MUVISYNC: REALTIME MUSIC VIDEO ALIGNMENT

Robert Macrae[*]

Centre for Digital Music, Queen Mary University of London
Mile End Road, E1 4NS, London
robert.macrae@elec.qmul.ac.uk

Xavier Anguera, Nuria Oliver[†]

Telefonica Research
Via Augusta 177, 08021 Barcelona, Spain
{xanguera,nuriao}@tid.es

ABSTRACT

In recent years, the popularity of compressed music files and online music downloads has increased dramatically. Today's users own large digital collections of high quality music on their computers and portable devices to be played in their homes or on the go. In addition, music videos are being offered online both for free and through monthly subscriptions, opening up the opportunity to turn the music listening activity into a multimedia experience. The work presented in this paper addresses the challenge of music audio and music video synchronisation. In particular, we have developed a prototype, named *MuViSync*, to automatically synchronise music videos to the songs that users are listening to in real-time. At the core of the *MuViSync* prototype are novel audio synchronization algorithms to tackle the differences in tempo, pitch, sampling rates, structure, and introductions and endings that are common in the various digital recordings of the same song in modern music. The music and the music video are initially aligned and then kept in sync within the limits of human perception. In experiments with 320 matching pairs of audio files and music videos, the proposed algorithms successfully synchronise music and video to within 100 milliseconds of each other in over 90% of the cases.

***Keywords*—** Music, Video, Synchronisation, Alignment

1. INTRODUCTION

The music industry has been targeting online music retail as an area of growth and as such there are an increasing number of users now buying their music from popular websites (*e.g.* Amazon[1]) or directly through media players (*e.g.* iTunes[2]). Although some of these sites also offer music videos, users already have large collections of audio-only content and this type of content is pervasive in portable music players.

In addition, users can now pay monthly subscription fees to stream music and videos over their internet connection using online services such as Spotify[3], Rhapsody [4] or Last.fm[5]. Alternatively, there are an increasing number of music videos uploaded by record labels on free to view media sharing sites (*e.g.* YouTube[6], Yahoo![7] Videos). However, the quality of these videos, and in particular their audio track, is generally low, especially when compared to the quality of the audio content that users have on their computers. For example, the typical audio attached to a YouTube video is 64 kbps, single channel with a sampling rate of 22 kHz. In contrast, iTunes AAC files are recorded in stereo at 256 kbps with a sampling rate of 44.1 kHz.

Therefore, there is an opportunity and a challenge for combining high quality music with its associated music video in order to provide a seamless high quality multimodal music experience to users. To the best of our knowledge, the system presented in this paper is the first to tackle this problem.

In order to combine the high quality music with the streamed music video, we have developed a prototype application named *MuViSync* that includes several novel robust audio synchronization algorithms. *MuViSync* combines a user's offline music library with online music video content, by automatically and seamlessly synchronising the music and videos, in real-time.

The remaining of the paper is structured as follows: The relevant work in audio/video synchronisation is reviewed in Section 2. Section 3 describes the proposed system, followed by an evaluation in Section 4. Finally, we summarise our conclusions and highlight lines of future work in Section 5.

2. RELATED WORK

We have found in the literature two related approaches to tackle the problem of music to video alignment: (1) *Audio to Video Matching*, where the audio track of the video is not analysed and the problem is considered as one of video to music alignment; and (2) *Audio to Audio Matching*, when the audio in the video matches the music, the audio channel in the video is analysed and standard audio to audio alignment methods are used in order to determine how to then warp the video to the song.

2.1. Audio to Video Matching

There have been a number of research projects aimed at aligning audio and video tracks to synchronise background music, in

[*]The first author performed the work while at Telefonica.

[†]During the development of this work X. Anguera was partially funded by the Torres Quevedo Spanish program

[1]http://www.amazon.com
[2]http://www.itunes.com
[3]http://www.spotify.com
[4]http://www.rhapsody.com
[5]http://www.lastfm.com
[6]http://www.youtube.com
[7]http://video.yahoo.com

an offline fashion, for sports videos [1], home videos [2] and amateur music videos [3]. These methods extract suitable video features and match these to a comparable set of features from the audio to find appropriately matching segments within the two sources. However, there is no bias for a linear progression through both recordings, which is desirable in this case. The work by Yoon, Lee and Byun [4] is an example of aligning music with arbitrary videos by comparing multi-level features.

