising critical loops in the code. Finally, further optimisation of the *Sequential-DTW Based Approach* includes a reduction in the width of the DTW blocks calculated so that it is less computationally intensive while maintaining the synchronisation performance. The DCGS algorithm was specifically designed with a mobile implementation in mind.

5. EVALUATION

MuViSync Mobile aims at synchronising music videos on low powered devices, in real-time and in potentially noisy environments. Therefore, in order to assess the suitability of our synchronisation algorithms we evaluate the accuracy and efficiency of the two algorithms with a dataset consisting of matching music and videos. The evaluation is run on a standard computer with simulated noise, equalling what is expected when recording audio through a smartphone.

5.1. Ground Truth Test Data

The evaluation material we used consists of matching pairs of music files from Amazon[7] and music videos from YouTube[8].

[7] www.amazon.com
[8] www.youtube.com

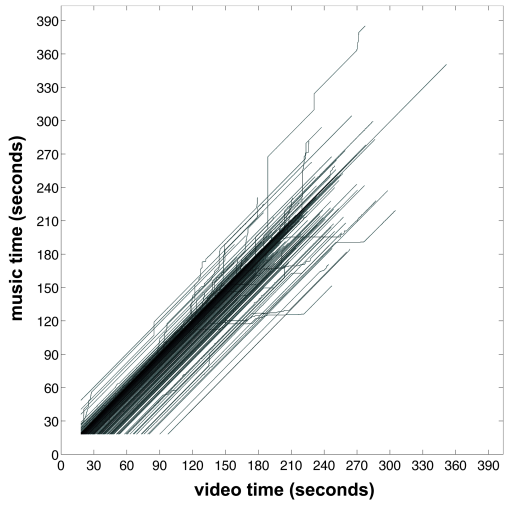

Fig. 4. Ground truth alignments for the music video data set calculated using an offline DTW algorithm.

In order to obtain the ground truth alignments between these files we used a standard off-line DTW, based on chroma features, and manually checked the alignments to be correct, pruning out any erroneous assignments. Although DTW is based on dynamic programming, similar to the techniques discussed here, DTW has the advantage of using the complete sequences. We used Meinard Müller et al's Chroma Matlab Toolbox[9] [**?**] with a hop size of 20 ms to extract chroma sequences from the two sources and aligned these using Dan Ellis's DTW in Matlab Toolbox[10] [**?**]. The finished dataset consists of 356 matching music, music video and reference alignment files. The ground truth alignments for all 356 music video sets can be seen in Figure **??**, where although many alignments are diagonal, some have structural differences and have different start-end points between the sequences.

5.2. Algorithm Comparison

Next we perform a comparison of the *DCGS* and the *SDBA* MuViSync algorithms. We measured the synchronisation accuracy by counting how many of the reference alignment points were discovered correctly, within varying degrees of accuracy requirement. Previous work [**?**, **?**] differs on the human perception of music and video asynchrony (in that we are less perceptive to audio lagging video) with estimated ranges and requirements of $15 - 45$ ms or $80 - 100$ ms. We have decided on two accuracy requirements of 25 ms for an audio lead and 100 ms for a video lead. At the present stage, our algorithms does not (yet) strive to reflect this asymmetry, consequently, we test our algorithms by determining which percentage of the resulting multi-media stream is within 25 ms and 100 ms to our ground truth. Tables **??** and **??** show

[9] Available at www.mpi-inf.mpg.de/ mmueller/chromatoolbox/
[10] Available at http://labrosa.ee.columbia.edu/matlab/dtw/

SDBA Algorithm					
Width (s)	Accuracy (%)				Time (s)
	No Noise		1:1 SNR		
	25 ms	100 ms	25 ms	100 ms	
2 s	81.1	89.3	74.8	87.8	0.296 s
4 s	81.4	89.6	75.5	88.6	0.589 s
6 s	82.3	90.5	76.1	89.4	0.886 s
8 s	82.8	91.2	76.2	89.5	1.178 s

Table 2. Evaluation of the *Sequential DTW* Based Approach (*SDBA*). The accuracy is the percentage of audio samples in the mixed multi-media stream that is correctly synchronised within 25 ms and 100 ms, respectively.

DCGS Algorithm					
Width (s)	Accuracy (%)				Time (s)
	No Noise		1:1 SNR		
	25 ms	100 ms	25 ms	100 ms	
2 s	81.7	89.4	81.2	89.0	0.181 s
4 s	83.7	91.6	81.5	89.1	0.361 s
6 s	84.4	92.4	79.1	86.4	0.548 s
8 s	85.3	93.3	77.3	84.4	0.724 s
10 s	85.6	93.8	78.3	85.7	0.907 s

Table 3. An overview of the *Diagonally-Constrained Greedy Search* (*DCGS*) algorithm using different video sequence search widths for the diagonal method, with noise and without.

the *SDBA* and *DCGS* algorithms at various window sizes and with and without noise. We simulated a signal-to-noise ratio (SNR) by combining the chroma features with randomly generated white noise, as this allowed us to efficiently control the SNR in a way that was consistently reproducible. For an SNR of 1:1, the energy of the added noise is equal to the energy of the chroma features of the signal. Recording through an iPhone's microphone, we found a typical SNR of 1.14, meaning the signal was slightly more significant than the noise.

