uc3m | Universidad Carlos III de Madrid

Master Degree in Multimedia and Communications
Academic Year 2018-2019

Master Thesis

"Automatic captioning synchronization in theatre"

Author: Diego Carrero Figueroa

Tutor: Ascensión Gallardo Antolín

Leganés, September 2019

AVOID PLAGIARISM

The University uses the Turnitin Feedback Studio program within the Aula Global for the delivery of student work. This program compares the originality of the work delivered by each student with millions of electronic resources and detects those parts of the text that are copied and pasted. Plagiarizing in a TFM is considered a **Serious Misconduct**, and may result in permanent expulsion from the University.

Automatic captioning synchronization in theatre

Diego Carrero Figueroa

Multimedia Processing Group
Signal Theory and Communications Department
University Carlos III Madrid
September 2019

***Abstract*— Theatre has certain features that make real-time subtitling of a performance a much more complex task when compared to audio-visual content like music or cinema. This work describes a system to synchronize the subtitling of a theatrical performance using the audio signal coming from the show and a content database built from previous performances. The underlying idea is to estimate the playback timing in which the current performance is found. For this purpose, techniques such as Automatic Content Recognition (ACR) and image processing over the spectrogram of the audio signal are applied. In a complementary way, the proposed system has been evaluated using audio recordings of a wide variety of sources. So, the influence of the audio quality over the performance of the system have also been studied. The main goal of this work is to lay the groundwork that allows the operation of the system under real-time conditions.**

***Index Terms*— automatic content recognition, subtitling, speech processing, synchronization, theatre**

I. Introduction

In recent years, the irruption of mobile devices has changed the way in which people consume digital content. The *second screen* is a concept referring to the use of mobile devices as a complementary way of information and services that does not directly interrupt the reproduction of the main content [1], [2]. Although the application of the *second screen* is evident in cinema or television, it can also be applied in environments such as theatre to distribute, for example, the subtitling of a play. However, live theatre cannot be considered as a common digital content like cinema or music. While the audio track of a movie is stored into a digital media and therefore, the same information is always played with each new playback, the audio track coming from each performance for a given play presents slight differences over time.

The audio signal coming from contents such as music or cinema is always the same between different playbacks. Here, the audio can suffer alterations mainly related to variation in the playback speed or pitch. However, theatre is a more complex scenario: there are other factors like modifications over the script, substitutions of actors or variations in the cadence when speaking that make the audio signal differs from one performance to another. The playback of subtitles in theatre through a *second screen* needs a synchronization algorithm adaptable to the particular alterations suffered by the audio signal in this environment.

Nowadays, platforms such as ACR Cloud (www.acrcloud.com), Axwave (www.axwave.com) or Mufin (www.mufin.com) offer both recognition and synchronization of audio-visual contents through a *second screen*. Likewise, applications like Shazam (www.shazam.com) or Sound Hound (www.soundhound.com) perform both song identification and lyrics synchronization. All these applications are based on extraction and indexing of acoustic fingerprints from audio sources by means of proprietary algorithms.

However, subtitling synchronization processes in theatre are not yet automated. Software tools like QLab (figure53.com/qlab), STARTIT (www.startit-app.com) or even Power Point (products.office.com/es-us/powerpoint) allow to create the captions for a play. However, synchronized playback of captions during the performance have to be done manually by means of a technician. The need for human intervention translates into high costs and hence, the need of finding a way of automating this process arises.

The research presented in this work is part of the "Teatro Accesible" project (www.teatroaccesible.com). The aims of this project are to provide sensory accessibility to the scenic spaces through the use of new technologies. As a result, more than 360 plays have been adapted in more than 85 theatres in Spain since 2011.

The main goal of this work is the definition of a real-time procedure to synchronize the playback of subtitles

in a theatre play. The key idea is to use an audio recording of a given theatre play in order to build a database which enables the estimation of the timing of a new show using the characterization done from the reference audio.

The rest of this paper is organized as follows. Section 2 presents a study of the state of the art concerning audio synchronization. Next, section 3 contains a detailed description of the proposed system. Section 4 refers to the experimental methodology adopted to evaluate the system and shows the results. Finally, in section 5, the main conclusions are discussed and the lines of future work are detailed.

II. Related Work

Nowadays several approaches can be found to address the synchronization of various audio sources. Generally, the main algorithms can be sorted in three different families: correlation-based, DTW-based and audio fingerprint-based algorithms. Acoustic fingerprint-based algorithms are used mainly in Automatic Content Recognition applications and multimedia synchronization processes. In this context, there are several works related to synchronization and timing estimation in movies; however, captioning synchronization in an environment like theatre is a barely explored field.

A. Correlation

This first approach is a very simple algorithm: given two audio signals, a measure of similarity is computed using some kind of distance function, typically the cross-correlation between them [3], which is applied while moving one signal over the other. The alignment is immediate done once the minimal distance is found. However, this procedure is very slow given its high computational load and thus, its application is not suitable in real-time applications.

B. Dynamic Time Warping (DTW)

DTW algorithms [4] are based on the alignment of audio signals using dynamic programming techniques over a specific feature space. For this, a distance function d is defined in order to determine the similarity between the samples of both signals. Also, a set of restrictions can be defined for setting up the process of alignment.

In general, the alignment of two audio signals involves the following operations:

1. The two signals are divided into fragments.
2. Each fragment is split into windowed frames of length ranging from 20 to 40 ms, according to the application.
3. A feature vector is computed for every frame.
4. A rectangular matrix M is constructed for storing the distance between the feature vectors of both fragments. In this way, $M(i,j)$ measures the distance between the feature vector i of the first fragment and the feature vector j of the second one.
5. The matrix is populated using the distance function over the feature vectors of the two fragments.
6. The alignment is done by finding the path in the matrix satisfying the set of restrictions with the minimum cost. This cost is computed as the sum of absolutes differences for each matched pair of indexes (i,j). The algorithm starts at the top left of the matrix and ends at the bottom right. In this process, two samples are aligned when the algorithm moves diagonally through the matrix. However, there is a shift between them when the algorithm moves down or right.

Figure 1 shows the distance matrix for two aligned signals x and y. The distance between both signals can be computed by summing all the costs over the highlighted alignment path: for the given example, the distance is 7.

	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]	x[8]
y[1]	0	1	1	2	1	5	3	2
y[2]	1	2	0	1	0	6	4	1
y[3]	2	3	1	0	1	7	5	0
y[4]	0	1	1	2	1	5	3	2
y[5]	5	4	6	7	6	0	2	7
y[6]	0	1	1	2	1	5	3	2
y[7]	1	2	0	1	0	6	4	1
y[8]	2	3	1	0	1	7	5	0

Figure 1. Distance matrix in DTW alignment.

Although the basis of DTW is simple, it is necessary to consider some aspects. First, the full knowledge of the signals to align is presupposed in an optimal scenario; however, the signals to be aligned are generated in fragments in a real-time application. Windowed Time Warping (WTW) [5] is a variation of DTW to address applications in real time. In this approach, the alignment path is divided into windowed DTW series for calculating small sub-alignments and combining them to form an overall path.

Next, it is important to determine the set of features to be extracted from the audio signal. An approach for tracking musical performances is to represent the audio data by the positive spectral difference between successive 20 milliseconds frames [6] [7].