2.2. Audio to Audio Matching

In audio to audio alignment, a common approach consists of synchronising both signals by means of either beat-tracking [5], Hidden Markov Models (HMMs) [6] or Dynamic Time Warping (DTW) [7] techniques. Unlike in audio to video techniques, the start times of both pieces are required.

Cati Dance [8] proposes a beat-tracking approach that alters the tempo of a video clip of a woman dancing to match the tempo given by a beat-tracker for different audio clips. *B-Keeper* [5] uses beat-tracking to keep a piece of music in sync with a drummer so that bands can loop audio segments that keep to the same tempo with what is being currently played.

Real-time systems, such as those used in speech recognition, tend to use HMMs to calculate likelihood states from observed features (*e.g.,* Mel Frequency Cepstral coefficients or MFCC). HMMs require training on suitable data to learn the model parameters (probabilities). This approach has been used to synchronise music with scores [9], lyrics [10] and also for video segmentation [11], among others.

Conversely, Dynamic Time Warping (DTW) [6] is typically used to find the best alignment path between two audio pieces in an offline context. However, the cost of computing the accumulated cost matrix and the path through this matrix does not scale efficiently for large sequences. Over the years there have been a number of efforts to improve the efficiency of DTW, such as Sakoe and Chiba's bounds [7], Itakura's slope constraints [12] and Salvador and Chan's multi-resolution "FastDTW" [13], as well as variations in the local constraints imposed on the dynamic path finding algorithm [6]. Alternatively, the *SyncPlayer* system [14] uses an offline DTW alignment with pre-computed alignment paths in order to provide metadata (scores and lyrics) in sync with the music that the user is playing.

Another limitation of the standard DTW approach for real-time is that it requires knowledge of both the start and end points of the sequences to align. In order to locate matching sub-paths within two sequences, Muller proposes in [15] a "Path Constrained" offline DTW that discovers multiple matching segments within a similarity matrix.

In Online Time Warping [16] (OTW), however, Dixon has shown it is possible to perform DTW in real-time. OTW combines slope constraints with an iterative and progressive DTW method so that it can synchronise two audio files or one audio file to live music. Due to the nature of how different paths can be switched to, within the bounding limits, the OTW method results in either a jumping real-time alignment or a delay incurred by waiting for the path to be firmly established. This makes OTW unsuitable for the problem tackled in this paper.

In sum, all of the presented algorithms require the starting point to be known and/or the structure of the two media to be the same. Hence, none meet the requirements of synchronising music to music videos in real-time and adjusting to structural differences in the recordings. The *MuViSync* prototype proposed in this paper performs such an alignment in real-time by: (1) automatically finding the starting point of the matching music video and then (2) synchronising the two sources from this starting point onwards.

3. MUVISYNC'S SYNCHRONISATION ALGORITHMS

MuViSync's synchronization algorithms are originally based on existing audio to audio alignment methods that are partly suitable because the audio to be aligned corresponds to the same content in both sources (music and music video). However, in order to fulfil our objectives of real-time synchronisation with smooth playback, we propose the following modifications to the standard DTW algorithm: (1) We calculate the paths in an iterative, progressive manner that allows for the end point to be unknown, as it is dependent on future audio content not yet received. These progressive steps are guided by an efficient forward path finding algorithm that is also used to compare and discover the correct starting position; and (2) rather than computing the entire similarity matrix of frame by frame difference costs, only the likely pairs that the paths may traverse are calculated.

Given an input music file S_1 and a music video file S_2, composed of a video track S_{2v} and an audio track S_{2a} to be synchronised with S_1, *MuViSync* proceeds as follows:

- **Initial buffering:** Retrieve initial buffers of S_1 (30-60 seconds) and S_{2a} (10-30 seconds) and compute their chroma features (explained in Section 3.2).
- **Initial path discovery:** Find the initial point of alignment using a multi-path selection approach (Section 3.3).
- **Real-time alignment:** Continue computing feature vectors and follow an incremental DTW guided by a forward path selection (Section 3.4).
- **Post-alignment processing:** Apply a smoothing function to the alignment and use the average differences between the audio and video to update the video playback where necessary (Section 3.5).

