Pattern Matching Techniques for Replacing Missing Sections of Audio Streamed across Wireless Networks

JONATHAN DOHERTY, University of Ulster
KEVIN CURRAN, University of Ulster
PAUL McKEVITT, University of Ulster

Streaming media on the Internet can be unreliable. Services such as audio-on-demand drastically increase the loads on networks; therefore, new, robust, and highly efficient coding algorithms are necessary. One method overlooked to date, which can work alongside existing audio compression schemes, is that which takes into account the semantics and natural repetition of music. Similarity detection within polyphonic audio has presented problematic challenges within the field of music information retrieval. One approach to deal with bursty errors is to use self-similarity to replace missing segments. Many existing systems exist based on packet loss and replacement on a network level, but none attempt repairs of large dropouts of 5 seconds or more. Music exhibits standard structures that can be used as a forward error correction (FEC) mechanism. FEC is an area that addresses the issue of packet loss with the onus of repair placed as much as possible on the listener's device. We have developed a server–client-based framework (SoFI) for automatic detection and replacement of large packet losses on wireless networks when receiving time-dependent streamed audio. Whenever dropouts occur, SoFI swaps audio presented to the listener between a live stream and previous sections of the audio stored locally. Objective and subjective evaluations of SoFI where subjects were presented with other simulated approaches to audio repair together with simulations of replacements including varying lengths of time in the repair give positive results.

Categories and Subject Descriptors: E.4 [**Coding and Information Theory**]: Error Control Codes; Data Compaction and Compression

General Terms: Streaming Audio

Additional Key Words and Phrases: Streaming media, forward error correction, audio repair, data compaction and compression

ACM Reference Format:
Jonathan Doherty, Kevin Curran, and Paul Mckevitt. 2015. Pattern matching techniques for replacing missing sections of audio streamed across wireless networks. ACM Trans. Intell. Syst. Technol. 6, 2, Article 25 (March 2015), 38 pages.
DOI: http://dx.doi.org/10.1145/2663358

1. INTRODUCTION

Streaming media across networks has been a focus of much research in the area of lossy/lossless file compression and network communications. Traditional communication techniques for error mitigation perform poorly and in a bandwidth-inefficient manner in the presence of large-scale defects in a digital audio stream. There are some techniques currently implemented to overcome the problems encountered. The

Authors' address: J. Doherty, K. Curran, and P. McKevitt, School of Computing and Intelligent Systems, University of Ulster, Derry, Northern Ireland; email: {doherty-j22, kj.curran, p.mckevitt}@ulster.ac.uk.

DOI: http://dx.doi.org/10.1145/2663358

increase in bandwidth across networks should help to alleviate the congestion problem. However, the development of audio compression, including the more popular formats such as Microsoft's Windows Media Audio WMA and the MPEG group's Mp3 compression schemes, have peaked; yet, end users want higher quality through the use of lossless compression formats on more unstable network topologies. When receiving streaming media over a low-bandwidth wireless connection, users can experience not only packet loss but also extended service interruptions. These dropouts can last for as long as 15 to 20 seconds. During this time, no packets are received, and if not addressed, these dropped packets cause unacceptable interruptions in the audio stream. A long dropout of this kind may be overcome by ensuring that the buffer at the client is large enough. However, when using fixed bit-rate technologies such as Windows Media Player, a simple packet resend request is the sole method of audio stream repair implemented. A novel solution that can complement existing techniques takes account of the semantics and natural repetition of music. Through the use of self-similarity metadata, missing or damaged audio segments can be seamlessly replaced with similar undamaged segments that have already been successfully received.

We propose a technology to generate relevant self-similarity metadata for arbitrary audio material and to utilise this metadata within a wireless audio receiver to provide sophisticated and real-time correction of large-scale errors. The primary objectives are to match the current section of a song being received with previous sections while identifying incomplete sections and determining replacements based on previously received portions of the song. This technology is unique in its approach to forward error correction (FEC) technology that is used to "repair" a bursty dropout when listening to time-dependent media on a wireless network. Ultimately, using self-similarity analysis on a music file, we can "automatically" repair the dropout with a similar portion of the music already received, thereby minimising a listener's discomfort. The framework is built on pattern matching techniques that replace missing sections of audio streamed across a bursty wireless network. The system utilises an FEC approach of audio repair on the client side without the need for a resend request building on an MPEG–7 feature extraction as a method of data reduction prior to similarity analysis. A k-means clustering for similarity identification of different sections within an audio file is used to implement string matching techniques that identify similarity between large sections of the clustered audio analysis. This identifies incomplete sections and determines replacements based on previously received portions of a song based on self-similarity analysis. Such a system has mass market appeal in various sectors. These sectors include broadcast digital audio, online streaming music sites, and wireless home streaming media servers. Other immediate market areas include encompassing the similarity encoding process onto CDs for use with high-end Hi-Fi systems that operate on a wireless basis. A facility to provide a library of similarity encoded metadata is also possible based on the principle that a song needs to be analysed once and be broadcast an infinite number of times with the similarity data encoded.

Ultimately, our method for streaming music over bandwidth-constrained networks uses a technique that can work alongside existing audio compression schemes by taking into account the syntax of the music. Songs generally exhibit standard structures that can be used as an FEC mechanism. FEC is an area that addresses the issue of packet loss with the onus of repair placed as much as possible on the listener's device. This article outlines a system called *Song Form Intelligence* (SoFI) that determines packet loss and uses previously received portions of the song to predict what possible match already received exists. In turn, this is used in place of the missing packet(s) before the buffer is empty by applying and improving state-of-the-art theories and techniques in pattern matching with a syntactic, semantic, and cognitive approach.

2. MUSIC STRUCTURE

To define the structure of music is to say that music can be represented in a variety of formats depending on the needs of the user. The structure of music is a series of work that has yet to agree on a definition of the term *structure* [Wiggins 1998; Salzer 1962]. Different representations of music enable different but salient information to be displayed and stored. For example, the current structure of the well-known music score has only been used since the mid 17th century. Music notation is one of continual evolution, and the 20th century is no exception. As composers have found new means of expression, they have developed new means of writing them down. For example, methods of indicating microtones were found in early 20th century work, and symbols borrowed from mathematics have been used to denote complex rhythmic relationships. There have even been attempts, particularly in the first half of the 20th century, to invent completely new systems such as Klavarscribo [Walker 1997], but these have not been adopted as a notational standard by many. Syntax can be defined as a set of principles governing the combination of discrete structural elements into sequences [Jackendoff 2002]. Ockelford [1991] investigates finding a common ground within the varying musical structure representations based on repetition, where the conclusion leads to creating yet another structure model with music being represented by a system of variables defined as perspective. This concludes with the common ground assumption that one perspective is deemed to exist in imitation of another by determining a zygonic theory of music-structural cognition. Since around the beginning of the 17th century, music notation has been defined in a standardised format in the style of music typography called *plate engraving*. This style of representation is ideal for musicians and composers, but when using a computer to read/interpret, the notation difficulties arise. This is compounded by the myriad computer programs for music notation, making sharing music between them difficult. Different programs use different representation styles: graphical, symbolic, numerical, and so forth. The reason for this is based on the particular task that the software has to perform, but because of this, no one program can do everything equally well.

Music information retrieval (MIR) is complex: the queries are often “fuzzy” (query by humming or singing), and the data relationships are complicated. Queries that work well with one format are unsuitable for another. Until recently, the only music interchange format commonly supported was MIDI. MIDI is an ideal format for performance applications like sequencers, but it is not as suitable for other applications, such as music notation. MIDI does not know the difference between an F-sharp and a G-flat, or many other aspects of music notation. Notation Interchange File Format (NIFF) and Standard Music Description Language (SMDL) have attempted to solve the interchange problem, but they still have their limitations depending on the specific needs of users. NIFF is used to interchange music between scanning and notation applications, whereas SMDL was an attempt to create a formal specification for music, but as yet it has limited implementation mainly due to its complexity [Good 2001].

Recordare is a manufacturer and retailer of digital sheet music software that has developed the MusicXML format to create an Internet-based method of sharing musical scores, with the aim to provide the same role for interactive sheet music that Mp3 files serve for recorded music. By using an XML-style layout, Recordare has developed a standardised notation for the representation of music in a format that can be used by almost any application. Scanning and reading applications can import the content and present it in a graphical format with precise representation, and other systems that previously relied on MIDI files can now import a more accurate conversion of the music. It should be pointed out that MusicXML has been developed with the need for a standardised representation; due to this, it is a verbose representation where size is of less importance in relation to being application independent.

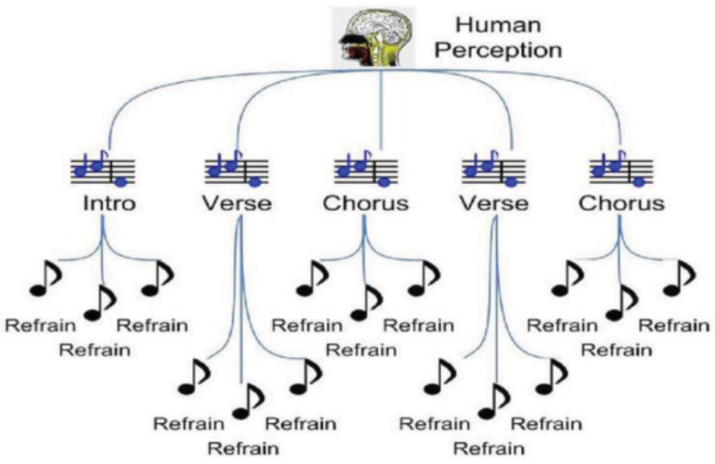


Fig. 1. Hierarchical breakdown of a song in western tonal format.

Extraction and classification of audio using semantics is a popular approach to music representation/interpretation [Herrera et al. 2004; Slaney et al. 2002]. Semantic-audio Retrieval (SAR) creates a connection between semantic space and acoustic space using cluster abstraction for higher-level representation similar to the MPEG–7 format, which uses hierarchical semantics for its tagging elements of data with descriptors. Semantic Interaction with MusicAudio Contents (SIMAC) [Semantic 2008] is a research group dedicated to providing meaningful descriptors of musical content to aid in describing music collections that are close to emulating the mind's way of organising and understanding its contents. Often we can describe songs in terms such as having a *VCVC* form, or a *VCBVC* form. However, not all songs follow a verse, chorus, and bridge pattern. A large percentage of songs these days follow a type of song form that includes a chorus, so a verse, chorus, verse, chorus type of song is *VCVC*. There is also a common song form, which includes a bridge, so a typical form with a bridge might be *VVCVCBC*. Other less common forms may include a prechorus that is a lead-up to the chorus (labelled *L*); intros (labelled *I*) at the very beginning of a song and extras (labelled *E*) are the lead-outs or endings to a song [Jackson 2008]. There are many variations of these forms, with some songs starting with the chorus and others having more than one bridge. Any individual artist may have all kinds of variations. Most songwriters do not start writing by coming up with a song form first. It usually reveals itself as the song is being written. It is, however, a quick and easy language to use when discussing the process with other writers.