Last, another key factor to consider is the computational complexity: the greater the length of the signals to be aligned the greater the complexity. With approaches such as Unbounded-DTW (UDTW) the

computational load is reduced and multiple paths can be located simultaneously [8]. This algorithm has three main characteristics: 1) there is not any restriction on the start and end positions of the audio patterns, 2) more than one matching segment may be found with a single pass and finally, 3) the used search algorithm avoids unnecessary comparisons.

C. Fingerprinting

A fingerprint is a compact representation, in form of content-based summary, that uniquely represents an audio signal [9]. The key point of this family of algorithms is the processing of the audio spectrogram in order to build a set of fingerprints which allow unequivocally identify the signal based on a given fragment. The scheme of operation is the following:

1. Given the spectrogram of an audio track, a set of fingerprints is built by finding representative points in the spectrogram. In addition to this, the information about time and frequency of the source point is attached to every generated fingerprint.
2. The set of fingerprints and the information about the audio track are stored in a hash table. In this way, indexing and searching processes are as efficient as possible.
3. In the matching phase and given a short audio fragment, a set of fingerprints is extracted and used as query to perform a search in the hash table. The name of the indexed audio and the instant of time for the fragment can be inferred from the results given by the query.

An acoustic fingerprint extraction system must offer high discrimination against a large number of fingerprints, ease of storage, scalability on large databases and robustness against distortions and environmental interferences [10].

The extraction of acoustic fingerprints is mainly used in tasks related to Automatic Content Recognition (ACR) like music identification. In this sense, there are different approaches to obtain the fingerprints. In a first approach, the spectrogram can be processed as an image using SIFT descriptors [10], [9] or nested cosine filters [11]. The spectrogram can also be processed to obtain local energy peaks whose time-frequency coordinates are indexed as landmark features [12]. Other forms of processing apply functions, called Constant Q Transforms (CQT), on the spectrogram to generate time and frequency representations with a constant number of bins for each frequency octave [13]. As last example, the spectrogram can also be filtered by two two-dimensional filters to generate features called spectral peaks [14]. In this approach, a max filter is first used to find coordinates of candidates and then a min filter is used to reject peaks extracted from uniform regions in the spectrogram. In complementary fashion, fingerprints can be built combining some of the techniques presented above. In [15] the local maximums in time-frequency are located and combined in triplets. Then, CQT is applied to ensure invariance against speed variations and pitch shift.

III. System Description

The description of the proposed synchronization system is presented throughout this section. In general terms, the estimation of the timing can be addressed as a problem of Automatic Content Recognition; however, compared to a standard ACR problem, it is not about identifying a multimedia content but a fragment of the theatre play. In this context, hundreds of audio fragments coming from the same performance play are required to build a database of audio descriptors.

In general, the duration of a theatre play varies between 90 and 120 minutes and contains between 1,500 and 2,500 captions approximately with a length ranging from 2 to 5 seconds. The number of captions depends largely on the duration and rhythm of the play. In a complementary way, the maximum jitter between voice and subtitle playback is around one second when manual synchronization is used.

As shown in Figure 2, the proposed procedure takes an audio recording of a given theatre play to build a database enabling the estimation of the playback timing for a new performance using the characterization done from the reference audio.

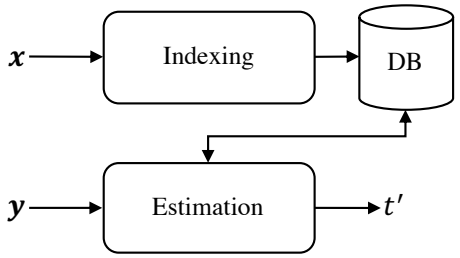

Figure 2. Overview of the process.

Given an audio frame $\boldsymbol{y}$ coming from a new performance, similar fragments can be retrieved from the database in order to estimate the playback timing t'. Thus, the playback of the captions can be synchronized once the timing is inferred. In rough outlines, the synchronization process involves three fundamental tasks:

1. Characterization of the spectrogram of the audio signal through the Scale Invariant Feature Transform (SIFT).
2. Indexing and searching by mean of Locality Sensitive Hashing (LSH).
3. Estimation of the timing using timestamps associated to the SIFT descriptors.

In a first stage, a database containing the indexed audio fragments of the play is built. During this process, time and frequency metadata are added to help to locate

the fragment within the original audio stream. In a second stage, this metadata is used to estimate the timing when a content recognition process is executed from an audio fragment.

A. Audio characterization

Prior to index the audio, a feature space has to be defined in order to enable the characterization of every fragment and measure the similarity between them. The feature space used to characterize the acoustic fingerprints must be sufficiently robust to deal with the variations existing between the audio coming from different performances of a play. Some approaches to the extraction of acoustic fingerprints take the process like an artificial vision problem. Following this scheme, the process adopted in this work greatly follows the procedure presented in [10]. Thus, the use of SIFT descriptors over the image of the spectrogram is adopted to generate acoustic fingerprints.

1) Audio segmentation

The starting point of the fingerprint extraction process is the audio stream sampled at 48 KHz coming from the recording of a play. This first audio file sets the reference clock used to later estimate the playback timing. As shown in Figure 3, the input stream $\boldsymbol{x}$ is split into K non-overlapping frames of 60 seconds length; thus, the audio frame $\boldsymbol{s}_k$ has $t_s = 60(k-1)$ and $t_e = 60k$ as beginning and ending instants, respectively. In these expressions, k is an integer ranging from 1 to K.

Figure 3. Audio segmentation diagram.

The process is defined for a mono audio stream and therefore, a conversion to a single audio channel is needed when the input stream has more than one.

2) Image generation

A visual representation for the audio frame $\boldsymbol{s}_k$ is needed before extract the fingerprints by means of the SIFT transform. Figure 4 shows the set of operations performed over the frame in order to obtain the image of the spectrogram.

In a first step, Short-Time Fourier Transform (STFT) is performed over the frame $\boldsymbol{s}_k$ to get its linear spectrogram S_k. For every frame, the module of a STFT is obtained using Hanning frames of 8192 samples (170,67 ms) with a three-fourth overlap. In a complementary way, two additional vectors are obtained for every spectrogram. These vectors will enable the building of metadata for the acoustic fingerprints:

- $\boldsymbol{t}_k$: contains the absolute time stamps associated to the audio stream for each column of the spectrogram.
- $\boldsymbol{f}_k$: contains the central frequencies of the bands of the linear spectrogram.

The lengths of $\boldsymbol{t}_k$ and $\boldsymbol{f}_k$ are 1403 and 4096 respectively, equal to the number of columns and rows of S_k.

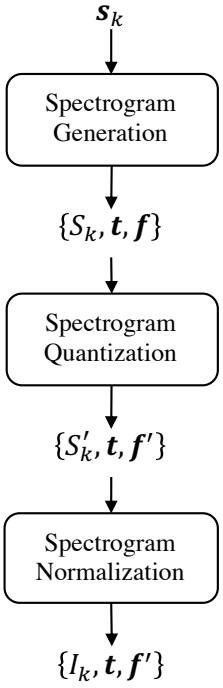

Figure 4. Spectrogram image generation diagram.