The challenge for real-time, as opposed to offline, DTW methods is in working efficiently and without the complete information, *i.e.* the full similarity matrix. Without the full similarity matrix, the DTW path is not guaranteed to be optimal and therefore the accuracy of the alignment may be adversely affected. Hence, the goal for the real-time DTW alignment is to be as accurate as that of a standard offline DTW method. Next, we briefly review the theory behind the classic DTW approach, as it is the basis for our algorithm, followed by a description of the two proposed improvements.

3.1. Dynamic Time Warping

Given two feature sequences $U := (u_1, u_2, ..., u_M)$ and $V := (v_1, v_2, ..., v_N)$, the standard DTW algorithm (see [6] for an introduction) finds the optimum path through the cost matrix $\mathcal{S}(m, n)$ with $m\epsilon[1 : M]$ and $n\epsilon[1 : N]$. The metric used in the cost matrix varies depending on the implementation: the Euclidean distance (the path represents the minimum average cost) or the inner product similarity (the path represents the maximum average similarity) are among the two most common metrics. In our implementation, we use a normalised inner product distance, $d_{U,V}$, which gives a value of 0 when both frames are identical, as is given by:

$$d_{U,V}(m, n) = 1 - \frac{< u_m, v_n >}{\|u_m\|\|v_n\|} \quad (1)$$

The result of the DTW algorithm is a minimum cost path $P := (p_1, p_2, ..., p_W)$ of length W, where each $p_k := (m_k, n_k)$ indicates that frames u_{m_k} and v_{n_k} are part of the aligned path at position k. The optimal $P$is chosen so that it minimises (or maximises, depending on the metric chosen) the overall cost function $D(P) = \sum_{k=1}^{W} d_{U,V}(m_k, n_k)$ and satisfies the: (1) *boundary*, $p_1 = (1, 1)$ and $p_W = (M, N)$, and (2) *monotonicity*, $m_{k+1} \geq m_k$ and $n_{k+1} \geq n_k$ for all $k\epsilon[1, L]$, conditions.

Additionally, local constraints are imposed that define the values that $(m_k, n_k)$are allowed to take with respect to their neighbors, *e.g.* $(m_{k-1}, n_{k-1}) = (m_k - i, n_k - j) \,|\, \underset{(i,j)}{\operatorname{argmin}}\{D(m_k + i, n_k + j)\}$. A common constraint is shown in Fig. 1b, where $(i, j)\epsilon \{(0, -1), (-1, 0), (-1, -1)\}$. The overall cost at any location (m, n) can be computed via dynamic programming as $D(m, n) = d_{U,V}(m, n) + min[D(m - 1, n), D(m - 1, n - 1), D(m, n - 1)]$. Other local constraints are presented in [7, 12, 6]. The computation of the cost matrix $\mathcal{S}(m, n)$ for all values of m and n has a quadratic cost with respect to the length of the feature sequences U and V. For this reason constraints are usually applied to bound how far from the diagonal the minimum cost path is allowed to go.

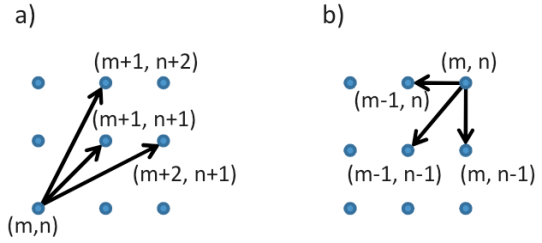

Fig. 1. Local path constraints: (a) forward path for initial path discovery, (b) backward path for real-time alignment

3.2. Audio Feature Extraction

The two sequences of audio S_1 and S_{2a} are divided into 200 ms overlapping frames with a hop size of 100 ms, filtered with a hamming window, and then transformed into the frequency domain using a standard Fast Fourier Transform. The resulting spectrum is mapped onto a 12-dimensional normalised chroma representation. The 12 dimensions of the chroma bins correspond to the 12 notes found in western music thus effectively the audio is reduced to a single octave. Chroma features are typically used in music alignment as they are robust to variations in how the music is played. For a more in depth explanation of chroma features see [15].

3.3. Initial Path Discovery Algorithm

Due to the boundary condition associated with a typical DTW method, prior knowledge of the start and end coordinates is typically required in order to compute the optimal DTW path. However, in the real-time case of synchronising music videos the start and end points of the music are unknown.