The method of applying the hierarchical approach to audio in the context of its structure to describe the different portions of the audio is shown in Figure 1, where an audio file has a root node of "song," which is then broken down into sections where the introduction, verse, and chorus are identified. Each section can then be broken into smaller sections again using the natural refrain of songs in western tonal format. To provide a cognitive representation, a piece of music can be defined using a form of semantic representation. Experience is not only related to the richness of perception; it also has a role in the construction of knowledge with meaning being characterised in terms of the experience: of the person becoming conscious of the music. As such, it is a basic claim of cognitive semantics [Jackendoff 1987; Lakoff 1988] that meaning is an account of reality by people. The perception of music has been a popular area of cognitive research, with the human mind's methods for interpretation of musical structure and patterns being a key area. Lerdahl and Jackendoff [1983] conducted some of their earlier work to provide investigations into the mind's cognitive approach to grouping and reduction within music. They showed that complex musical structures were reduced to more abstract representations in the human mind: "Reduction Hypothesis: The listener attempts to organise all the pitch-events of a piece into a single coherent structure, such that they are heard in a hierarchy of relative importance" [Lerdahl and Jackendoff 1983].

One of the more challenging areas of automatic transcription of polyphonic music is the estimation of a song's fundamental frequency. Primarily this is due to the nature of music with a mixture of various musical instruments that have diverse spectral characteristics. When an object is forced into resonance vibrations at one of its natural frequencies, it vibrates in a manner such that a standing wave pattern is formed within the object. Each natural frequency an instrument produces has its own standing wave pattern. These patterns are only created within the instrument at specific frequencies of vibration known as harmonic frequencies. At any frequency other than a harmonic frequency, the resulting disturbance of the medium is irregular and non-repeating. For musical instruments that vibrate in a regular and periodic fashion, the harmonic frequencies are related to each other by simple whole number ratios. It is at these frequencies that instruments sound pleasant. The lowest frequency produced by any particular instrument is known as its fundamental frequency (F0). Performing autocorrelation yields information about repeating events, such as identifying the fundamental frequency of a signal, which does not actually contain that frequency component but implies it with many harmonic frequencies. This is particularly common when multiple instruments are combined. Multiplication between the signal and a shifted version of itself results in a graph illustrating peaking patterns [Wallach 2004].

2.1. Feature Extraction and Audio

The choice of features that can be extracted from audio depend greatly on the criteria of the analysis performed. These features can vary from pitch estimation to fingerprinting depending on the nature of the queries involved. MPEG–7 can be used as a feature extraction for a variety of extraction needs.

MPEG–7 is an international standardised description of various types of multimedia information [Martinez et al. 2002]. Whereas MPEG4 defines the layout and structure of a file and codecs, MPEG–7 is a more abstract model that incorporates a markup language to define description schemes and descriptors—the Description Definition Language (DDL). Using a hierarchy of classification allows different granularity in the descriptions. MPEG–7 as a descriptive tool can greatly enhance this area by adding additional metadata based on the content of the audio as well as standard keyword descriptors. It should be noted that the MPEG–7 standard only specifies the format for descriptions of content, not the algorithms to utilise these descriptions. Developers have begun implementation of MPEG–7 only recently. The MPEG–7 Library is a comprehensive list of more than 800 description schemes and descriptors using classes in C++, which enables developers to use the functionality of MPEG–7 in their own applications. The Java MPEG–7 Audio Encoder is a complete software package for MPEG–7 analysis, with the audio analysis results stored in XML format. The Java MPEG–7 Audio Encoder can be launched from the Web using a Java virtual machine (JVM), or on a local machine as a command line application. It has been widely used as an analysis tool for similarity analysis and pattern recognition [Cho and Choi 2005]. Through the combination of descriptors, description schemes, and a DDL, MPEG–7 can facilitate efficient searching and filtering of files. The two basic descriptors are the AudioWaveform (AW) and AudioPower (AP) presented as temporarily sampled scalar values. The AW descriptor gives a minimum/maximum value of the signal range within

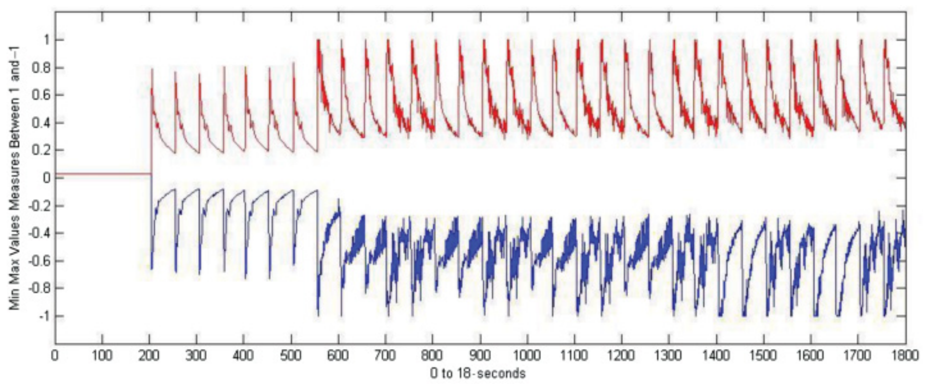


Fig. 2. Example MPEG–7 AW representation.

a specified temporal resolution. The AP descriptor provides a measure of the square of the waveform values and thereby provides a simplified representation of the signal showing peaks where the signal has a higher amplitude. A comparative study of some of the LLDs by Lukasiak et al. [2003] shows that when facilitating comparison between audio segments, the AW performs better based on the principle that it has two measurements for each frame as opposed to a single value from the AP analysis. An example giving the first 18 seconds of a guitar playing a rendition of the "12 Bar Blues" is shown in Figure 2. When compared, the level of detail in the AW representation is higher with regard to the content of the audio file.

The audio spectrum envelope (ASE) is a log-frequency power spectrum that can facilitate generation of a reduced spectrum of the original audio. This is performed by summing the energy of the power spectrum within a series of frequency bands. Bands are equally distributed between two frequency edges: loEdge and hiEdge. Default values of 62.5Hz and 16KHz correspond to the lower/upper limit of hearing. The spectral resolution *r* of the frequency bands within these limits can be specified based on eight possible values, ranging from 1/16 of an octave to 8 octaves. Each ASE vector is extracted every 10ms from a 30ms frame (window) that gives a compact representation of the spectrogram of the audio. Audio spectrum flatness (ASF) describes the flatness properties of the spectrum of an audio signal within a given number of frequency bands. The flatness of a band is defined as the ratio of the geometric mean (i.e., the central tendency of a vector) and the arithmetic mean of the spectral power coefficients within the band. The AudioSpectrumBasis descriptor is a container for basis functions for projecting a spectrum onto a lower-dimensional subspace suitable for probability model classifiers (e.g., neural networks and hidden Markov models.) The reduced basis consists of decorrelated features of the spectrum with salient information described more efficiently than with the direct spectrum representation. The audio spectrum centroid (ASC) is the centre of gravity of a log-frequency power spectrum. Unlike the previous MPEG–7 low-level descriptors, the ASC is of a scalar type and provides a high level of dimensional reduction at the cost of high information loss. The ASC provides information on the shape of the power spectrum and indicates whether a power spectrum is dominated by low or high frequencies. The ASC can be regarded as an approximation of the perceptual sharpness of the signal by indicating the location of the centre of mass of the spectrum. Perceptually, it has a robust connection with the impression of brightness[1] of a sound [Schubert et al. 2004]. Seo et al. [2005] applied

[1]The brightness of a sound is indicated by the amount of high-frequency content.

the ASC to audio fingerprinting. An audio fingerprint is applied to recognise audio in the same way a human fingerprint is applied to identify an individual. Fingerprints are perceptual features (short summaries) of a multimedia object and can be useful in applying search/retrieval queries and copyright detection. By converting the audio signal to mono, down-sampling it, and then transforming it to the frequency domain with FFT, Seo et al. [2005] were able to create a reliable fingerprint matching system. The audio spectrum obtained was divided into 16 critical bands, and the normalised frequency centroid for each band was calculated. These centroids acted as fingerprints of the audio frame. Seo et al. [2005] showed this approach to be robust through "quality-preserving" signal processing steps, and it outperformed other commonly used features, such as tonality and MFCC, in the context of audio fingerprinting.

3. MUSIC INFORMATION RETRIEVAL

Music is a combination of pitch, tempo, timbre, and rhythm, making analysis of music more difficult than text. Structuring a query for music is made difficult owing to the varying representations and interpretations including natural transitions in music. Adding to the complexity of music structure and query structure is the method of audio analysis. The format of an audio file limits its type of use, as different file formats exist to allow for better reproduction, compression, and analysis. Hence, it is also true that different digital audio formats lead to different methods of analysis. Musical Instrument Digital Interface (MIDI) files were created to distribute music playable on synthesisers of both the hardware and software variety among artists and equipment, and because of its notational style, the MIDI format allows analysis of pitch, duration, and intensity [Doraisamy and Ruger 2004]. An excellent tool for analysis of MIDI files is the MIDI Toolbox [Eerola and Toiviainen 2004], which is based on symbolic musical data, but signal processing methods are applied to cover such aspects of musical behaviour as geometric representations and short-term memory. Recent work within polyphonic music has shown that similarity within different sections of a piece of music can aid in both pattern matching for searching large datasets and pattern matching within a single audio file [Foote and Cooper 2003; Meredith et al. 2001; Dannenberg and Hu 2003]. Results from analysis of an audio stream are stored in a similarity matrix created by Foote and Cooper [2003]. The similarity matrix is generated by measuring the difference between the row and column for the same data. Data along the i^0, j^0 to i^1, j^1 diagonal will have an exact similarity, but any comparisons "off" the diagonal give a measure of the similarity of the two values. Analysis is performed using short-time Fourier transform (STFT) to determine the spectral properties of the segmented audio; this is a variation of the discrete Fourier transform (DFT), which allows for the influence of time as a factor. Bartsch and Wakefield [2001] used the chroma-based spectrum analysis technique to identify the chorus or refrain of a song by identifying repeated sections of the audio waveform, with the results also being stored in a similarity matrix. The following range of applications vary from database search/retrieval applications and indexing systems that allow quicker user browsing to automatic music replication systems based on a specific composer's style. It should be noted that music recognition is only in its infancy and has limited accurate results. The recognition of scanned text can have an accuracy of up to 95% and programs using speech recognition have 70% to 80% accuracy, whereas systems for music recognition only claim a 60% to 70% accuracy rating depending on the audio format, although new research is producing results of up to 90% when using the MIDI format [Doraisamy and Ruger 2004].

MELody inDEX (MELDEX) [McNab et al. 1997] is a query by humming application similar to systems developed by Ghias et al. [1995] and Cater and O'Kennedy [2000]. MELDEX allows a user to use a microphone to enter notes by humming a tune and then

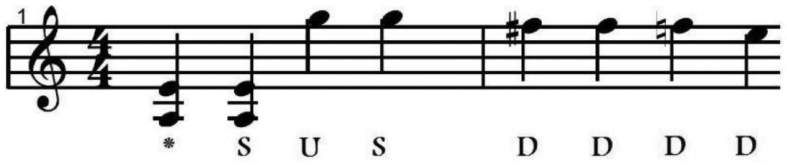


Fig. 3. Parsons code representation of a music score.

searches in a database for a similar match. To match user input with content held in the database, MELDEX primarily uses pitch and the fundamental frequency to process the signal for similarity matching. MELDEX filters the input to remove as many harmonics as possible while preserving the fundamental frequency. The beginnings and ends of notes are defined using a technique primarily found in voice recognition. Notes depend on the user using "ta" or "da" to hum the input, which causes a drop in amplitude of the waveform of 60ms at each utterance, allowing each note to be more easily identified. MELDEX then uses "string matching" to identify the input from the user with audio held in a database using approximation to score the results, which are returned in order of accuracy. Similarly, the MelodyHound melody recognition system [Prechelt and Typke 2001] was developed by Rainer Typke in 1997. It was originally known as Tuneserver and hosted by the University of Karlsruhe. It was designed initially for query by whistling—that is, it will return the song in the database that most closely matches a whistled query. A more unusual method of input is where the user can enter information about a song in the form of Parsons code. Each pair of consecutive notes is coded as "U" ("up") if the second note is higher than the first note, "R" ("repeat") if the pitches are equal, and "D" ("down") otherwise. The first note of any tune is the reference point, so it does not show up explicitly in the Parsons code and is entered as an asterisk (*). Parsons [1975] showed that this simple encoding of tunes, which ignores most of the information in the musical signal, can still provide enough information for distinguishing between a large number of tunes. Figure 3 shows the first line of a "12 Bar Blues" music score with the resultant Parsons code following.