Our evaluation shows that both algorithms yield a reasonable synchronisation between the audio stream and the audio track of the video stream in difficult conditions (real-time processing without knowledge of the future and with the hardware restrictions of the mobile platform). On examining Table **??**, we can see that the width of the sub-DTW boxes (first column) has a strong effect on the average execution time (last column), but a larger width results in a slightly better accuracy. In Table **??**, the width of the *DCGS* algorithm refers to the size of the search window, which affects the computation time in a similar fashion and also improves the accuracy slightly. A direct comparison of the two algorithms confirms that the *DCGS* algorithm indeed executes faster. When comparing the accuracies of the two algorithms, we see that the *DCGS* algorithm performs better in three out of four conditions. Only in the presence of noise and with the more tolerant

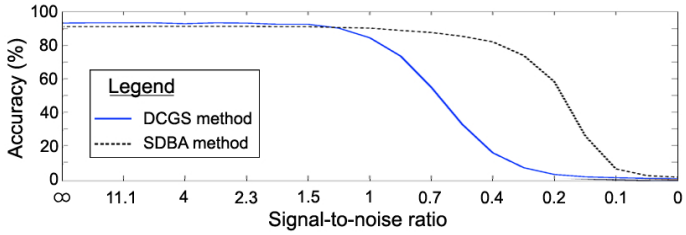

Fig. 5. A comparison of how accurate the *SDBA* algorithm and *DCGS* algorithm are (within 100 ms) at different ratios of signal to noise. Both algorithms had a width of 8 seconds.

100 ms criteria, the *DCGS* algorithm showed a worse performance. In order to shed further light on the comparison of the accuracy of the two algorithms, we evaluate the accuracy of the two algorithms for various noise levels.

Figure **??** shows the effect that noise, added to the computed chroma features, has on the accuracy of the synchronisation. In the absence of noise (signal-to-noise ratio ∞), the graph reflects the numbers given in Tables **??** and **??**. When we add noise to the chroma features, we observe that the accuracy of both algorithms decreases. An informal measurement of the chroma features of a song that is either played from the library or captured with the device's microphone in a quiet room indicates that a signal-to-noise ratio of the chroma features of about 1.14 is a realistic operational point, which corresponds to 92.1% accuracy for DCGS and 90.3% for SDBA. We also see that the *DCGS* algorithm performs worse at larger signal-to-noise ratios (for the weaker criterium of 100 ms).

6. CONCLUSIONS

MuViSync Mobile is designed to take advantage of the growing computational power of smartphones to resolve the discrepancy between the poor audio bit rate of music video streaming services and the high quality music we already own. In this work we have presented a novel, fit for purpose, synchronisation technique that is efficient, able to synchronise music and music videos on low powered devices and have implemented this system on an iPhone as a proof of concept. In our evaluation we have compared this algorithm with previous work and found an improvement in the accuracy/processing time trade off.

7. REFERENCES

[1] Robert Macrae, Xavier Anguera, and Nuria Oliver, "Muvisync: Realtime music video alignment," in *Proceedings of IEEE International Conference on Multimedia and Expo-ICME*, 2010.

[2] Xian-Sheng Hua, Lie LU, and Hong-Jiang Zhang, "Automatic music video generation based on temporal pattern analysis," in *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, NY, USA, 2004, pp. 472–475, ACM.

[3] Jong-Chul Yoon, In-Kwon Lee, and Siwoo Byun, "Automated music video generation using multi-level feature-based segmentation," *Multimedia Tools Appl.*, vol. 41, no. 2, pp. 197–214, 2009.

[4] Andrew Robertson and Mark Plumbley, "B-keeper: a beat-tracker for live performance," in *NIME '07: Proceedings of the 7th international conference on New interfaces for musical expression*, NY, USA, 2007, pp. 234–237, ACM.

[5] Tristan Jehan, Michael Lew, and Cati Vaucelle, "Cati dance: self-edited, self-synchronized music video," in *SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications*, New York, NY, USA, 2003, p. 1, ACM.

[6] Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of speech recognition*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[7] Hiroaki Sakoe and Seibi Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[8] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. on Acoustics, speech and signal processing*, vol. 23, pp. 52–72, 1975.

[9] Stan Salvador and Philip Chan, "FastDTW: Toward accurate dynamic time warping in linear time and space," in *Workshop on Mining Temporal and Sequential Data*, 2004, p. 11.

[10] Simon Dixon, "Live tracking of musical performances using on-line time warping," in *Proceedings of the 8th International Conference on Digital Audio Effects*, Madrid, Spain, 2005, pp. 92–97.

[11] Robert Macrae and Simon Dixon, "Accurate real-time windowed time warping," in *Proc. ISMIR*, 2010.

[12] Petteri Repo, Kaarina Hyvonen, Mika Pantzar, and Paivi Timonen, "Users inventing ways to enjoy new mobile services - the case of watching mobile videos," in *Proc. Hawaii International Conference on System Sciences*, 2004.

[13] Satu Jumisko-Pyykko and Jukka Hakkinen, "Evaluation of subjective video quality of mobile devices," in *Proc. ACM MM*, 2005.

[14] Stefan Wilkler and Christof Faller, "Perceived audiovisual quality of low-bitrate multimedia content," *IEEE Trans. Multimedia*, vol. 8, pp. 973 – 980, 2006.

[15] Meinard Müller, *Information Retrieval for Music and Motion*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[16] Robert J. Turetsky and Daniel P.W. Ellis, "Ground-truth transcriptions of real music from force-aligned midi syntheses," in *In 4th International Conference on Music Information Retrieval*, 2003, pp. 135–141.

[17] Advanced Television Systems Committee, "Relative timing of sound and vision for broadcast operations," *ATSC Implementation Subcommittee Finding*, p. 191, June 2003.

[18] Isidor Kouvelas, Vicky Hardman, and Anna Watson, "Lip synchronisation for use over the internet: Analysis and implementation," in *Proceedings of the IEEE Conference on Global Communications, GLOBECOM96*, 1996, vol. 2, pp. 893–898.