Therefore, the first step is to discover the starting point before making an estimate of the end point. We propose a forward path finding algorithm to discover an initial path $P_i := (p_{i1}, p_{i2}, ..., p_K)$ of length K that corresponds to the optimum initial alignment between feature sequences U and V by minimising $D(P_i)$. For this path, we use the local constraints shown in Figure 1a, which are equivalent to the forward path version of the type III local constraints described by Rabiner and Juang in [6]. The global cost $D(m, n)$ at any location (m, n) can be found in our implementation 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 rate of $2x$ and $\frac{1}{2}x$ respectively.

In order to find the optimal starting position within a given buffer of audio $U := (u_1, u_2, ..., u_M)$ and video $V := (v_1, v_2, ..., v_N)$, the forward 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) is computed. 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 $D(P_i)$ is above the average cost (of all the paths currently in consideration) are discarded. When two paths collide, the path with the highest cost is discarded. This selective process is suspended during silent frames as the noise from these frames would make the selection process random. Figure 2.a shows the path pruning effect on multiple forward paths.

When there is only one path remaining (after approximately 275 ms [8]), the real-time synchronisation is started from that point. If the limit of the buffer is reached and there are multiple paths still in contention, the lowest cost path is taken as the initial alignment (although this rarely occurred in our experiments). Therefore the size K of the initial path P_i is either the length of the only remaining path or the length of the audio buffer.

[8]The exact time it takes to discover the initial path depends on the amount of frames being considered. This selective procedure is demonstrated in Figure 2 in the middle similarity matrix. There is a detailed exploration of the accuracy and time trade-offs of such buffer lengths in the evaluation section

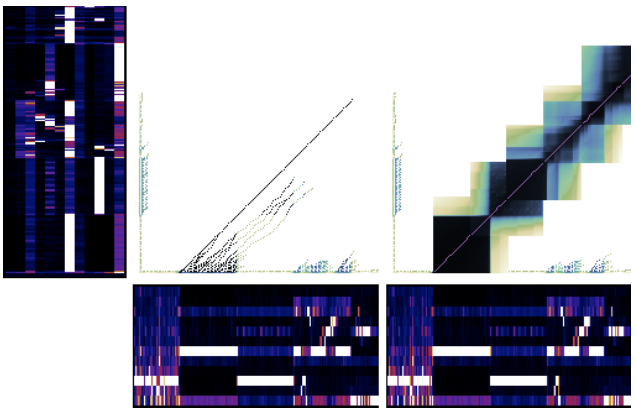

Fig. 2. Sequential DTW method used showing, from left to right: (a) the initial path discovery and (b) the DTW segments

3.4. Real-time Alignment

Once the initial alignment path P_i has been found between the two acoustic signals, we proceed to find the optimum alignment path P from there on, ensuring that the playback of the video remains synchronized throughout the remainder of the song. Initially $P := (p_{i1})$ with total length $W = 1$. This real-time alignment algorithm cannot use standard DTW applied to the full sequences U and V as the future chroma frames are unknown and its computation would have quadratic costs. Instead, a local variation is used that has linear costs .

The algorithm starts at the position where the initial alignment started its forward path ($P_{f1} = P_1$). From that point on two steps are then alternated:

1. A forward path $P_f := (p_{f1}, p_{f2}, ..., p_{fL})$ with length L is made starting at $p_{f1} = p_W$ and ending at position $p_f L$.
2. A standard DTW is made from p_{fL} to p_{f1} to find a backward path P_b, whose first half is appended to P.

In the first step, a forward path P_f is found using the same local constraint as in Sec. 3.3 until L matching elements are found. In our experiments, L is set to 50 frames (5 seconds). The obtained path is a sub-optimal alignment but is useful to obtain an estimate for the end position at distance L. In the first instance of this step, the initially discovered path is used.

In the second step, a conventional DTW path is calculated backwards from p_{fL} to p_{f1}. To do so, the accumulated cost matrix $\mathcal{S}(m, n)$ needs to be computed for $m\epsilon[m_{f_1} : m_{f_L}]$ and $n\epsilon[n_{f_1} : n_{f_L}]$ which is only a small portion of the cost matrix for the entire segments. Here the type I local constraint described by Rabiner and Juang [6] and shown in Figure 1b is used. This results in a backward path of $P_b := (p_{b1}, p_{b2}, ..., p_{bL})$ that contains the optimal alignment between both signals at that time segment. The first half of the backward path, $P'_b := (p_{b1}, p_{b2}, ..., p_{b\frac{1}{2}L})$, is appended to the end of the final alignment path P, resulting in an extended final path with a length of $W = W + \frac{1}{2}L$. This allows subsequent steps to benefit from how the reverse DTW path through accumulated costs can overcome areas of high cost and pick the optimal sub-alignment. Additionally, vertical and horizontal movement is possible, bounded by the guiding forward path, allowing the system to handle a pause in either of the sources.