Humdrum is a general-purpose software system intended to assist music researchers and consists of two distinct components: the Humdrum Syntax and the Humdrum Toolkit [Humdrum 2008]. The syntax provides a common framework for representing information in ASCII format. Within the syntax, an endless number of representation schemes can be "user defined." The Humdrum Toolkit provides a set of more than 70 interrelated software tools.

Themefinder [Sapp and Aarden 2008] identifies common themes in western classical music, folksongs, and Latin motifs of the 16th century. Themefinder provides a Web-based interface of the Humdrum Toolkit, which indexes musical data for searching [Humdrum 2008]. This in turn allows searching of databases containing musical themes. Currently, the most commonly used Humdrum representation is **kern. Kern is a core pitch/duration representation.

C-BRAHMS (Content-Based Retrieval and Analysis of Harmony and other Music Structures) [Lemstrom et al. 2003] concentrates on retrieving polyphonic music from large-scale music databases containing symbolically encoded music. C-Brahms uses a number of different algorithms that allow music to be in various formats, including MIDI, monophonic, and polyphonic. It also allows partial and exact matching approaches. C-Brahms uses a geometric representation of both the query pattern and the source pattern, allowing a Euclidean measurement of difference.

CubyHum is a query by humming application [Pauws 2002] that attempts to detect pitches in a sung melody and compares these with symbolic representations of known melodies within a stored database. CubyHum estimates the pitch from the query by a technique called *subharmonic summation* (SHS), which initially was proposed by Hermes [1988]. In short time frames, SHS computes the sum of harmonically compressed spectra and selects the maximum sum result as the pitch estimate in that time frame. Standard signal processing techniques using short-time energy, pitch-level shifts, and amplitude envelopes are then used for finding note onsets. The resultant data is then combined to describe the pitch and duration of the query, allowing normal transcription to the MIDI format for comparison with songs stored in the database.

notify!Whistle is a query by whistling/humming system for melody retrieval similar to CubyHum, along with a similar conversion of user queries to a MIDI format. By using a piano roll representation of the query, the user is allowed to change the original input to account for errors [Kurth et al. 2002]. However, unlike CubyHum, which uses a string-based approach for comparisons, Kurth et al. [2002] use an index-based approach for pattern matching. By describing songs as notes within documents represented by the form $Di\ N$ and queries as $Q\ N$ allows queries to be performed using set theory and is an alternative approach to the problem of incorrect notes and mismatches that are common with user-generated input queries.

Muscle Fish content-based retrieval CBR technology searches for audio files on the basis of how they sound [Wold et al. 1996]. It can also be used to classify sound files or live sound inputs. An additional feature of Muscle Fish is its ability to cluster sound files according to category and search for sounds that are similar in their features. Muscle Fish's approach is to analyze sound files for a specific set of psychoacoustic features. This results in a query vector of attributes that include loudness, pitch, bandwidth, and harmonicity. A covariance-weighted Euclidean (Mahalonobis) distance is then used as a measure of similarity between a given sound example and all other sound examples. Sounds are then ranked by distance, with the closer ones being more similar.

Marsyas (MusicAl Research SYstem for Analysis and Synthesis) is a collection of tools aimed at audio analysis. The principle behind Marsyas is to allow researchers to utilise a standardised set of analysis tools, allowing them to collaborate and compare results using a level platform. Marsyas uses a semiautomatic approach that combines both manual and fully automatic annotation, giving a necessary degree of flexibility that depends on the research approach. Marsyas combines spectral analysis with pitch and harmonicity, as well as techniques using singular value decomposition (SVD), principal component analysis (PCA), and multidimensional datasets [Jolliffe 1986].

SAM is a set of programs designed to read/record digital audio, extract a pitch contour, compute a similarity matrix, find clusters of similar sequences, and build an explanation of the music in terms of structural relationships [Tanguiane 1993]. Using pitch extraction, SAM identifies potential areas where the signal amplitude is low (i.e., signal noise) and areas where there are clear peaks (notes). Similar groups of notes are then identified, and a similarity matrix is built. This matrix is then used to identify similar groups of notes within the audio file. One area identified by SAM is that the similarity between groups of notes is not transitive, in that if group A was found to be similar to group B, and group B was found to be similar to group C, it did not mean that group A was similar to group C. This was because exact pattern matching was not used, and limits were set as to how exact the match had to be.

Within the area of MIR, cellular automata, genetic algorithms, and neural networks are primarily used as machine learning and composition tools. They have been used for the analysis of a particular composer's style and then to create/simulate a similar piece of music based on this analysis [Pearce and Wiggins 2002]. Initial results have shown that some (experienced) musicians who are not familiar with a particular composer's work find it difficult to tell the difference between the original and synthesized music. For self-similarity and pattern matching, DSP techniques combined with pattern matching using scalars, vectors, and matrices are more common.

4. NETWORKED AUDIO

Packet delay from network congestion has been partially alleviated using routing protocols and application protocols such as Real-Time Transport Protocol (RTP). These have been developed to assign a higher priority to time-dependent data. However, it is also the case that some servers automatically dump packets that are time sensitive, so streaming applications have had to resort to "masking" the packets by using "HTTP port 80" so that packets appear as normal Web traffic. The latest addition to network protocols specifically addressing "real-time" communication include Voice over Internet Protocol (VoIP), a technology that allows telephone calls using a broadband Internet connection across a packet-switched network instead of a regular (or analog) phone line.

It is necessary for standards to be defined for communication across networks. Computers need a specific set of rules and guidelines to communicate in the same way humans need language to be able to communicate with each other. Without a predefined set of guidelines, one computer would not understand what the other was saying, just as a French person (who does not speak Chinese) would not understand a Chinese person when they both talk in their native languages. TCP/IP was the first recognized standard for communication between computers across a network. Protocols are an open set of rules of behaviour that are independent of an operating system and architectural differences. They are available to everyone to allow for development and are changed on consensus. These protocols are published as Requests for Comments [Socolofsky and Kale 2008] and contain the latest versions of the specification of all standard TCP/IP protocols. Each layer can contain any given number of protocols that perform specific functions relating to that layer. It should be noted that each layer does not know or care how layers above and below work, simply that data is passed between them. The main benefit of TCP/IP is that it provides interoperable communications between all types of hardware and operating systems [Stevens 1993]. Each layer does not define a single protocol but represents a communication function that can be performed by any amount of different protocols.

One of the most recent additions to network communication is the inclusion of VoIP. This new technology allows users to make telephone calls using a computer to either another computer with an Internet connection or a telephone for the cost of a local call. VoIP converts the voice signal from a telephone/microphone into a digital signal that travels over the Internet and is converted back at the receiving computer. One of the driving forces behind VoIP is its cost; the Internet does not recognize state and country borders. With VoIP technology, the distinction between local, long-distance, and international calling largely disappears, so callers can save on long-distance and international charges. VoIP is not without its problems, which are associated with real-time traffic across networks: packet delays/losses [Jiang and Schulzrinne 2002]. One of the main issues faced by Internet telephone applications such as Skype and Vonage is the quality and reliability of communication via the Internet. The traditional public switched telephone network sets a high standard for IP telephony to match before mainstream acceptance.

To make audio files more manageable, it is necessary to reduce their size; there are several ways in which this can be done. One method is to reduce the sampling frequency [Menin 2002] of the recording system. However, this has some serious side effects as far as sound quality is concerned, as high-frequency content of the sound is lost, leading to recordings lacking in brightness and clarity. Mp3 compression uses a number of perceptual coding techniques to reduce file size and yet maintain quality audio. Through the use of lossy compression, the sample rate of Mp3 files determines the level of quality. The compact disc (CD) audio format uses a 16-bit sample rate (samples measured every 44.1kHz or 44,100 slices every second), which equates to

5.2Mb per minute of recording. The result in real terms is that Mp3 coding shrinks the original audio signal from a CD (PCM format) by a factor of 12 without sacrificing sound quality—that is, from a bit rate of 1411.2Kbps of stereo music to 112 to 128 Kbps.

4.1. Jitter Control

Streaming audio over a network has one serious problem associated with it: jitter. Jitter is when media being played back starts and stops as the packets of the stream are sent inconsistently. Because of the nature of networks, it is possible for packets sent to arrive in a different order from which they were originally sent. The receiving application then has to restructure these into their correct order. In the context of streaming, this can be problematic as portions of audio may arrive too late to be played, leading to sections of the audio being dropped altogether and making the audio sound jittery. This effect is compounded by the quality of the transmission, and high-quality audio signals require a large number of packets that in turn require a larger bandwidth [Bush 2000].

Jitter control can be managed at hops across the network. At each hop, a packet is examined to determine its position relative to the rest of the stream. If packets are found to be "lagging behind," they can be forwarded with priority over other packets in the same stream. Likewise, packets that have managed to jump the queue are "slowed down" to allow others to catch up. Jitter occurs more frequently when streaming audio across wireless networks. The nature of wireless communication and its inconsistencies amplify the effects of packet loss when bursty packet losses occur. Streaming media players are almost indifferent to the format of an audio file before streaming, but results from analysis vary greatly depending on the format used. The quality of the audio signal received very much depends on both jitter and file formats.

4.2. Streaming Media

When surfing the Web, it is common to find embedded audio and video that need additional applications to handle the content. Streaming media is the action of sending encoded (digitized) audio and/or video data out across the Internet as a series of small data packets that may be viewed by the end user in real time. The data stream is accessed via a media player (Windows Media Player, iTunes, or QuickTime). Essentially, the media player captures the data packets (audio stream) and places them in their respective order for real-time viewing. The audio and/or video input source or file is streamed via either a hardware or software encoder. A hardware encoder takes input from an external audio/video input source, encodes it, and then streams it directly. However, a software encoder can encode files being played by an internal (software) player as well. In most cases, the nature of the broadcast determines whether a hardware or software encoder is used. For example, a "live" broadcast across the Internet of a concert will utilise hardware encoders directly encoding the input before broadcast, and a radio station using the Internet as a broadcast medium will use software encoders to convert the "prerecorded" digital media into audio streams for broadcast.