After that, the linear spectrogram S_k is quantized into 88 logarithmically spaced frequency sub-bands to build S'_k. This change of scale is made using the Twelve-Tone Equal Temperament (12-TET) tuning system. Thus, and starting at some initial frequency f_0, each frequency octave is divided into twelve equal parts (semitones) [16]. The central frequency b_m of the band m in S'_k is determined according to the next expression:

$$b_m = f_0 2^{\left(\frac{m-1}{12}\right)} \tag{1}$$

In the above expression, f_0 defines the central frequency of the lowest quantization sub-band. Given that 88 sub-bands are defined and f_0 is 80 Hz, the upper limit of S'_k is set on 12,177 KHz. This wide range of frequencies enables the characterization of both speech and music contents. For a given band m with a central frequency b, the lower and upper bounds can be computed using these expressions, respectively:

$$b_{min} = b - \frac{b}{2}\left(1 - 2^{\left(\frac{-1}{12}\right)}\right) \tag{2}$$

$$b_{max} = b + \frac{b}{2}\left(2^{\left(\frac{1}{12}\right)} - 1\right) \tag{3}$$

In the same manner, vector $\boldsymbol{f}$ is also quantized into $\boldsymbol{f'}$ to reflect the change in the frequency scale.

Finally, the quantized spectrogram is converted to a grayscale image. At this point, the magnitude of S'_k is first converted to a logarithmic scale using the expression (4). This operation enhances the details of areas with small magnitude values and improves the extraction of acoustic fingerprints.

$$S''_k = log(|S'_k|) \tag{4}$$

Finally, a 256-levels grayscale image I_k of the spectrogram is computed by means of the following expression:

$$I_k = \frac{255\left(S''_k - min(S''_k)\right)}{max(S''_k) - min(S''_k)} \tag{5}$$

In the above expression, $min(S''_k)$ and $max(S''_k)$ are the minimum and maximum values of S''_k.

3) Fingerprint extraction

The acoustic fingerprint F_k from the visual characterization I_k of the audio frame $\boldsymbol{s}_k$ is computed following the operations presented in Figure 5. In this way, the acoustic fingerprint is defined by the set of J SIFT normalized descriptors and theirs time-frequency locations as show in the expression (6):

$$F_k = \left\{\boldsymbol{d}'_j, t_j, f'_j\right\}_{j=1}^{J} \tag{6}$$

In a first stage, SIFT descriptors [17] are extracted from I_k in order to characterize the spectrogram. The SIFT transform is a procedure which enables the extraction of robust and invariant features from images for recognition or classification in a wide range of computer vision applications. The process of finding the SIFT descriptors involves four major steps:

1. Scale-space extrema detection. The first step of key point detection is to identify locations and scales that can be repeatably assigned under different views. Thus, the algorithm detects key points that are invariant to scale and orientation using a cascade filtering approach by means of a Difference-of-Gaussian function [18]. This filtering performs a convolution between the image and the Difference-of-Gaussian function, which can be efficiently computed from the difference of two nearby scales separated by a constant multiplicative factor. The detection of local maxima and minima is performed by comparing each point value with its eight neighbours in the image and nine neighbours in the scales both above and below.
2. Key point localization. Key points are selected based on measures of their stability: points localized along edges or having low contrast are rejected. For this purpose, a detailed model is fit to the nearby data for location, scale and ratio of principal curvatures.
3. Orientation assignment. One or more orientations are assigned to each key point location based on local image gradient directions to prevent the invariance regarding the rotation of the image. The computation of orientation is based on a histogram of the directions weighted by the gradient magnitudes. The orientation of the key point is given by the largest peak in the histogram. Likewise, other peaks above 80% of the most important are used to create other key points in the same position and scale but with different orientation.
4. Key point description. For each key point, the local image gradients are measured in a 16x16 points neighbourhood. This neighbourhood, in turn, is divided into 4x4 points sub-blocks and an 8-bin histogram of orientations is created for each one. After all, two output vectors are generated: on the one hand, a 128-element feature vector $\boldsymbol{d}_j$ containing the descriptor and on the other hand, a two-dimensional vector $\boldsymbol{p}_j$ containing the location point of $\boldsymbol{d}_j$ within I_k.

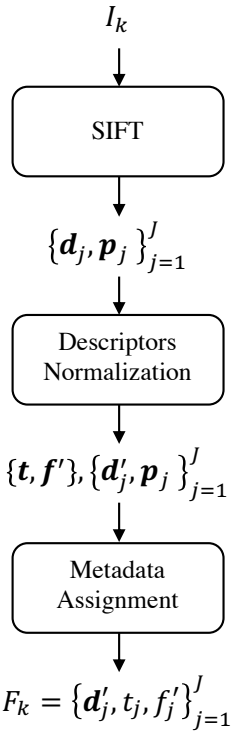

Figure 5. Fingerprint extraction diagram.

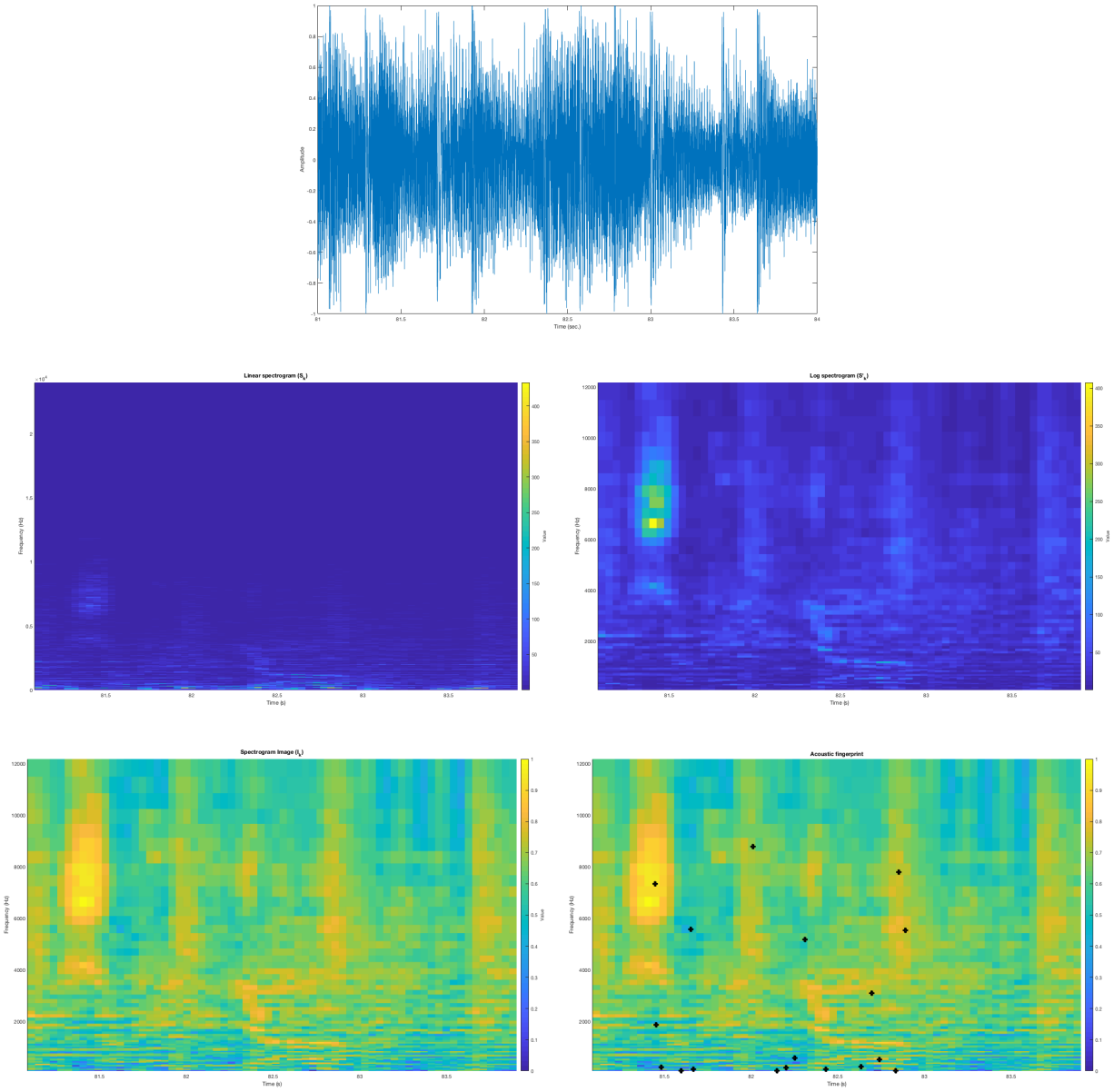

Figure 6. Results obtained after every step of the process: audio frame (top), linear spectrogram (center left), quantized spectrogram (center right), normalized spectrogram (bottom left) and SIFT descriptors on normalized spectrogram (bottom right).