Another forward path P_f then starts where $p_{f1} = p_{W+\frac{1}{2}L}$ and so on until the end of either source is reached. This real-time iterative alignment can be seen in Figure 2.b. Finally, the alignment path is smoothed to ensure a seamless experience.

3.5. Post Alignment Smoothing

As the rate at which acoustic frames are aligned is 10 times per second and the video playback is usually 25 or 30 frames per second, the obtained path may contain jumps between alignment points. Hence, a post-alignment smoothing is applied in order to reduce these artifacts.

The final path is interpolated so that for any point in the music there is a corresponding time (in milliseconds) where the video should be. In addition, the smoothed path is used to extrapolate the projected estimate of the alignment of the signals and hence allow real-time performance. This estimate is modified every time we compute new alignments.

Every time the video is updated with a new frame, *i.e.* 30 times a second[9], the difference (in milliseconds) between where the video and the audio (music) should be is computed by the projected alignment path, and is equivalent to where the video should be in relation to the audio (*i.e.* + 3200 ms). Then, the time differences are averaged over the last 5 seconds. If the average difference (where the video should be in relation to the audio) differs from the video's actual difference (as known by the media player) by more than 35 ms (or one frame), video frames are skipped or replayed until the correct difference between the video and audio is reached.

This post-processing smoothing is depicted in Fig. 3. The initially computed DTW alignment points, limited by the chroma window's hop size, are represented by dots. These points are interpolated, as shown by the light grey line, to obtain an alignment value for each video frame. Finally, the path is smoothed as indicated by the dark grey line.

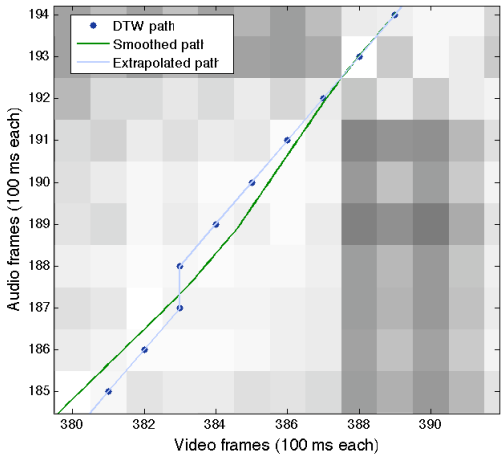

Fig. 3. Post processing on the DTW path

[9]We consider a video sampling rate of 30 fps or 33 ms per frame.

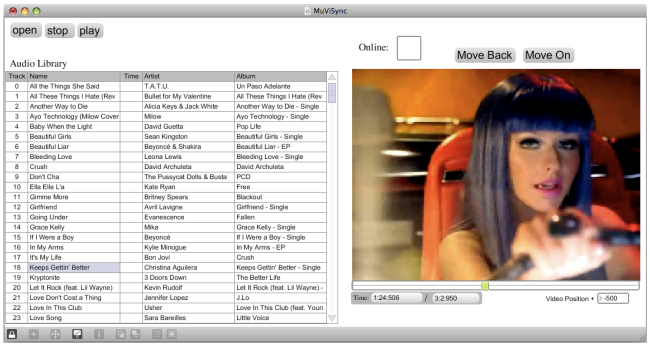

Fig. 4. *MuViSync*'s User Interface

3.6. MuViSync's User Interface

The *MuViSync* prototype was implemented in Java and MAX/MSP[10], using the FFMPEG to convert audio formats and QuickTime to control video playback. Videos can either be in the MP4 format or downloaded directly from YouTube. The audio can be in any format accepted by FFMPEG [11].

Figure 4 shows the graphical user interface. On the left there is a list of the user's songs and on the right is the video playback area. An "Online" check box above the video lets the user specify that the video should be streamed from YouTube. A scroll bar at the bottom allows the user to change the playback position, which the video then follows. In case of errors two buttons, "Move Back" and "Move On", allow users to change the playback location. These buttons restart the initial path discovery method, limited to regions before or after the current alignment.