There are two major parts to most streaming media servers: (1) the component providing the content (i.e., source clients) and (2) the component responsible for serving that content to listeners. Icecast [Icecast 2008] is a streaming media server that currently supports Ogg Vorbis and Mp3 audio streams. It can be used to create an Internet radio station or a privately running jukebox, and many things in between. It is versatile in that new formats can be added relatively easily, and it supports open standards for communication and interaction. Through a Web-based interface, the user can manipulate many server features. Icecast allows the administrator to move listeners from one source stream to another (mountpoints), disconnect connected sources, disconnect connected listeners, gather statistics, and many other activities. Ices is a source client

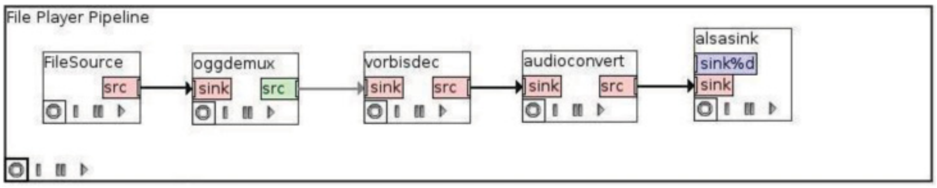


Fig. 4. A basic GStreamer media player.

for a streaming server (e.g., Icecast). The purpose of Ices is to provide an audio stream to a streaming server without any regard for the number of listeners connected or the limitations in place, such as the amount of bandwidth or the number of ports available. It is not necessary for Ices to be on the same physical machine as the streaming server (Icecast) because using separate machines helps to alleviate the processing needs. However, it is easier to manage both the server and source client when they are located on the same machine. The Ices configuration file is also an XML file.

GStreamer is a development framework for creating streaming media applications. GStreamer's development framework makes it possible to write almost any type of streaming multimedia application. The GStreamer framework is designed to make it easy to write applications that handle audio, video, or both. One of the most obvious uses of GStreamer is using it to build a media player. GStreamer includes components for building a media player that can support a wide variety of formats, including Mp3, Ogg Vorbis, MPEG–1 and 2, AVI, QuickTime, MOD, and more. Its main advantages are that the pluggable components can be mixed and matched into arbitrary pipelines so that it is possible to write a full-fledged video or audio editing application. GStreamer's core function is to provide a framework for plugins, dataflow, and media type handling/negotiation. An element is the most important class of object in GStreamer. An element has one specific function, which can be the reading of data from a file, decoding of this data, or outputting this data to a sound card or any other form of output. By chaining together several such elements, a pipeline is created that can perform a specific task, such as media playback or capture. GStreamer ships with a large collection of elements by default, making the development of a large variety of media applications possible simply by chaining different elements depending on the needs of the developer. A basic media player is shown in Figure 4, which presents a pipeline containing elements and their source pads required for basic playback of an audio file encoded in the Ogg format. It should be noted that the output element "alsasink" is required on the Unix/Linux operating system for soundcard output.

4.3. Streaming Audio Approaches to Packet Loss

Solutions to packet loss, jitter, and associated problems within streaming audio have included research in several varying techniques. The probability of packet loss across bursty networks has been modelled where time delay is used to control the flow of packets and measure the difference between the current time and the time the packet arrives [Lee and Chanson 2004]. This technique can be used to predict network behaviour and adjust audio compression based on current network behaviour. Higher compression results in poorer-quality audio but reduces network congestion through smaller packets. A variation of this theme has been used to create new protocols that allow scalable media streaming [Mahanti et al. 2003]. Randomising packet order to alleviate the large gaps associated with bursty losses has been implemented, where the problem was reduced by reordering the packets before they are sent and reassembled into the correct order at the receiver [Varadarajan et al. 2002]. This reduced the bursty loss effect since packets lost were from different time segments. Although nothing is

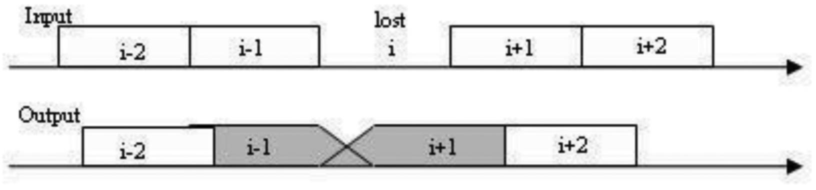


Fig. 5. WSOLA loss concealment [Liang et al. 2003].

done to replace the missing packets, overall audio quality improved through smaller gaps in the audio—albeit more frequent.

Several techniques have been developed that use some form of redundancy where repetition replaces lost audio segments. Sending packets containing the same audio segments (but with a lower bit rate) alongside the high bit-rate encoding increases the likelihood of packet arrival, but at the loss of audio quality, and increases the overall network bandwidth usage [Perkins et al. 1998]. Another approach to using redundancy in the form of unequal error protection (UEP) has been developed, where improvement is achieved with an acceptable amount of redundancy using advanced audio encoding (AAC) [Wang et al. 2003]. Segmentation of the audio into different classes, such as drumbeats and onset segments, allows priority to be applied to more important audio segments with an Automatic Repeat-reQuest (ARQ) applied to high-priority segments and a reconstruction technique for the replacement of low-priority segments based on the AAC received in previous segments. One of the more recent methods of interpolation of low bit rate coded voice, where observation of high correlation of linear predictors within adjacent frames allows descriptions to be inserted using linear spectral pairs (LSP) and then reconstruct lost packets using linear interpolation has been used by Wah and Lin [2005]. This allowed packet-loss replacement without increasing the transmission bandwidth. Wah and Lin [2005] do point out that this approach is a trade-off between the quality of the received packets and the ability to reconstruct lost packets.

4.4. Voice Communication

Traditional methods for interpolation between lost packets are still popular with Internet telephone applications where timing is critical and limited signal degradation is acceptable. Forward Error Checking (FEC) approaches include Waveform Similarity Overlap Add (WSOLA) [Liang et al. 2003] where lost packets of a section of voice are merged based on pitch similarity (Figure 5) rather than straightforward interpolation. WSOLA decomposes the input into overlapping segments of equal length, which are then realigned and superimposed to form an audio output of equal and fixed length. Using a windowing technique minimises the changes in signal strength of the two segments. This leads to increased processing overhead, but concealment is possible if packet loss is limited to one or two sections.

FS-CELP [Wah and Lin 2005] is an implementation of the Federal Standard 1016 Code Excited Linear Prediction based on the principle of linear predictive coding (LPC). LPC is a powerful speech analysis technique, and one of the most common methods for encoding good quality speech at a low bit rate. LPC provides relatively accurate estimates of speech parameters. Using multidescription coding, finite state code excited linear pairs (FS-CELP) allows multiple descriptions of the signal to be encoded into two streams: odd and even samples. Reconstruction of the original signal is possible only if one of the two streams is lost. Only when there is loss of both odd and even streams is error correction not possible. Building redundancy into packets is a popular method for FEC. FreePhone [Bolot et al. 1999] uses an adaptive approach where redundancy is coded depending on the loss characteristics of the network at that time using the real-time control protocol (RTCP) feedback. Bolot et al. [1999] justify this by noting that there is little point in using high levels of redundant information encoded into the packets if there is little chance of it being used. The actual method for FEC used is a simple “next packet scenario” where packet n is encoded with not only its own data but also with a redundant version of packet n-1. In the event that packet loss occurs for packet n-1, the packet can be reconstructed using the information encoded into packet n. Using an adaptive approach minimises the extra network bandwidth required to carry the redundant data, thereby reducing the overhead required in using FEC. Traditional ARQ [Lin et al. 1984] methods have been improved with the use of gap detection in packets where a large number of packets in sequence are lost. Detection of large gaps allows for a retransmission request before buffer levels run low, thereby allowing sufficient time for the missing segments to be resent. Another approach is to use timeout detection, where packet loss is detected by estimating the arrival time of packets. If a packet has not arrived by a certain deadline, it is assumed to be lost and an ARQ is sent. Both techniques have their merits under certain conditions, but a combination of these has been used to improve overall performance [Sze et al. 2001]. The main drawback of this approach is that there is no reduction in the buffer size.

4.5. Audio Repair

Most forms of audio analysis using computers to identify the characteristics of a sound (e.g., amplitude, velocity, wavelength, and frequency) involve digital signal processing (DSP). A varied number of different techniques have been used to analyse audio in respect to different qualities. The results required determine the type of analysis used. However, almost all forms of DSP are based on one core principle: Fourier analysis (also known as spectral analysis, frequency analysis, or harmonic analysis). Fourier analysis is a mathematical technique for describing a series of waves in terms of repeated cycles of components. One of the core principals of Fourier analysis is that it is based on an infinitely repeating signal. DFT transforms a series of discrete observations measured over a finite range of time into a discrete frequency-domain spectrum [Williams 1997]. The resultant output of DFT analysis is a continuous frequency spectrum that includes all frequencies. It should be noted that results from Fourier analysis depend on the sampling interval used, and a large sample interval can lead to information being missed. The fast Fourier transform (FFT) is a DFT algorithm that reduces the number of computations needed from $O(N^2)$ to $O(N \log N)$ operations.

One of the shortcomings of the Fourier transform is that it does not give any information on the time at which a frequency component occurs. STFT is one approach that gives information on the time resolution of the spectrum. STFT uses a moving window over the signal, and the Fourier transform is applied to the signal within the window as the window is moved. One of the main problems associated with DFT analysis is leakage. A DFT is calculated over a finite sample using a rectangular window, where abrupt changes at the beginning and end of the window can cause leakages (nonzero values). Other windows that reduce leakage more than using a rectangle window include the Hann window and the Hamming window, which is similar to the Hann window except it is raised on a pedestal [Lyons 2004]. A window is applied to both the beginning and the end of the sample interval to smooth out to a single common amplitude value. Figure 6 shows a waveform truncated with a rectangular window. By applying a Hann/Hamming window, this truncated signal can be smoothed at both ends, bringing the waveform to zero. This is achieved by multiplying the signal samples by the Hamming function; the samples at the centre are “windowed” by the largest factor, and this decreases in a smooth sinusoidal fashion as it moves away from the centre with the samples at each end being multiplied by zero. The choice of “window” is signal dependent and varies

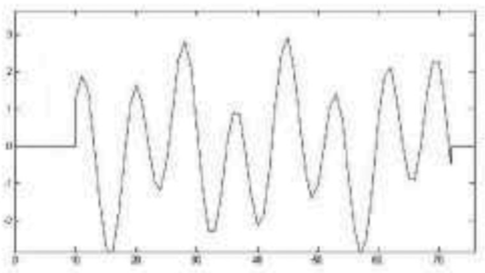


Fig. 6. A waveform with rectangle windowing applied.

depending on analysis requirements. The Hann window results in less leakage than the Hamming window, but with a trade-off of more signal-to-noise ratio.

5. PATTERN CLASSIFICATION AND MATCHING

There are distinct differences between the definitions of pattern classification and pattern matching. Pattern classification aims to classify data based either on a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations defining points in an appropriate multidimensional space. This is in contrast to pattern matching, where the pattern is rigidly specified. Pattern recognition is more complex when templates generate variants. For example, in English, sentences often follow the noun–verb phrase pattern, but some knowledge of the English language is required to detect the pattern [Jurafsky and Martin 2000]. Regardless of whether pattern classification or matching is used, there are three elementary steps to be performed prior to matching/classification: sensing, segmentation/grouping, and feature extraction [Duda et al. 2000].

Schedl et al. [2011] address the problem of similarity measurement between music artists via text-based features extracted from Web pages through evaluation of different term-weighting strategies, normalization methods, aggregation functions, and similarity measurement techniques. Zhang et al. [2009] use an incremental locality sensitive hashing algorithm to support efficient retrieval processes with different kinds of queries for music similarity measures. Turnbull et al. [2008] developed a computer audition system that can both annotate novel audio tracks with semantically meaningful words and retrieve relevant tracks from a database of unlabelled audio content given a text-based query. In addition, Turnbull et al. [2007] use the weighted mixture hierarchies expectation-maximization algorithm, which has been specifically designed to handle real-valued semantic association between words and songs rather than binary class labels.