The extraction of SIFT descriptors has been performed by means of *VLFeat* library for Matlab (www.vlfeat.org/overview/sift.html). VLFeat is an open source library which implements the most popular computer vision algorithms including SIFT transform.

After the application of SIFT transform, L2-normalization is applied on every descriptor $\boldsymbol{d}_j$ as shown in the next expression:

$$\boldsymbol{d}'_j = \frac{\boldsymbol{d}_j}{\|\boldsymbol{d}_j\|} \tag{7}$$

The last step in the generation of the acoustic fingerprint is the assignment of metadata to every normalized descriptor $\boldsymbol{d}'_j$.

By using the metadata information is possible to get both the frequency and the precise instant of time of the original audio stream in which the descriptor occurs. Thus, metadata are obtained from vector $\boldsymbol{p}_j$ as described in expression (8).

$$\begin{aligned} t_j &= \boldsymbol{t}\left[p_j[1]\right] \\ f'_j &= \boldsymbol{f}'\left[p_j[2]\right] \end{aligned} \tag{8}$$

In previous page, Figure 6 shows the results obtained when processing the audio frame to generate the acoustic fingerprint.

B. Indexing and searching

Once the mechanism of acoustic characterization has

been presented, the next paragraphs show the procedure to index and retrieve the descriptors in an efficient way. Given the large number of acoustic fingerprints extracted for a play, a mechanism to optimize the searching time is needed because the using of brute force is highly inefficient. In this sense, Locality Sensitive Hashing (LSH) [19] [20] becomes an efficient approximation to nearest neighbour search algorithms. As shown in Figure 7, the key idea behind this family of algorithms is the following: two similar points $\boldsymbol{p}$ and $\boldsymbol{q}$ in an input space can be hashed into the same bin with high probability.

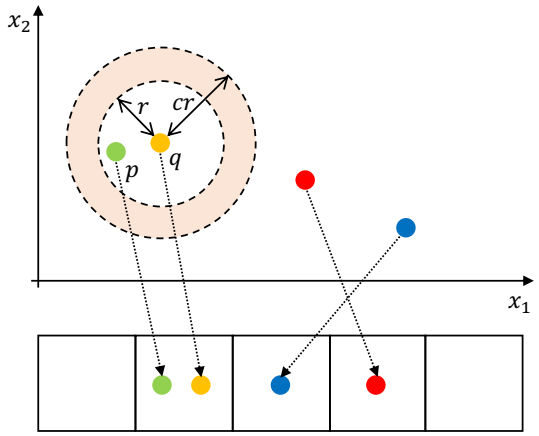

Figure 7. Diagram showing the key idea behind LSH.

LSH proposes the use of a family of hash functions $\mathcal{G}$ of the form $g: R^d \rightarrow U$. So, for any pair of points $\boldsymbol{p}$ and $\boldsymbol{q}$, the probability of $g(\boldsymbol{p})$ being equals to $g(\boldsymbol{q})$ is considerably high if $\|\boldsymbol{p}-\boldsymbol{q}\| < r$. Furthermore, when $\|\boldsymbol{p}-\boldsymbol{q}\| > cr$ the probability of $g(\boldsymbol{p})$ being equals to $g(\boldsymbol{q})$ will be small. The construction of $\mathcal{G}$ is based on another family of functions $\mathcal{H}$ in which any function h has the following form:

$$h(\boldsymbol{p}) = \left\lfloor \frac{\boldsymbol{a}^T \boldsymbol{p} + b}{w} \right\rfloor \tag{9}$$

In the above expression, $\boldsymbol{a}$ is a 128-element random projection vector whose elements are chosen independently from a Gaussian distribution $N(0,1)$, b is a random shift chosen uniformly from $U(0, w)$ and w is the width of intervals for random projection lines estimated from the range of the input data.

LSH defines two main parameters: the hash width k and the number L of hash tables. In a first step, the family of hash functions $\mathcal{G}$ is defined to contain L functions. Every function $g(\boldsymbol{p})$ is obtained as the random concatenation of k functions $h(\boldsymbol{p})$ belonging to $\mathcal{H}$ as shown below:

$$g(\boldsymbol{p}) = [h_1(\boldsymbol{p}), \ldots, h_k(\boldsymbol{p})] \tag{10}$$

Next, a hash table is built for every generated function $g(\boldsymbol{p})$. While indexing, each one of the L hash tables is fed with all the descriptors extracted from the audio frame. The bucket in which a descriptor is placed for a hash table L_i is determined by the value of the corresponding hash function $g_i(\boldsymbol{p})$.

In the search stage, a set of descriptors $Q = \{\boldsymbol{q}_m, t'_m, f'_m\}_{m=1}^{M}$ is first extracted for a given audio fragment $\boldsymbol{y}$ as described above. Now, t'_m in Q denotes the instant of time of the descriptor $\boldsymbol{q}_m$ with respect to the origin of $\boldsymbol{y}$. Next, for a query descriptor $\boldsymbol{q}_m$ in Q, LSH iterates over the L hash tables and retrieves the descriptors located into the same bucket as $\boldsymbol{q}_m$ by evaluating $g_1(\boldsymbol{q}_m), \ldots, g_L(\boldsymbol{q}_m)$ until either the descriptors from all L buckets are retrieved or the total number of descriptors retrieved exceeds $2L$. The resulting set P_m of descriptors has the smallest Euclidean distance with $\boldsymbol{q}_m$.

In practice, the implementation of the database considers the particular characteristics of a theatre play:

- Always starts at timing 0.
- Moves forward in small steps.
- Moves backwards or makes great leaps with very low probability.

With these premises in mind, instead of using a single global LSH index to store all descriptors, K partial LSH indexes has been built: one for each audio fragment $\boldsymbol{s}_k$ (60 seconds) used in the indexing stage. While indexing, each of the LSH indexes is labelled with start and end timestamps. Later in the search process, only those indexes located in the temporal environment of the last timing estimation t' are used to estimate the new timing. The next figure illustrates this process:

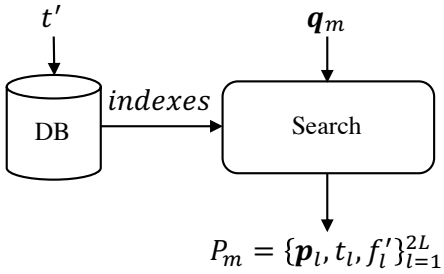

Figure 8. Searching process for a query descriptor.