4. EVALUATION

Evaluating alignment techniques is typically problematic as gathering test data requires hand annotating the alignment. An alternative technique consists of generating matching pairs using MIDI and audio recordings and then modifying one of the pieces with the aim of discovering the same modification during alignment. Both of these techniques suffer drawbacks in being time-consuming or producing easily sync-able test data, respectively. To evaluate the accuracy of *MuViSync*'s synchronisation method, we instead used a supervised standard offline DTW to create a "ground truth" alignment. Matching the accuracy of the standard offline DTW, which has the complete audio information and is guaranteed to find the optimal path, is a requirement of our method.

The test data consists of 320 music videos from YouTube and their corresponding MP3 files bought from Amazon. In order to determine the ground truth alignments, we applied a standard offline DTW method[12] between the audio files and the audio tracks from the videos. This process was manually supervised so that incorrect alignments were discarded (as it would be too time-consuming to correct these) and any non-matching start and end sections were cropped, leaving alignment reference points of only musically matching sections.

To illustrate the differences between the audio pieces to be aligned, Fig. 5.a is a scatter graph showing the durations of the matching pairs in the data-set used. Points away from the diagonal indicate different lengths that are due to alternative starts, endings, and structures. Fig. 5.b depicts the spread of start time differences, between the matched pairs, given by the offline DTW. The values refer to the delay of the video from the audio and are taken at 30 seconds to ensure both media have passed their alternative introductory segments.

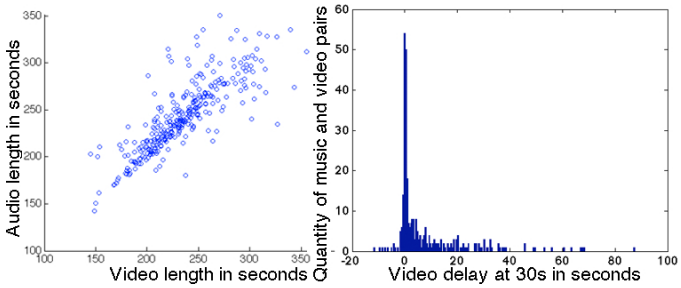

Fig. 5. (a) Comparing durations for the matching audio and video files. (b) . The differences in music start times.

4.1. Initial Path Discovery Tests

We assess the accuracy of the initial path discovery algorithm whilst changing two possible factors of the initial accuracy. An accuracy requirement of 5 audio frames (0.5 seconds) was chosen to measure whether the start was found correctly or not, as this was found to be sufficient to start an accurate alignment. In the first test we evaluated the path discovery process with varying start times within the music file to simulate a user choosing to synchronise at various points after the audio has begun to play. The accuracy of the different start times varied by a maximum of 2% between starting at 0 seconds (92.8%) and starting at 100 seconds (91.5%). Hence, we conclude the initial alignment time has little impact on the performance of the system.

In the second test, we vary the amount of audio information known prior to the start of the alignment. As previously shown, most of the musically equivalent start locations are not located at the beginning of the files (see Fig. 5). However, due to the constraints in streaming YouTube videos online, it is important that the alignment is started *before* all of the content is downloaded. Fig. 6 shows the trade-off between different video buffer lengths used in the initial alignment (X axis), the accuracy of the initial path discovery (dark dashed line) and the time taken to find the initial path (dark solid line). The theoretical maximum accuracy for different buffer lengths (grey dotted line in Fig. 6) is based on how many of the pairs start within any specific buffer length (as in Fig. 5.b). As expected, the start time accuracy decreases as the video buffer length approaches 0, *i.e.*, many videos cannot be initialised correctly as a matching music segment hasn't occurred within the video buffer.