5.1. Sensing

The input to a pattern recognition application can be in many forms, such as images, speech/sound, or text. The difficulty lies in the limitations of the input image resolution, the volume of data in a sound file, or signal distortion.

5.2. Segmentation and Grouping

Segmentation is one of the most difficult problems in pattern recognition. For example, in speech recognition, an application may need to recognize individual phonemes and combine these to form the word. Consider the words *sheep* and *shop*; the speaker will use different positions of his or her lips to pronounce the "sh" of both words. When saying the word *sheep*, the speaker will have the lips tight to the face, but when saying the word *shop*, the lips will be in a rounder form in preparation for the “op” portion. This is commonly referred to as anticipatory coarticulation, commonly known as rounding [Bilmes and Bartels 2005], and lowers the spectrum of the “sh” when compared to the word *sheep*. Another example of the segmentation problem is in the area of image recognition. Lamdan et al. [1988] present the problem of objects in a scene that may be overlapping and partially occluded.

5.3. Feature Extraction

The border between feature extraction and classification is difficult to specify. A feature extractor that gives an ideal representation of a subject would make classification almost unnecessary. Conversely, a classifier that was all powerful would make a feature extractor redundant. The definition of the role of a feature extractor is to characterise an object to be recognised by measurements whose values are very similar for objects in the same category, and very different for objects in a different category (i.e., ideally to extract distinguishing features). The choice of distinguishing features is a critical step and depends on the characteristics of the problem domain. Having prior knowledge can be invaluable when choosing a feature. However, example data for training sets can be equally, if not more, valuable depending on the classification method used [Duda et al. 2000]. Until recently, the most common features extracted for audio processing were the Mel-frequency cepstral coefficients (MFCCs), more specifically speaker recognition, sound classification, and segmentation of audio using sound/speaker identification [Kim et al. 2004].

5.4. Classification

A complete pattern recognition system consists of a sensor that gathers the observations to be classified or described for all feature extraction mechanisms that computes numeric or symbolic information from the observations, and a classification or description scheme that performs classification or observation description, relying on the extracted features. The classification or description scheme is usually based on the availability of a set of patterns that have already been classified or described. This set of patterns is termed the *training set*, and the resulting learning strategy is characterised as supervised learning. Learning can also be unsupervised in the sense that the system is not given an a priori labelling of patterns. Instead, it establishes the classes itself based on the statistical regularities of the patterns. The classification or description scheme typically uses either a statistical or syntactic approach. Statistical pattern recognition is based on statistical characterisations of patterns, assuming that the patterns are generated by a probabilistic system. Syntactic pattern recognition is based on the structural interrelationships of features. A wide range of algorithms can be applied to pattern recognition, from very simple Bayesian classifiers to much more powerful neural networks. Typical applications can include automatic speech recognition, classification of text (e.g., spam email), the automatic recognition of handwriting, or face recognition applications.

5.5. Pattern Matching

Pattern matching is the act of checking for the presence of the constituents of a given pattern. In contrast to pattern recognition/classification, the pattern is rigidly specified. Such a pattern contains either sequences or tree structures. Pattern matching tests whether the data-relevant structure exists, retrieves the aligning parts, and substitutes the matching part with something else. A common application of pattern matching is with text/string patterns. Queries are often posed with regular expressions and matched with respective algorithms. Sequences can also be seen as trees branching for each element into the respective element and the rest of the sequence, or as trees that immediately branch into all elements. Pattern matching is of most benefit when the underlying data structures are as simple and flexible as possible.

5.6. Mel-Frequency Cepstral Coefficients

MFCCs [Stevens et al. 1937] are derived from a cepstral representation of an audio clip. The difference between the cepstrum and the Mel-frequency cepstrum (MFC) is that in the latter the frequency bands are equally spaced on the Mel scale, which approximates the human auditory system's response more closely than the linearly spaced frequency bands used in the normal cepstrum. The cepstrum can be seen as information about rate of change in different spectrum bands. MFCCs often are featured in speech recognition systems, such as systems that can automatically recognise numbers spoken into a telephone. They are primarily used in speaker recognition, which is the task of recognising people from their voices. MFCCs are increasingly applied to MIR applications such as genre classification [Tzanetakis and Cook 2002] and audio similarity measures [Logan and Salomon 2006]. However, as the MFCC is cepstrum based, it is most successful in voice recognition. Voice recognition is divided into two classifications—voice recognition and voice identification—and is the method of automatically identifying who is speaking on the basis of individual information integrated in speech waves. Voice recognition is widely applicable in the use of a speaker's voice to verify his or her identity and control access to services such as banking by telephone, database access services, in-formation services, voicemail, security control for protected information areas, and remote access to computers.

5.7. Clustering

Pattern classifiers typically fall into one of two categories: supervised or unsupervised. A supervised classifier predicts the value of the function for any valid input object after having seen a number of training examples. To achieve this, the classifier has to generalise from the presented data to unseen situations in a reasonable way. Supervised classifiers typically are featured in the following areas:

—*Artificial neural networks*: Artificial neural networks are an abstract simulation of a real nervous system that contains a collection of neuron units communicating with each other via axon connections. Such a model bears a strong resemblance to axons and dendrites in a nervous system. Rabiner [1989] and Frakes and Baeza-Yates [1992] have derived various techniques within the field of information retrieval for pattern recognition.

—*Bayesian statistics*: The main distinguishing feature of a Bayesian approach is that it makes use of more information than non-Bayesian approaches. Whereas the latter are based on analysis of hard data that is well structured and well defined, Bayesian statistics accommodates prior information that usually is less well specified and can even be subjective. Abdallah et al. [2005] report work on music structure extraction with Bayesian probability methods.

—*Nearest neighbour algorithm*: The nearest neighbour algorithm allows a small number of neighbours to influence the classification of an individual value and is a non-parametric classifier. It has also been shown that the error rate of the nearest neighbour algorithm is at most twice as large as the best possible Bayesian error rate [Tzanetakis et al. 2003]. Chuan and Chew [2004] have successfully implemented a nearest neighbour algorithm to determine the key of a polyphonic music piece.

—*Gaussian mixture models*: Gaussian mixtures model the distribution of feature vectors. They widely feature in the MIR community, notably to build timbre models as reported in Tzanetakis and Cook [2002]. In Burred and Lerch [2003], a tree-like structure of Gaussian mixture models the underlying genre taxonomy: a divide-and-conquer strategy first classifies items on a coarse level and then on successively finer levels.

The K-nearest neighbour (KNN) algorithm is a supervised learning algorithm in which the result of a new instance query is classified based on a majority of KNNs. The purpose of this algorithm is to classify a new object based on attributes and training samples. The classifiers do not have any model to fit and are only based on memory. Given a query point, the *k* number of objects (or training points) closest to the query point is found by finding groups of objects such that the objects in a group will be similar to one another and different from the objects in other groups.

5.8. String Matching Algorithms

String matching algorithms are a basic component that is used in the practical implementation of software ranging from operating systems to search tools on Web sites (e.g., Google, Yahoo). The algorithms underlying string matching are not new but have been refined over time to provide more efficient algorithms, as well as algorithms that are more suited to a particular purpose. For example, interest in string searching has increased dramatically within the fields of information retrieval and computational biology owing to the dramatic increase in text/database sizes in need of management. A string can be defined as a sequence of characters over a finite alphabet å. The principal objective of string matching is to find all instances of a string *p* in a large string *T* of the same alphabet. There are three common approaches to the problem [Navarro and Raffinot 2002]. One is to read all of the characters in the text one after the other and at each step update some variable to identify a possible occurrence. This is commonly referred to as the brute force algorithm. Another is to use a sliding window along the text *T* and within the window search backward for an occurrence that matches *p*. The Boyer-Moore algorithm [Boyer and Moore 1977] takes this approach. Finally, the backward DAWG matching (BDM) and backward nondeterministic DAWG matching (BNDM) algorithms by Navarro et al. [1998] are similar to the second approach but are more efficient by also searching for the longest suffix of the window that is also a factor of *p*.

The brute force algorithm consists of checking, at all positions in the text between 0 and n-m, whether an occurrence of the pattern starts there or not. Then, after each attempt, it shifts the pattern by exactly one position to the right. The brute force algorithm requires no preprocessing phase, and a constant extra space of memory in addition to the pattern and the text. During the searching phase, the text character comparisons are done in any order. The time complexity of this searching phase is **O**(mn), with an expected number of 2n character comparisons. The Knuth-Morris-Pratt (KMP) algorithm [Charras and Lecroq 2004] turns a search string into a finite state machine and then runs the machine with the string to be searched as the input string. Execution time is **O**(m+n), where *m* is the length of the search string and *n* is the length of the string to be searched. The KMP algorithm uses information about the characters in the string being searched to determine how much to move along that string after a mismatch occurs. For example given the strings s1 = "aaaabaaaabaaaaab" and s2 = "aaaaa," on the fifth comparison the "a" of s2 does not match the "b" of s1, whereas the brute force algorithm would simply move onto the next character with the mismatch of "b" in s1 found on the fourth comparison. This is avoided in the KMP algorithm by moving $s1_i + 1$ and beginning the comparison again as seen in Figure 7(c).

The fastest known exact string matching algorithms are based on the Boyer-Moore algorithm [Boyer and Moore 1977]. Such algorithms, on average, are sublinear in the sense that it is not necessary to check every symbol in the text, as was the case with KPM in the previous section. The larger the alphabet and the longer the pattern, the faster the algorithm works. It compares characters from right to left, starting with

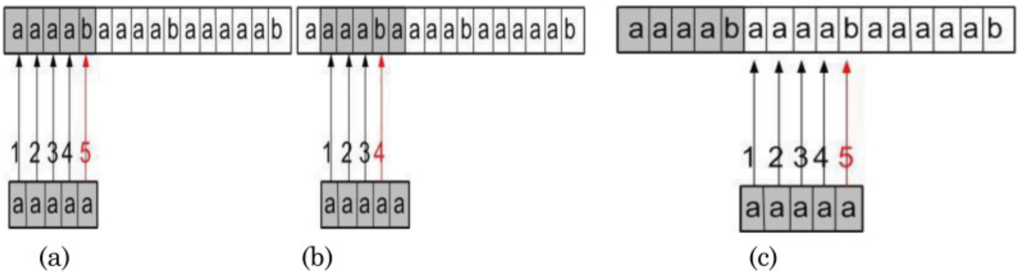


Fig. 7. Example string matching comparison.

the last character in the search pattern. The speed of the Boyer-Moore algorithm is attributed to how well it deals with characters that do not match. When a mismatch is detected, the algorithm checks if the nonmatching character is in the search pattern. If it is not in the pattern, then the pattern is shifted over the entire length of the search pattern.

5.9. Regular Expressions

Regular expressions are incorporated into many text editors, utilities, and programming languages to search and manipulate text based on patterns. For example, the programming languages Perl, Ruby, and Tcl have a powerful regular expression engine built directly into their syntax, and they feature in Unix type systems as the command, grep. As an example of the syntax of a grep query, the regular expression *nbet* can be used to search for all instances of the string *et* that occur after word boundaries (signified by the *nb*). Therefore, in the string “Better than Eternity,” nbet matches the *Et* in “Eternity” but not in “Better,” because the *et* occurs inside a word and not immediately after a word boundary. One of the main differences between regular expressions and other exact string matching algorithms is the use of wildcards. These give a fixed/unknown number of unknown characters to be ignored. For example, in the string “To say that no-one is better than Eternity” in this genre is foolish; the regular expression *n*ter* will return “better” and “Eternity,” as characters preceding the *ter* are ignored. A vast library of functions allow regular expressions extensive flexibility, and it is for this reason that they are also popular within text editor applications. The list of string matching algorithms is endless. The Karp-Rabin algorithm, shift-or algorithm, Simon algorithm, Colussi algorithm, forward DAWG matching algorithm, and Horspool algorithm are but a few [Charras and Lecroq 2004]. Each algorithm is best suited to a particular purpose. An algorithm may have a faster performance depending on the nature of the match, the data being used, the size of the query, or volume of data to be queried.