The procedure to determine the indexes in which to perform the search is the following:

1. Indexes whose timestamps differ less than two minutes from the last estimation of timing are obtained from database.
2. The search for descriptors is done.
3. The new playback timing is estimated.
4. The search timestamp is updated based on the difference between the last timing estimation and the new one:

- The difference is less than 1.5 times the duration of $\boldsymbol{y}$: the search timestamp is updated to the new timing estimation.
- The difference is greater than 1.5 times the duration of $\boldsymbol{y}$: the search timestamp is not updated and the search range is doubled.

The adoption of this approach showed better performance than the use of a single global LSH index in the preliminary tests.

For the implementation of the LSH algorithm, the *lshcode* library for Matlab (ttic.uchicago.edu/~gregory/download.html) has been used. This toolbox is a simple implementation of LSH algorithm.

C. Timing estimation

The set $Q = \{\boldsymbol{q}_m, t'_m, f'_m\}_{m=1}^{M}$ of normalized descriptors extracted from audio query $\boldsymbol{y}$ is the starting point of the timing estimation. The goal is, therefore, to determine which is the timing t' corresponding to the beginning of $\boldsymbol{y}$ with respect to the whole audio of the theatre play.

In a first stage, a search is performed over the hash tables for each descriptor $\boldsymbol{q}_m$ in order to retrieve a set P_m containing its nearest neighbours as described in the previous section.

Prior to estimation, retrieved descriptors are filtered: results located outside the 3-band range with respect to the query descriptor $\boldsymbol{q}_m$ are removed. In this sense, the frequency f'_m at which the query appears is taken as central frequency and the lower and the upper limits of the range are calculated according to expressions (2) and (3). All those results $\boldsymbol{p}_l$ whose frequencies f'_l are out of range are automatically removed. By means of this approach of filtering in frequency, descriptors located in other frequency bands which are not relevant are removed whereas some variability in the pitch of the voice is allowed.

Next, the descriptor $\boldsymbol{p}_l$ having a smaller Euclidean distance with respect to $\boldsymbol{q}_m$ is chosen as a result and its timestamp t_l is taken as timing reference t_m. Any query descriptor which does not yield results, either because it is not found in the tables or all its candidates are discarded, is removed from the estimation process. At this point, the set of timestamps $T = \{t_m\}_{m=1}^{M}$ is built from the metadata of the resulting descriptors.

The set T is processed to eliminate those values that may cause a deviation in the estimation. For this, the median of the set is calculated and then, timestamps whose absolute difference with the median is greater than 0.5 times the duration of $\boldsymbol{y}$ are discarded. In addition, the minimum number M_{min} of remaining timestamps needed to estimate the playback timing depends too on the length of $\boldsymbol{y}$ and is given by the following expression:

$$M_{min} = 2 + 0.5l \tag{11}$$

In the above expression, l is the duration of $\boldsymbol{y}$ expressed in seconds. Finally, the timing associated to the beginning of the audio fragment $\boldsymbol{y}$ can be estimated by the following expression:

$$t' = \frac{1}{M}\sum_{m=1}^{M}(t_m - t'_m) \tag{12}$$

In the above expression, M is the number of descriptors remaining after the process.

D. Real-time operation

In the next paragraphs the procedure of applying the algorithm in a real-time scenario is described. Although the algorithm presented in the previous section is not strictly real-time since it is frame-based, the synchronization process can be periodically executed throughout the performance in order to update a master clock which controls the playback of captions or any other multimedia content.

In this sense, given the audio stream of a new performance, frames of 10 seconds duration are captured and processed every 15 seconds. When setting the frame size, a trade-off is needed between the quality and the frequency of update of the timing estimation. On the one hand, the shorter the frame length the fewer descriptors and the lower the probability of estimating the timing for the frame. As an advantage, the timing will be updated more frequently. On the other hand, the longer the length the greater the number of descriptor and hence the greater the probability of finding the timing for the frame. In return, the timing will be updated less frequently.

The following figure describes the process of timing estimation in real-time.

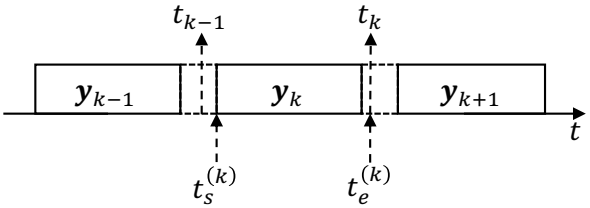

Figure 9. Real-time operation.

Given a master clock establishing the temporary reference, the estimation process for an audio frame $\boldsymbol{y}_k$ can be done using the following data:

- $t_s^{(k)}$: Absolute timestamp in which the capture of the audio frame begins.
- $t_e^{(k)}$: Absolute timestamp in which the timing estimation is obtained.

- t'_k: Timing estimation for the beginning of the audio frame with respect to the audio indexed into the database.
- t_k: Timing estimation for the performance currently in progress.

Hence, the value of t_k can be obtained by means of the following expression:

$$t_k = t'_k - \left(t_e^{(k)} - t_s^{(k)}\right) \quad (13)$$

If the process fails in estimation, the timing is updated by simply adding the time elapsed since the previous timing as follows:

$$t_k = t_{k-1} - \left(t_e^{(k)} - t_e^{(k-1)}\right) \quad (14)$$

IV. EXPERIMENTS

Throughout this section, the methodology adopted to perform the evaluation of the synchronization system is described. First, the audio database is presented, next, the works of labelling and audio processing are detailed and, finally, experiments are described.

A. Database

Given the limited availability of data and the lack of existing databases, an ad-hoc database was built to carry out the evaluation of the system. The database was built from different audio recordings of the play "Cinco horas con Mario" by Miguel Delibes, made during several days of July 2019 at the "Teatro Bellas Artes" in Madrid.

For every recording session, the audio was captured in three different ways:

- Source 1: Omni-directional head microphones (Sennheiser HSP 2 EW 3) were used to obtain a direct audio signal from the actors and thus, minimize the effect of the environment over the recording. This is the audio source taken as baseline in the experiments.
- Source 2: In this case, the audio was taken by means of a hyper cardioid microphone (Rode NTG 2) placed at the first floor.
- Source 3: Finally, the recording was made using a stereo omnidirectional recorder (Zoom H1N) placed in the same location that source 2.

The database consists of a total of 15 audio recordings: three audio sources of equal duration for each one of the five sessions. On the one hand, the audio recordings belonging to the first day (the oldest) have been used as a training set. On the other hand, the files of the remaining four days have been used as testing set.

B. Labelling and pre-processing

The comparison of the timing between testing and training files becomes into a need at the moment of evaluating the performance of the system. While files of the training set remained intact, testing files were split into consecutive non-overlapping fragments of 10 seconds duration. The following table shows the number of fragments obtained for the audio files in each of the test sessions. As exposed, the duration of the show varies slightly from one session to another.

Table 1. Number of audio fragments in the test sessions.

Test session	001	002	003	004
# of fragments	481	480	477	475

The labelling task was done using the software Logic Pro X (https://www.apple.com/uk/logic-pro/). By means of two audio tracks, each test fragment was manually placed at the equivalent timing location of the training file and MIDI markers were added in each timing location. After this, the whole set of markers was processed in order to obtain the equivalence in seconds of each test fragment with respect to the training audio.