[10] An audio/visual prototyping environment which offers graphical tools and allows for Java modules. http://www.cycling74.com

[11] For all of the FFMPEG formats see: http://ffmpeg.org/general.html

[12] For the offline alignment we used Dan Ellis's algorithm which can be found at http://labrosa.ee.columbia.edu/matlab/dtw/

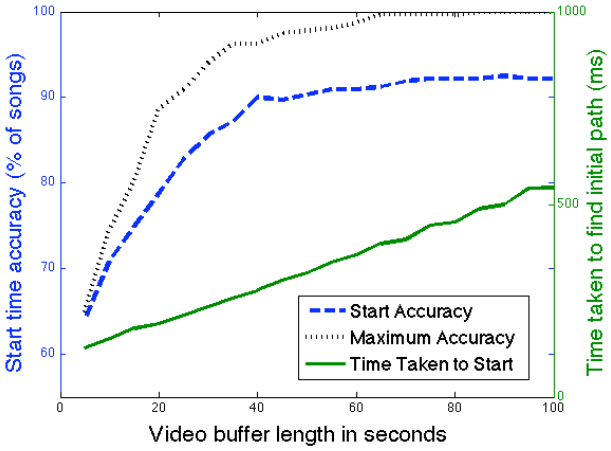

Fig. 6. Accuracy and time versus buffer length

Error $\leq$		Cumulative error counts(frames hit)		
Frames	Seconds	All Pieces	Similar	Different
0	0	52.02%	53.69%	41.26%
1	0.1	90.55%	93.31%	72.81%
2	0.2	93.07%	95.89%	74.94%
3	0.3	93.23%	96.02%	75.27%
5	0.5	93.38%	96.14%	75.63%
10	1	93.54%	96.25%	76.12%
25	2.5	93.86%	96.41%	77.45%

Table 1. Alignment Accuracy results

4.2. Alignment Accuracy

Once the initial alignment is made, the system needs to keep the audio and video in-sync despite any deviations from the current playback rate or differences in the musical structure. In order to test the alignment accuracy, we recorded the path found by *MuViSync* and compared each frame with the corresponding frame of the known (offline) path. We found that the structurally different pieces had a significant effect on the accuracy of the system. Table 1 depicts the results of the overall path alignment accuracy in three categories: all 320 pieces, structurally similar pieces (88% of all pieces) and structurally different pieces. The rows show alternative accuracy requirements for each point of the alignment and the columns show how many of the total points are within the given accuracy requirement (out of $723,000$ reference alignment points). From this test we can see that the number of frames that would be perceived as in-sync (according to a typical user sensitivity of $80 - 100$ ms [17] or 1 frame) is of 93.3**%** for structurally similar pieces and 72.81% for structurally different pieces.

5. CONCLUSIONS AND FUTURE WORK

Today's users are increasingly downloading their digital music collections from online music services and watching music videos of popular artists online (streamed either from free or subscription-based services). To date, however, there are no tools that allow users to enjoy their high fidelity music content while watching *in-sync* the corresponding music videos. In this paper, we have presented the *MuViSync* prototype and described two novel algorithms that can find matching musical starting points and keep the media in-sync. This approach can synchronise music and music videos to within 100 milliseconds of each other over 90% of the time. Future work will include a longitudinal user study and mobile version of the *MuViSync* prototype.

6. REFERENCES

[1] Jinjun Wang, Changsheng Xu, Engsiong Chng, Lingyu Duan, Kongwah Wan, and Qi Tian, "Automatic generation of personalized music sports video," in *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, NY, USA, 2005, pp. 735–744, ACM.

[2] Jonathan Foote, Matthew Cooper, and Andreas Girgensohn, "Creating music videos using automatic media analysis," in *MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia*, NY, USA, 2002, pp. 553–560, ACM.

[3] 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.

[4] 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.

[5] 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.

[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] 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.

[9] Christopher Raphael, "Music plus one: A system for flexible and expressive musical accompaniment," in *Proceedings of the International Computer Music Conference*, Havana, Cuba, 2001.

[10] Min-Yen Kan, Ye Wang, Denny Iskandar, Tin Lay Nwe, and Arun Shenoy, "Lyrically: Automatic synchronization of textual lyrics to acoustic music signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 338–349, 2008.

[11] John Boreczky and Lynn Wilcox, "A hidden markov model framework for video segmentation using audio and image features," 1998.

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

[13] 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.

[14] Frank Kurth, Meinard Muller, David Damm, Christian Fremerey, Andreas Ribbrock, and Michael Clausen, "Syncplayer - an advanced system for multimodal music access," in *ISMIR '05: Proceedings of the 6th International Conference on Music Information Retrieval*, 2005, pp. 381–388.

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

[16] 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.

[17] 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, GLOBECOM'96*, London, UK, 1996, vol. 2, pp. 893–898.