5.10. Approximate String Matching

The scope of string matching [Boyer and Moore 1977] is rich in problems with substantial mathematical and algorithmic structure. Quite often, these problems are well motivated from an application standpoint. In standard string matching, the problems typically involve finding all occurrences of a pattern string of size m in a larger text string of size n. In these problems, a text location matches a location in the pattern providing that the associated symbols are identical. Approximate string matching differs in that it is not always possible to find an exact match to the query string. One of the best-studied cases of this problem is the so-called edit distance, which allows the deletion, insertion, and substitution of simple characters in both strings. The edit distance has received much attention because its generalised version is powerful enough for a wide range of applications. The edit distance is calculated as D(A,B) between strings A = a_1......am and B = b_1......bn, A,B 2 å (å denotes the set of all sequences over å), where the distance is the minimum number of editing operations, including insertions and deletions of characters within the search string, required to transform string A into string B [Crochemore and Rytter 1994]. A popular approach within the area of MIR is to use edit distance as a measurement of similarity. The work of Hu and Dannenberg [2002], in an investigation of several variations of search algorithms, was aimed towards improving search precision. Their aim was not to find the best matching algorithm for searching but rather to find the best form of music representation by applying edit distance to similarity retrieval based on symbolic notes, pitch, loudness, and tempo, as well as combinations of these together with different windowing approaches. A popular choice of music representation to aid string searches is a monophonic format. Monophonic music can be represented by a one-dimensional string of characters, where each character describes one note or one pair of consecutive notes. Strings can represent interval sequences or sequences of pitches. Lemstrom and Ukkonen [2000] applied an edit distance measure to music comparison and retrieval with a simplified representation of monophonic music that gave only the pitch levels of the notes and ignored note duration.

Hamming distance is a special case of the edit distance similarity metric. Whereas edit distance enables insertion and deletion of characters, Hamming distance can only compare strings of equal length and return only the number of differences between characters, or the number of errors that transformed one string into the other, and not the total difference value. For example, an edit distance measurement will return the value of 4 when measuring the distance of string A and B where A = 5 and B = 1. However, Hamming distance measurement will return a value of 1, as the size of the difference between the values is not taken into account, but only the number of characters that vary. A common example of Hamming distance is given through the use of a cube.

6. SIMILARITY AND CLASSIFICATION OF MUSIC FEATURES

Clustering procedures are essential tools for unsupervised machine learning. Of the array of clustering methods, the k-means clustering is one of the more common choices for solving clustering problems. The choice of starting point of the clusters has a direct result on the outcome. There is no optimum initial cluster positioning, but some work has given consideration to this problem with varying outcomes [Bradley and Fayyad 1998; Zha et al. 2002] and a common rule of thumb, where the initial cluster centroids initialized evenly across the data is the most often proposed solution. Our stance here is not to find some unique, definitive grouping of the ASE output but rather to obtain a qualitative and quantitative understanding of large amounts of N-dimensional MPEG–7 data by finding similarity within it through clustering the audio data. Since similarity is fundamental to the definition of a cluster, a measure of the similarity between two patterns drawn from the same feature space is essential to most clustering procedures. Because of the variety of feature types and scales, the distance measure(s) must be chosen carefully. Often the dissimilarity between two patterns using a distance measure defined on the feature space is calculated. The most popular metric for continuous features is the Euclidean distance. The Euclidean distance has intuitive appeal, as it is often used to evaluate the proximity of objects in two- or three-dimensional space. It works well when a dataset has compact or isolated clusters [Mao et al. 1996]. The drawback to the direct use of Minkowski metrics is the tendency of the largest-scaled feature to dominate the others.

This problem can be seen clearly in Figure 8(b), where clustering of the data is grouped at the middle cluster. Solutions to this problem include normalization of the

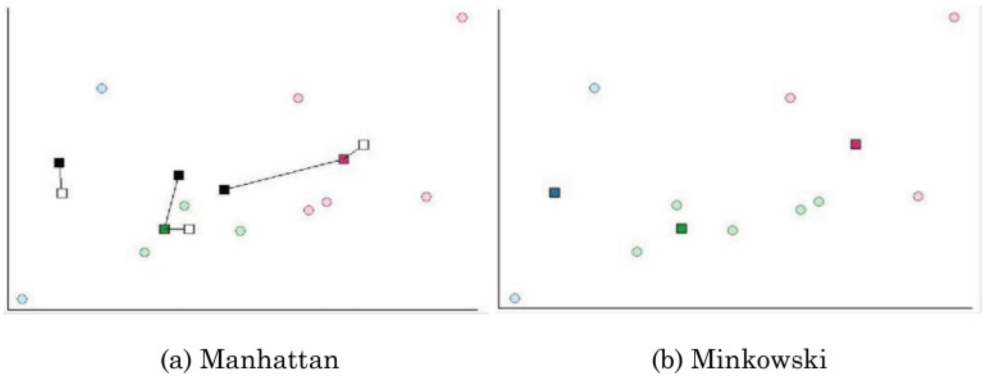


Fig. 8. The k-means distance measures (Manhattan and Minkowski).

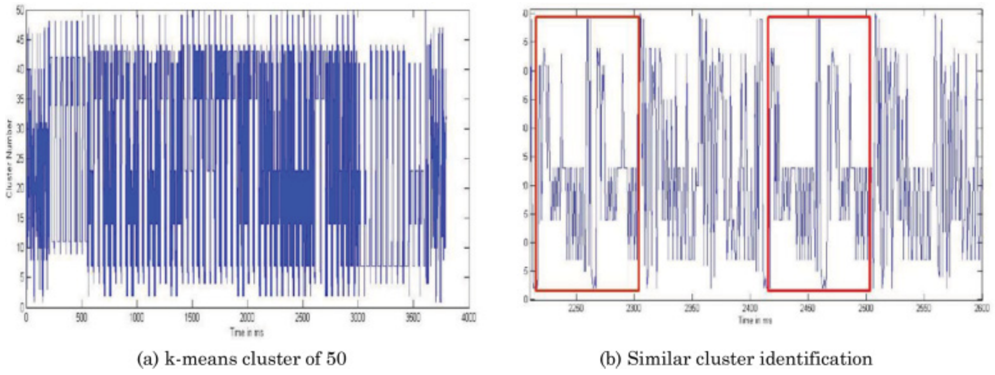


Fig. 9. Example k-means cluster of 50 groups (a) and repeating clusters (b).

continuous features to a common range or variance or other weighting schemes that take into account data with large variances.[2] Applying a cluster of 50 to the sample “12 Bar Blues” audio, the repeating pattern of notes can still be seen clearly, as is shown in Figure 9(a). With closer inspection, the grouping of each 10ms hop can be seen with different repeating sections highlighted in red in Figure 9(b).

With the polyphonic audio now in a clustered format, identification of large sections of audio can be performed with various string matching techniques. Section 3 discussed the various methods of measuring the differences/distance between two fixed length strings that again depend on the nature of the data. Although the clusters presented in Section 4.2 are identified by digits, there is no actual numerical association other than as an identifier, and hence the clusters are presented in a nominal scale. For example, consider the sequence of numbers 1, 2, 3. It can be said that 3 is higher than 2 and 1, whereas 2 is higher than 1. However, during the clustering process when two similar samples are found, they could be as easily identified by using characters or symbols provided that a nominal scale is used. By comparing a string of clusters using the Hamming scale, any metric value is ignored and only the number of differences between the two strings are calculated. However, if a ranking system is applied, then ordinal variables can be transformed into quantitative variables through

[2]Changing the scale can adversely affect the cluster outcome.

normalization. To determine the distance between two objects represented by ordinal variables, it needs to transform the ordinal scale into the ratio scale. This allows the distance to be calculated by treating the ordinal value as quantitative variables and using Euclidean distance, city block distance, Chebyshev distance, Minkowski distance, or the coefficient correlation as distance metrics. Without rank, the most effective measure is the Hamming distance.

7. SONG FORM INTELLIGENCE (SoFI) AUDIO REPAIR FRAMEWORK

We now discuss Song Form Intelligence (SoFI), an intelligent music repair system that repairs dropouts in broadcast audio streams on bursty networks. On the server, the feature extractor analyses the audio from the audio database prior to streaming and creates a results file that is then stored locally on the server ready for the song to be streamed. The streaming media server then streams the relevant similarity file alongside the audio to the client across the network. On the client side, the client receives the broadcast and monitors the network bandwidth for delays of the time-dependent packets. When the level of the internal buffer of the audio stream becomes critically low, the similarity file is accessed to determine the best previously received portion of the song to use as a replacement until the network can recover. The best-matching portion of the song is retrieved from a temporary buffer stored on the client machine specifically for this purpose. In a typical MIR system, similarity assessment is performed in three stages: data reduction, feature extraction, and similarity comparisons. One of the key aspects of feature extraction is to maintain as high a level of reduction as possible without the loss of pertinent data. SoFI makes use of MPEG–7 features in the ASE representation. Songs stored in the database are analysed, and the content description generated from the audio is stored in XML format.

7.1. Clustering the Audio Spectrum Envelope

SoFi uses k-means clustering discussed in Section 4.2 as a method of identifying similarities within different sections of audio. Using a set number of clusters derived from iterative experimentation of the ASE data provides sufficient grouping. The ASE data files contain a varying number of vectors depending on the length of the audio, but considering that each vector contains a finite value in that each sample contains a variable quantity that can be resolved into components, an optimal value of $k = 50$ clusters is used, a sample output of which is shown in Figure 10. This enables a reasonable computational process with the minimum processing power possible while maintaining maximum variety. Experiments above this value produced little or no gain, and with processing time increasing exponentially with each increase in cluster number, it was considered too computationally expensive.

The k-means output results in an array of numbers of 1 ! x, where x is the number of samples in the ASE representation ranging from 1 to 50. A file lasting 30 seconds will result in 3,000 clustered samples, and a file of duration 2 minutes 45 seconds will produce 16,500 clustered samples. At this stage of the similarity computation process, the cognitive representation of music can be construed from the output. Figure 10. Example k-means output.

Figure 11 shows the entire k-means cluster groupings for a full-length audio song. For the human eye, it is difficult to see similarities between sections at this level of detail, but what can be clearly seen is the bridge section in the middle that is dissimilar to any other sections of the audio.