C. Experiments

Throughout this section the methodology adopted and the results obtained in the experiments are described. As starting point, the training file corresponding to audio source 1 was indexed in fragments of 60 seconds. In this way, around 46,000 descriptors were indexed in a database built with 78 LSH indexes.

In order to measure the performance, the typical parameters for evaluating information retrieval systems were used:

- Recall: Fraction of fragments for which the system generated a timing estimation.
- Precision: Fraction of recovered fragments whose absolute difference between the estimation and actual timing value is less than one second.
- Accuracy: Fraction of fragments with respect to the total whose difference between the estimation and actual timing value is less than one second. Two values for accuracy are shown in result tables: first, accuracy concerning only to the LSH algorithm and next, the accuracy of the timing estimation.

A second has been established as an absolute margin to determine the correct synchronization of an audio fragment since this value is an acceptable gap between voice and caption playback.

In a complementary way, given the random nature of the initialization of LSH, every experiment was performed five times. Thus, results presented in the next paragraphs are the average of the different executions.

1) Influence of LSH parameters

First, the evaluation of the system for different configurations of LSH is presented. The performance of the system has been explored for different combinations of parameters L and k in the indexing and synchronization of audios belonging to source 1 using fragments of 10 seconds length as query.

Generally, the accuracy of LSH is around 75% and the only scenario in which there is a loss of performance is when increasing k for $L = 2$. On the other hand, the fact of assuming that elapsed time from last query fragment is equal to the length of the current fragment when timing estimation cannot be done (use of equation 14 when there is no timing estimation) improves the accuracy of LSH by around 8-10% according to the configuration adopted. Table 2 shows the results.

Table 2. System performance for different LSH parameters.

L	k	Recall	Precision	LSH Accuracy	Timing Accuracy
2	1	0,913	0,840	0,767	0,839
2	2	0,907	0,842	0,764	0,841
2	4	0,878	0,836	0,735	0,830
2	8	0,762	0,726	0,566	0,677
4	1	0,915	0,837	0,766	0,838
4	2	0,915	0,839	0,768	0,840
4	4	0,914	0,835	0,763	0,832
4	8	0,880	0,831	0,731	0,822
8	1	0,916	0,837	0,766	0,838
8	2	0,916	0,837	0,766	0,838
8	4	0,917	0,836	0,766	0,837
8	8	0,909	0,845	0,769	0,842
16	1	0,916	0,837	0,766	0,838
16	2	0,916	0,837	0,766	0,838
16	4	0,916	0,837	0,766	0,838
16	8	0,918	0,836	0,767	0,836

Figure 10 shows the accuracy of timing according to the configurations presented in the previous table.

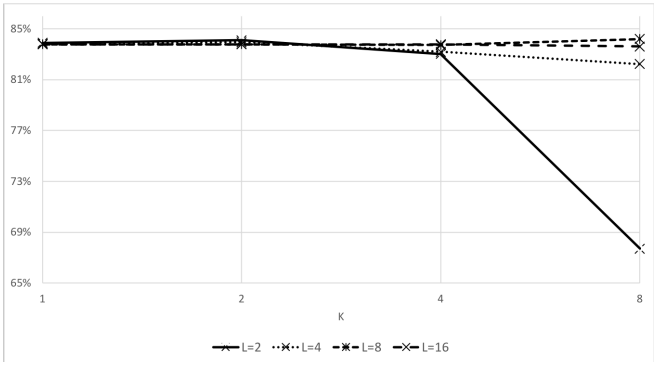

Figure 10. Timing accuracy.

In addition, the difference between configurations is minimal and, in most of them, the accuracy of the system is above 83%. Thereby, 84.1% and 84.2% of the fragments are synchronized with an absolute error of less than one second for $\{L = 2, k = 2\}$ and $\{L = 8, k = 8\}$ configurations respectively.

2) Influence of query length

Results obtained after evaluating the performance of the system using audio queries of different lengths are presented in the Table 3. In these experiments, LSH configurations $\{L = 2, k = 2\}$ and $\{L = 8, k = 8\}$ have been used since they have offered the best accuracy as shown in previous section.

Table 3. System performance for different fragment lengths.

LSH	T	Recall	Precision	LSH Accuracy	Timing Accuracy
2-2	10	0,907	0,842	0,764	0,841
2-2	8	0,750	0,875	0,657	0,807
2-2	6	0,536	0,933	0,502	0,755
2-2	4	0,328	0,986	0,324	0,792
8-8	10	0,909	0,845	0,769	0,842
8-8	8	0,766	0,888	0,681	0,831
8-8	6	0,534	0,934	0,501	0,756
8-8	4	0,335	0,989	0,332	0,786

In the experiments, audio fragments of length ranging from 10 to 4 seconds in 2-seconds steps have been tested. Looking at the search performance, the accuracy of LSH dramatically decreases as the query length is shorter. For both configurations, the performance of the search algorithm is reduced by 50% when fragments of 4 seconds are used. On the other hand, and considering the timing accuracy, the performance obtained with the 10-seconds fragments is never reached.

The following figure shows the results obtained for each query length under test.

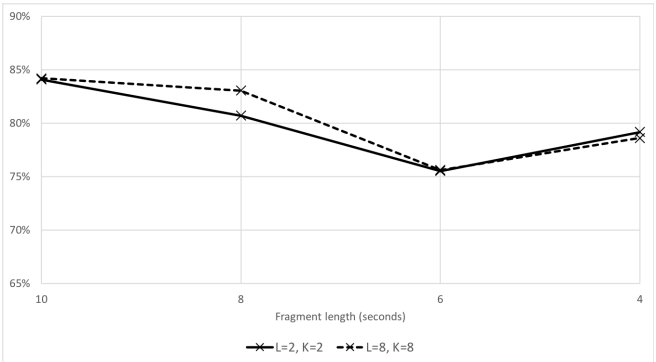

Figure 11. Timing accuracy vs. fragment length.

3) Influence of audio sources

In a third group of experiments, the performance of the system has been evaluated according to the acoustic characteristics of the audio sources used to make the synchronization.

In a first experiment, the training audio from source 1 has been used to index the database and the synchronization has been made with the testing audios from sources 1 (baseline), 2 and 3. In the following table, the results are presented and compared with the baseline obtained for configurations $\{L = 2, k = 2\}$ and

$\{L = 8, k = 8\}$.

Table 4. System performance for different audio sources.

LSH	Source	Recall	Precision	LSH Accuracy	Timing Accuracy
2-2	1	0,907	0,842	0,764	0,841
2-2	2	0,278	0,704	0,203	0,471
2-2	3	0,444	0,853	0,380	0,749
8-8	1	0,909	0,845	0,769	0,842
8-8	2	0,287	0,727	0,214	0,490
8-8	3	0,376	0,790	0,300	0,584

The existence of a strong situation of dependence between SIFT descriptors and acoustic characteristics of the source can be observed. In addition, search capabilities degrade significantly when the source used to estimate is different from the one used for the indexing.

Figure 12 shows the comparison of the different results obtained.