7.2. Similarity Measurement

Having an audio file classified and clustered into groups is the preliminary step in determining similarity between large sections of the file. Where the ASE is a minimalist

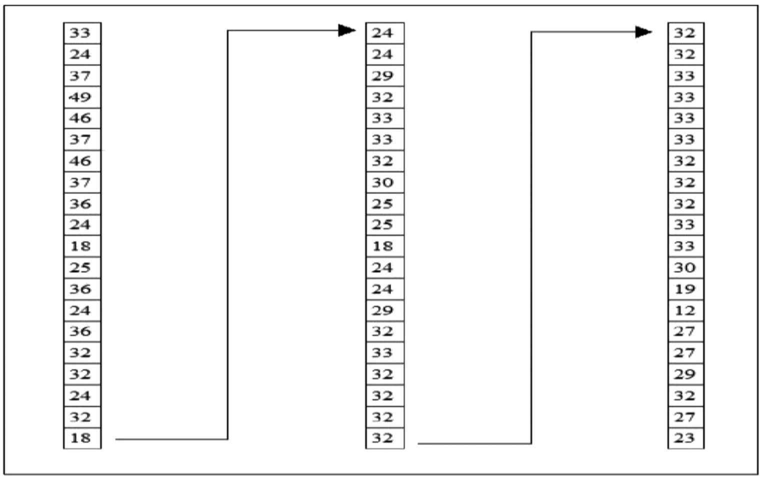


Fig. 10. Example k-means output.

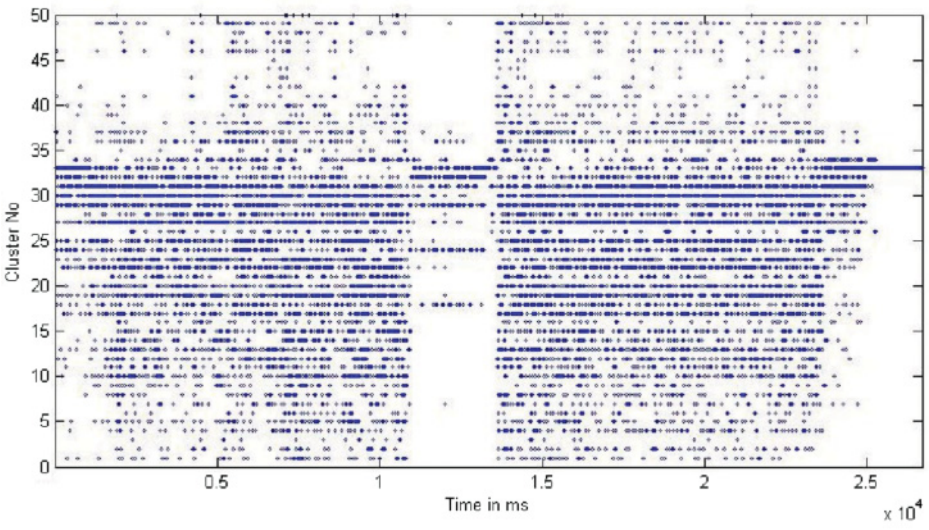


Fig. 11. Example k-means cluster representation of a song.

data representation/description and the k-means grouping is a cluster representation of similar samples at a granular level, SoFI makes use of a traditional string matching approach to identify large sections of audio. The k-means clustering identifies and groups 10ms vectors of audio, but this needs to be expanded to a larger window to facilitate network dropouts. Applying string matching, large sections of the k-means cluster output can be compared for overall similarity and the best-effort match can be stored for reference. This file is then used on the client side for reference at a later time on the client machine when dropouts occur. To reduce unnecessary computation, SoFI only compares the clusters in previous sections for similarities as shown in Figure 12. This is based on the principle that when attempting a repair, SoFI can only use portions of the audio already received. Any sections beyond this have not yet been received by the client and hence cannot be used. This reduces analysis comparisons considerably

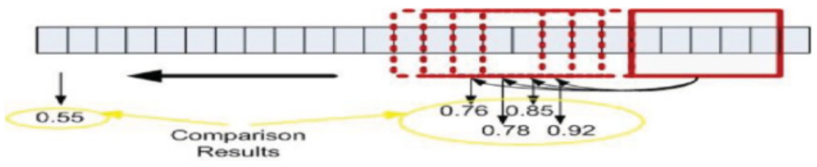


Fig. 12. A backward string matching search.

Table I. Example String Matching Output

Current Time Point	Matching Time Point	Match Result
7.4130000000000000+03	5.4400000000000000e+02	7.1199999999999997e-01
7.4140000000000000e+03	5.4500000000000000e+02	7.1199999999999997e-01
7.4150000000000000e+03	5.4600000000000000e+02	7.1199999999999997e-01
7.4160000000000000e+03	5.4700000000000000e+02	7.1199999999999997e-01

in early sections of the audio; however, as the time point progresses, the number of comparisons increase exponentially.

Sample output in the example given below in Table I shows three different values. The left column is the starting point of the frame to search for, the middle column is the best-match time point of all previous sections, and the last column is the matching result—that is, how close the best match is represented in a scale between zero and one, and the closer to zero the better the match. Layout of the data was initially intended to be in an XML format similar to the MPEG–7 data, but this was considered unnecessary, as there is no change in the data layout throughout the entire content of the file. Incorporating XML tags would allow including metadata for song and artist identification that is already stored in the filename. Incorporating XML tags would also include complexity when parsing the file, increasing processing requirements of the media application.

7.3. Streaming Server

SoFI uses the Ogg Vorbis audio file format as an audio compression tool for preparing files for broadcast. SoFI differentiates between fragmented packets and network traffic congestion. As with any media player, SoFI makes use of the resend request for corrupt individual packets, where one or two packets have time to be re-sent, which will not affect the overall audio output. However, when large dropouts of 5, 10, or 15 seconds occur, this will be unrecoverable and the audio output is affected. Ices2 sends audio data to an Icecast2 server for broadcast to clients. Where Icecast2 handles network connections from clients, Ices2 handles the data to be streamed. Ices2 has a command-line interface; once Ices2 is invoked, it requires no more interaction from the administrator. Ices2 sends the encoded audio to Icecast2 for streaming. The configuration files enable SoFI to be configured for optimum sound quality in music broadcasts, as bandwidth consumption by listeners is not salient. This is because SoFI is a prototype that is not expected to cater to a large number of listeners. Alongside initial configuration settings, Icecast2 has a Web interface that enables administrators to control mountpoints, metadata attached to the audio stream, and the number of listeners on each mountpoint. The main administration page also gives a breakdown of mountpoints detailing quality of the stream, including the audio bit rate, the number of channels, the audio sample rate, and the number of listeners currently connected to that particular mountpoint. From the main administration page, navigation to the mountpoints enables the administrator to control listener access to individual mountpoints, move listeners to another mountpoint and update metadata, and finally stop (kill) a mountpoint. Individual listeners are identified and controlled by navigating to the Listeners tab. Details for each mountpoint list each client connected, the client's IP address, the application that clients are using to listen to the stream, and an option to "kick" a listener from the mountpoint. Ices2 and Icecast2 are an essential to SoFI in that audio streams can be controlled directly for development and testing purposes, including the ability to start, kill, and restart, when required, during development and testing. However, any streaming server would suffice provided that the similarity file either resides on the client machine or is contained in the metadata of the stream prior to commencement of the audio broadcast.

7.4. Network Monitoring

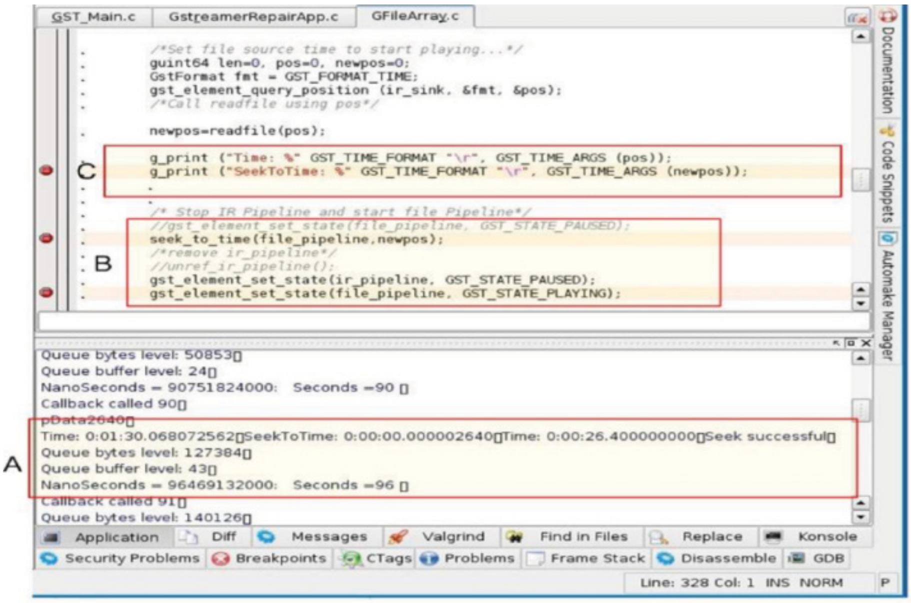
Built into GStreamer is a message bus that constantly handles internal messages between pipelines and handlers. This message system enables alerts to be raised when unexpected events occur, such as end-of-stream (EOS) and low internal buffer levels. A watch method is created to monitor the internal buffer from the audio stream; when a preset critical level is reached, an underrun message is sent to alert the application of imminent network failure. It should be noted that a network failure is not that a network is completely disconnected from the client machine but instead is a network connection that is of such poor signal quality with a low throughput that traffic flow is reduced to an unacceptable level. For the purposes of testing, this was simulated by throttling the bandwidth on the local client machine using the following command under Linux: *sudo tc qdisc add dev eth0 root handle 1:0 netem delay 5000msec 25%.* The qdisc function is the major building block for all Linux traffic control. Using qdisc allows the scheduling of packets between input and output of a particular queue. In the preceding example, the eth0 network input is delayed by 5,000ms with 25% of the incoming traffic suffering from an additional jitter effect. This simulates a typical almost out of range or signal interference scenario attributed to wireless networks and thereby prevents complete network connection failure while at the same time throttling the throughput to almost a failure point.

7.5. SoFI Output

SoFI has a command-line interface and has no graphical user interface. However, feedback is provided through the use of an application window while SoFI is running. This facilitates the notification of events occurring, including EOS, network underrun, and seek events occurring. Figure 13 shows a screenshot of the SoFI media application running with three important sections highlighted in red boxes. The highlighted box A in the figure shows the text output to the listener in the application window. This window is encapsulated in the KDevelop IDE that was used to create and test the SoFI media application. However, when running as a stand-alone application, the terminal window under Linux or the command window under Windows contains the same output: text-based feedback of the current stages of playback.

Synchronised to the gstClock() is a callback function within SoFI and on each cycle of 1 second, a number of statistics are displayed. These include the current level of bytes held in the queue that provides feedback on the current buffer level of network data; the actual level of the buffer in size that can vary from 0% to 100% and nanoseconds providing the precise moment currently being played in the receiving audio stream. Another statistic displayed is seconds which is a conversion from nanoseconds to a more readable format for the listener but it also assists in determining the current number of times feedback has been provided when checked with the preceding output. Callback can provide valuable information in regard to synchronisation problems during playback of audio.