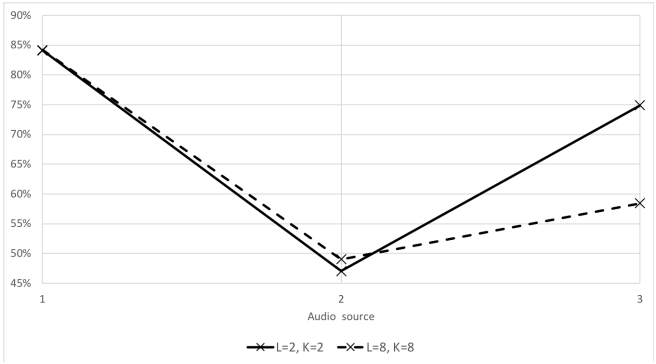

Figure 12. Timing accuracy vs. audio sources.

In a second experiment, indexing and estimation has been done with the training and testing audios of the same audio source. Table 5 shows the results of this second experiment and their comparison to the baseline with configurations $\{L = 2, k = 2\}$ and $\{L = 8, k = 8\}$.

Table 5. System accuracy when testing and training audios come from the same source.

LSH	Source	Recall	Precision	LSH Accuracy	Timing Accuracy
2-2	1	0,907	0,842	0,764	0,841
2-2	2	0,764	0,736	0,561	0,712
2-2	3	0,784	0,830	0,652	0,792
8-8	1	0,909	0,845	0,769	0,842
8-8	2	0,770	0,754	0,579	0,737
8-8	3	0,802	0,849	0,681	0,827

In this second case, the loss of performance of LSH is not as high as in the previous experiment. While the accuracy is reduced by more than 50% in the previous case, now in the worst case, the reduction only reaches 20%. In addition, the timing accuracy of the system keeps close to the achieved by the baseline: the accuracy for audio source 3 is only 5% below the baseline.

Figure 13 shows the results for this second case.

A clear conclusion appears after performing this set of experiments. When the limitation of acquiring the audio from a location placed far away from the stage is present, the stereo omnidirectional microphone offers better performance than the hypercardioid one, possibly due to the better coverage of all the stage areas by the omnidirectional microphone.

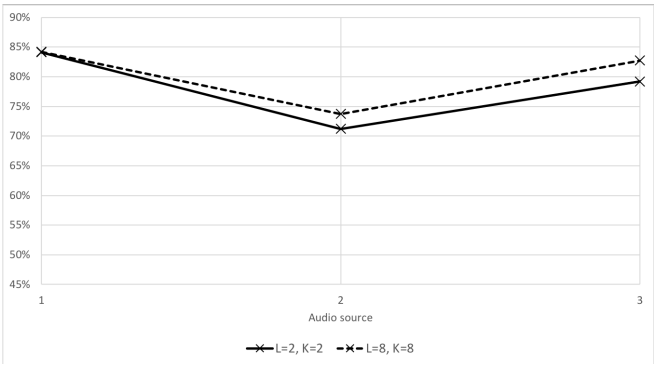

Figure 13. Timing accuracy when testing and training audios come from the same source.

4) Indexing performance

Finally, the following figures show some performance measures of the database. Indexing times and sizes are presented depending on the parameters L and k.

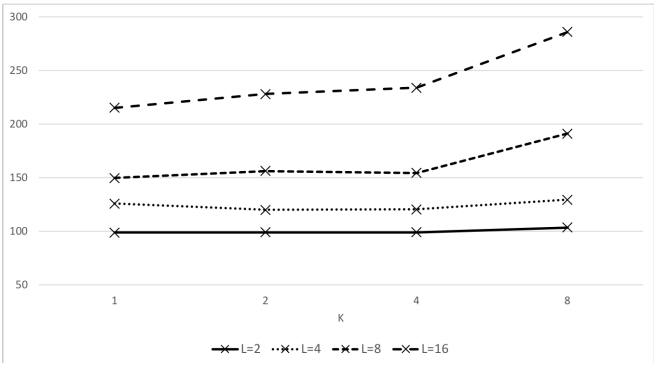

Figure 14. Indexing time.

As shown in the figure above, indexing time ranges from 98 seconds for configuration $\{L = 2, k = 1\}$ to 285 seconds for $\{L = 16, k = 8\}$. Given that the accuracy of LSH is very similar for all configurations, indexing time is a factor favored by the use of low values for L and k.

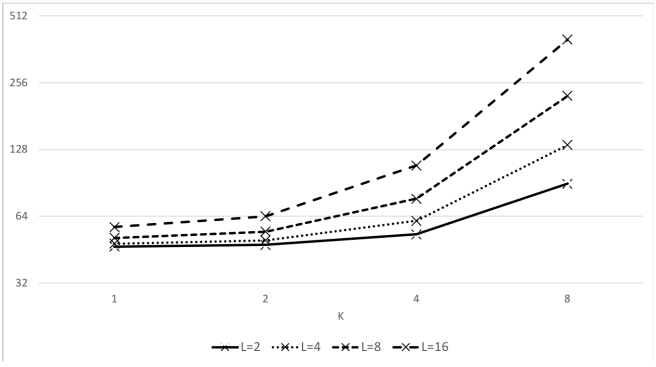

Figure 15. Database size.

In addition, the size of the database varies from 46 MB for configuration $\{L = 2, k = 1\}$ to 400 MB for $\{L = 16, k = 8\}$. However, given the use of several independent indexes, an optimal implementation is obtained because each index can be loaded on-demand as the show progresses. In the experiments, the average index size has been always below 1MB for most configurations. Only in the configurations in which $k = 8$, a notable growth in size can be seen as L grows.

The previous figures show an exponential rise for both indexing time and database size according to the increase of the values of the LSH configurations.

V. CONCLUSIONS

Throughout this work, a system with high capacity to solve the problem of synchronization between different theatre performances has been presented. The proposed system successfully synchronizes 84% of the audio fragments tested even when there are notable variations in the audio source. Likewise, acoustic fingerprints based on the SIFT transform have also demonstrated their application to the synchronization of audio sources and not only to the recognition of content, as described in the original work.

At the implementation level, a database consisting of small LSH indexes is more efficient. In preliminary experiments, the use of a single LSH index was rejected for several reasons:

- Indexing time was high.
- Low precision at searching. In this sense, given the nature of the descriptors, many of them may appear several times at different moments throughout the performance.

Since LSH indexes are one minute long and contain about 600 descriptors on average, there is no need to use high values for the number of tables and the length of the keys. In fact, and except in some specific configuration, the mean accuracy of LSH is around 76%. Although the accuracy of LSH stands at 76%, the timing accuracy of the system rises to 84% by simply assuming that, if the timing of the current fragment cannot be estimated, the elapsed playback time is the duration of the fragment itself.

There is a clear dependence between the performance of the system and the audio source used to carry out the synchronization. It is clear that performance is better when the following factors occur:

- Good acoustic conditions. The better the acoustic conditions, the better the performance. In this way, the best performance is achieved by using the head microphones (audio source 1).
- The audio used for estimations has the same acoustic characteristics as the audio used for indexing. The performance is degraded when estimation is done with audio from a different source (sources 2 or 3) than the one (source 1) used to do the indexing.
- Stage coverage is wider. If the audio capture has to be done from outside the stage, the omnidirectional microphone (audio source 3) offers better performance than the hypercardioid one (audio source 2) because its coverage is much better.

A. *Future work*

In this work, a first version of a synchronization system for theatre performances has been designed and tested. However, given the large number of configuration parameters (see Appendix), there are numerous configurations that have not been explored. In addition, there are several lines of research which can be studied:

- To explore new procedures for estimating the playback timing.
- To evaluate more extensively the performance of the system with other types of theatre plays.
- To investigate in depth the characteristics and behaviour of the SIFT descriptors. In this sense, it is possible to explore the existence of specific descriptors according to speakers or phonemes.
- To develop a filtering mechanism to determine the ability of a descriptor to be indexed. Thus, the performance of the system can be evaluated when using more specific descriptors.
- To explore the use of SIFT descriptors beyond content synchronization and evaluate their use in biometric applications such as speaker verification.

VI. ACKNOWLEDGMENTS

This work is dedicated to my wife, Beatriz, for her infinite patience, to my daughter, Candela, for not letting me write a single paragraph at one go and to my parents, Carmen and Ignacio, for their support in each one of my projects.

REFERENCES

[1] C. H. Albuquerque, E. d. Lima and V. Ferreira, "Second screen prototype for broadcasted digital TV users in ISDB-Tb standard," in *Proceedings of the 2014 IEEE 3rd Global Conference on Consumer Electronics*, Tokyo, Japan, 2014.

[2] E. D'Heer, C. Courtois and S. Paulussen, "Everyday life in (front of) the screen: The consumption of multiple screen technologies in the living room context," in *Proceedings of the 10th European Conference on Interactive Tv and Video*, Berlin, Germany, 2012.

[3] A. Papoulis, in *The Fourier integral and its applications*, New York, McGraw-Hill, 1962, pp. 244-245.

[4] H. Sakoe and S. 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.

[5] R. Macrae and S. Dixon, "Accurate real-time windowed time warping," in *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, Utrecht, Netherlands, 2010.

[6] S. Dixon and G. Widmer, "MATCH: A music alignment tool chest," in *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, London, UK, 2005.

[7] S. Dixon, "An on-line time warping algorithm for tracking musical performances," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, Edinburgh, Scotland, 2005.

[8] X. Anguera, R. Macrae and N. Oliver, "Partial sequence matching using an Unbounded Dynamic Time Warping algorithm," in *Proceedings of the 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, Dallas, USA, 2010.

[9] B. Zhu, W. Li, Z. Wang and X. Xue, "A novel audio fingerprinting method robust to time scale modification and pitch shifting," in *Proceedings of the 18th International Conference on Multimedia*, Firenze, Italy, 2010.

[10] X. Zhang, B. Zhu, L. Li, L. Li, X. W. W. Li, P. Lu and W. Zhang, "SIFT-based local spectrogram image descriptor: a novel feature for robust music identification," *EURASIP: Journal on Audio, Speech and Music Processing,* vol. 2015:6, 2015.

[11] M. Ramona and G. Peeters, "Audioprint : An efficient audio fingerprint system based on a novel cost-less synchronization scheme," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[12] N. Duong and F. Thudor, "Movie synchronization by audio landmark matching," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[13] S. Fenet, G. Richard and Y. Grenier, "A scalable audio fingerprint method with robustness to pitch-shifting," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, Miami, USA, 2011.

[14] R. Sonnleitner and G. Widmer, "Quad-based audio fingerprinting robust to time and frequency scaling," in *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx)*, Erlangen, Germany, 2014.

[15] J. Six and M. Lemman, "Panako: a scalable acoustic fingerprinting system handling time-scale and pitch modification," in *Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, Taipei, Taiwan, 2014.

[16] C. Bellettini and G. Mazzini, "A framework for robust audio fingerprinting," *Journal of Communications,* vol. 5, no. 5, pp. 409-424, 201.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[18] T. Lindeberg, "Image matching using generalized scale-space interest points," *Springer Lecture Notes in Computer Science,* vol. 7893, pp. 355-367, 2013.

[19] M. Datar, N. Immorlica, P. Indyk and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," *Proceedings of the 20th Annual Symposium on Computational Geometry,* pp. 253-262, 2004.

[20] L. Paulevé, H. Jégou and L. Amsaleg, "Locality sensitive hashing: a comparison of hash function types and querying mechanisms.," *Pattern Recognition Letters,* vol. 31, no. 11, pp. 1348-1358, 2010.

[21] *UNE 153010:2012. Subtitling for deaf and hard-of-hearing people,* 2012.

VII. APPENDIX

A. Audio characterization

- $\boldsymbol{x}$ (vector): Indexed audio stream.
- K (integer): Number of audio frames extracted for $\boldsymbol{x}$.
- $\boldsymbol{s}_k$ (vector): k-th indexed audio frame for a given $\boldsymbol{x}$.
- t_s (double): Beginning timestamp in seconds of $\boldsymbol{s}_k$.
- t_e (double): Ending timestamp in seconds of $\boldsymbol{s}_k$.
- S_k (matrix): Linear spectrogram of $\boldsymbol{s}_k$.
- $\boldsymbol{t}_k$ (vector): Timestamps values for the columns of S_k.
- $\boldsymbol{f}_k$ (vector): Frequencies values for the rows of S_k.
- S'_k (matrix): Quantized spectrogram of $\boldsymbol{s}_k$.
- $\boldsymbol{f}'_k$ (vector): Frequencies values for the rows of S'_k.
- S''_k (matrix): Logarithmic spectrogram of $\boldsymbol{s}_k$.
- I_k (matrix): Grayscale image representation of S''_k.
- m (integer): m-th frequency band.
- f_0 (double): Central frequency for the initial band.
- b_m (double): Central frequency for the m-th band.
- b_{min} (double): Lower bound for the m-th band.
- b_{max} (double): Upper bound for the m-th band.
- J (integer): Number of descriptors extracted from $\boldsymbol{s}_k$.
- F_k (set): Set of descriptors and metadata generated from $\boldsymbol{s}_k$.
- $\boldsymbol{d}_j$ (vector): j-th descriptor for $\boldsymbol{s}_k$.
- $\boldsymbol{d}'_j$ (vector): j-th normalized descriptor for $\boldsymbol{s}_k$.
- t_j (double): Absolute timestamp for $\boldsymbol{d}'_j$.
- f'_j (double): Frequency value for $\boldsymbol{d}'_j$.

B. LSH parameters

- $\boldsymbol{p}$ (vector): Normalized descriptor indexed.
- $\boldsymbol{q}$ (vector): Normalized descriptor used as query.
- r (double): Euclidean distance between $\boldsymbol{p}$ and $\boldsymbol{q}$.
- $\mathcal{G}/g$ (functions): Family of hash functions.
- $\mathcal{H}/h$ (functions): Family of projection functions.
- $\boldsymbol{a}$ (vector): Projection vector.
- b (double): Shift parameter.
- w (double): Function width.
- L (integer): Number of hash tables.
- k (integer): Number of buckets in each hash table.

C. Timing estimation parameters

- $\boldsymbol{y}$ (vector): Audio frame used to perform timing estimation.
- l (double): Length in seconds of the audio frame $\boldsymbol{y}$.
- t' (double): Timing estimation for $\boldsymbol{y}$.
- M (integer): Number of descriptors extracted from $\boldsymbol{y}$.
- Q (set): Set of descriptors and metadata generated from $\boldsymbol{y}$.
- $\boldsymbol{q}_m$ (vector): m-th normalized descriptor extracted for $\boldsymbol{y}$.
- P_m (set): Set of descriptors and metadata retrieved for a query descriptor $\boldsymbol{q}_m$.
- T (set): Set of timestamps used to perform timing estimation.
- t_m (double): Timing reference of $\boldsymbol{q}_m$ with respect to the indexed audio.
- t'_m (double): Timing value of $\boldsymbol{q}_m$ with respect to the origin of $\boldsymbol{y}$.