Boxes B and C highlighted in Figure 13 show the relevant method calls for displaying output to the listener when a source swap occurs during a critical network dropout.



```
/*Set file source time to start playing...*/
guint64 len=0, pos=0, newpos=0;
GstFormat fmt = GST_FORMAT_TIME;
gst_element_query_position (ir_sink, &fmt, &pos);
/*Call readfile using pos*/

newpos=readfile(pos);

g_print ("Time: %" GST_TIME_FORMAT "\r", GST_TIME_ARGS (pos));
g_print ("SeekToTime: %" GST_TIME_FORMAT "\r", GST_TIME_ARGS (newpos));
.
.
/* Stop IR Pipeline and start file Pipeline*/
//gst_element_set_state(file_pipeline, GST_STATE_PAUSED);
seek_to_time(file_pipeline,newpos);
/*remove ir_pipeline*/
//unref_ir_pipeline();
gst_element_set_state(ir_pipeline, GST_STATE_PAUSED);
gst_element_set_state(file_pipeline, GST_STATE_PLAYING);
```

```
Queue bytes level: 50853▯
Queue buffer level: 24▯
NanoSeconds = 90751824000:   Seconds =90 ▯
Callback called 90▯
pData2640▯
Time: 0:01:30.068072562▯SeekToTime: 0:00:00.000002640▯Time: 0:00:26.400000000▯Seek successful▯
Queue bytes level: 127384▯
Queue buffer level: 43▯
NanoSeconds = 96469132000:   Seconds =96 ▯
Callback called 91▯
Queue bytes level: 140126▯
```

Fig. 13. SoFI swapping audio sources.

Box C presents the current time played, the seek time of the best match held in the similarity file, and the start time of the playback from the locally stored audio of the radio stream. At the end of this, a message indication of either seek successful or seek failed is displayed. Box B shows the actual states of the ir pipeline and file pipeline being changed from PAUSED and PLAYING, respectively. Figure 13 shows application messages relayed to the user through the application window and the actual method calls changing the playback state of the file pipeline and the ir pipeline. Whereas playback of the audio is from a previous section of the audio stored locally, the ir pipeline maintains a buffer of received audio from the Internet radio stream, and this has to be cleared to ensure as smooth a transition as possible from local playback back to the live stream. As a side effect, a network error is flagged and reported. The EOS message handle deals with the incoming Internet audio stream. When Icecast2 reaches the end of the current song in its playlist, it sends an EOS signal to indicate the end of the current song. This enables Icecast2 to change not only the audio being played but also the audio format to a different format—that is, from .ogg to .mp3 or vice versa. This EOS signals SoFI to close current threads, queues, and pipelines in preparation for the next song to be broadcast. At this point, the previously stored audio is removed and a new locally stored recording of the new incoming audio stream begins. This allows SoFI to begin anew for the new song being received.

8. EVALUATION

Initial investigations into identifying an optimal value for *k* were explored previously, and the number of clusters is set to 50. Values above this offer no gain for the level of detail attained. Figure 14 shows a 5-second sample of audio with clusters of 30, 40, and 50 plotted. The different groupings for each 10ms sample can be seen in both box A and box B. Depending on the number of k clusters specified, each sample will be classified differently. The majority of samples using 30 clusters are shown in green

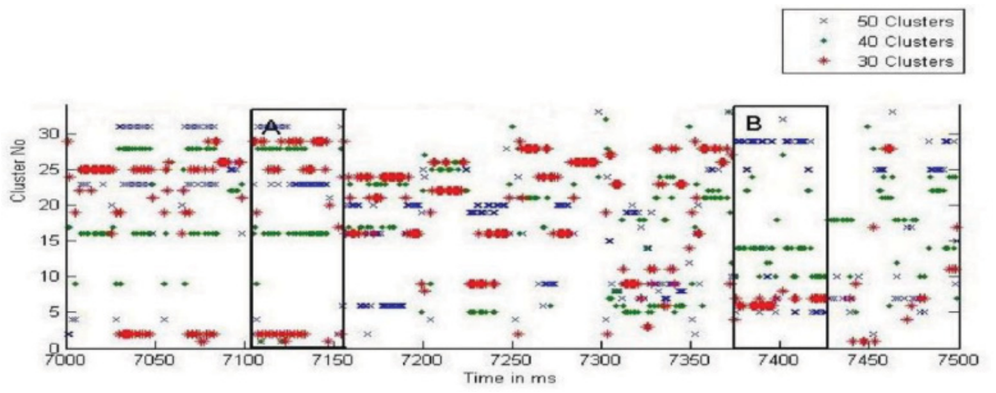


Fig. 14. A comparison of cluster selection.

Table II. The k Cluster Computations Relative to the Size of k

k Clusters	Iterations			
	Song A	Song B	Song C	Song D
30	8040	3270	4410	6240
40	6520	13280	6650	10160
50	8750	12000	11000	13550
60	10260	17040	15060	16800
80	13520	19680	31360	18080
100	19700	30700	39300	45700

Table III. The k Cluster Computations Relative to the Size of k

Song	Audio Properties	
	Duration(s)	Degree of Western Tonal Format
A	3.86	Medium
B	3.86	Medium
C	3.20	High
D	4.43	Low

(*), the red samples (+) are for a value of 40 clusters, and the 50-cluster grouping is shown in blue (x). In box A, a distinct difference between the values can be seen, where the 30-cluster group has been identified as predominantly between 0 and 5. The 30-cluster grouping in box A also has a high number associated with groups at the high end of clusters between 25 and 30, which shows a high level of inconsistency between samples. Although the k cluster number is initially arbitrarily defined, consistency between clusters improves as the number of groupings increase. Box B in Figure 14 shows the k cluster of 50 predominantly classified as the same cluster, whereas the 30 and 40 clusters produced more varied classifications. Tests involving k clusters greater than 50 produced similar results but created large increases in processing time.

Table II shows the number of calculations required based on the chosen value of k clusters. The number of computations does not increase in a linear or exponential manner but is based on the complexity of the music and its composition together with the duration of the audio. Song A requires more calculation due to its composition. Song A is the “12 Bar Blues” sample audio used as a testbed throughout this work. Since it contains a high level of variation between time frames, centroids and distances need to be re-evaluated more frequently. Songs B, C, and D are a random collection from the music collection used within this work. The basic descriptions are presented in Table III. When a choice of k clusters is set to greater than 40, a steady increase of computations can be seen in song A. This is contradicted when using clusters below the level of 40, where songs A, B, C, and D all produce varying results. This can be partially attributed to the limited number of k clusters to which differing values can be assigned [Kriegel et al. 2005]. More variations with fewer clusters mean more comparisons, as

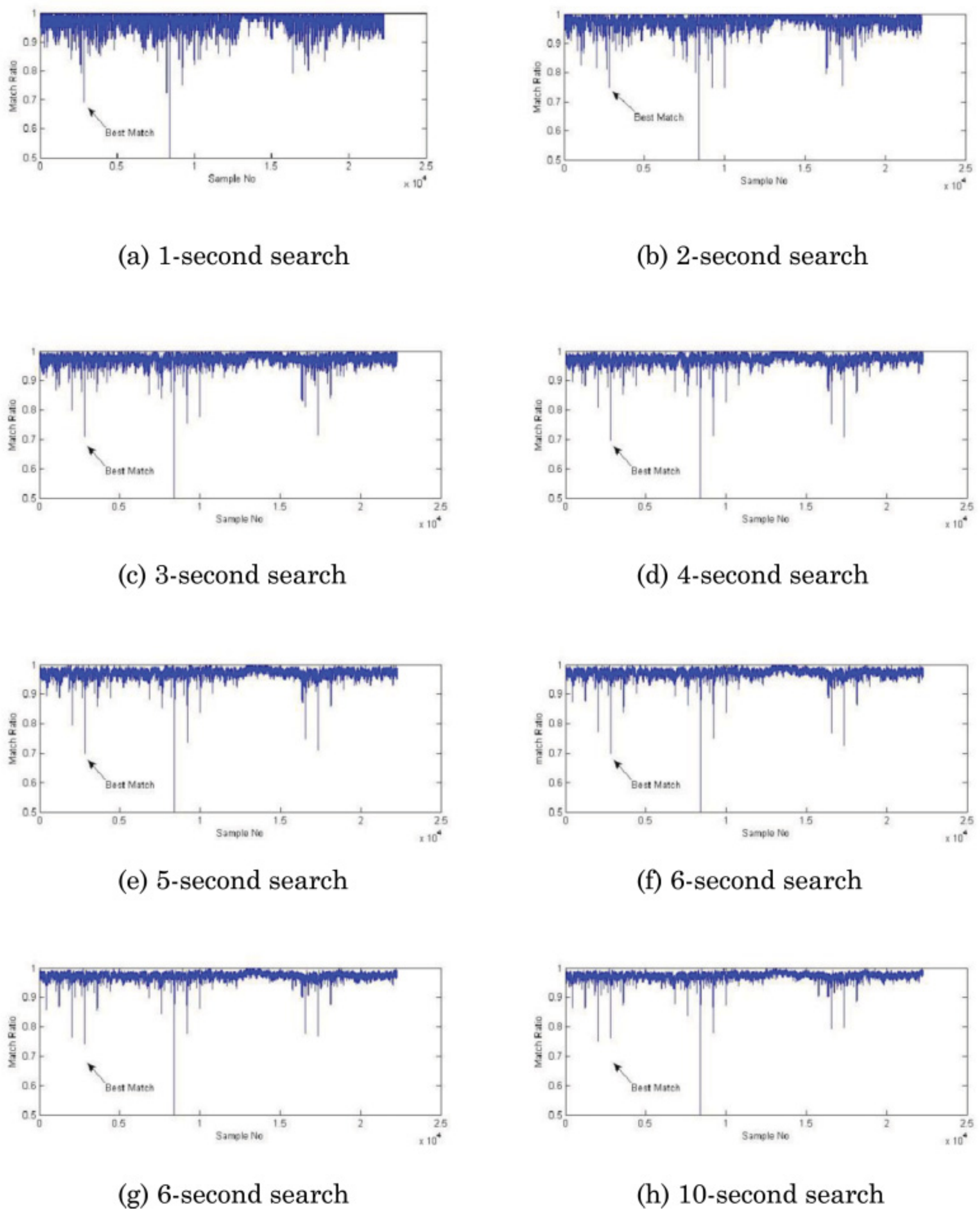


Fig. 15. A string matching comparison: a string length equivalent to 1 second (a), stepped by 1 second (b–f), and stepped by 2 seconds (g and h).

one sample with a high value can offset the centroid of the associated cluster, and this needs to be adjusted more frequently.

8.1. String Matching Large Clusters

Having one 10ms sample classified presents the audio in a readable format that enables larger sections to be compared for similarities. Investigations into string length are shown in Figure 15. Within this figure are 10 graphs showing the results for a complete

Table IV. String Comparison Results for 1 to 10 Seconds

Search String (seconds)	Match Result	Number Matches below 0.8
1	0.6931	6
2	0.7463	5
3	0.7110	5
4	0.6983	4
5	0.7005	4
6	0.7006	4
8	0.7416	3
10	0.7662	5

search of a file given a specific "query." The query in question is a fixed-length string taken from the k-means clustered output, which results in a series of numerical values. Each numerical value is the cluster value for each given time point and can be seen in the following sample output:

. . . 6,2,23,17,42,36,35,16,23,11,35,16,2,6,35,16,6, . . .
. . . ,2,35,41,40,46,42,17, . . .

A query string of 1 second in length contains 100 values, and the entire clustered output of an audio file contains more than 23,000 identified clusters. An example result is where the query string is taken from a random point in the middle of the file without any preconceptions—that is, it is not known whether the query string time point is part of the chorus or a verse or even a bridge. This query string is then compared to the entirety of the clustered file and is noted as to how close a match it is to each segment across all time points from beginning to end.

The closer to a ranked score of zero, the better the match. Within each graph in Figure 15, one clear match value can be seen. This is the time point at which the original query string was sampled and will always give an exact match. Other matching points have clearly been shown as the best match in the first quarter of the song, as indicated in each graph. However, the main focus is on the number of matches across the full duration of the audio file that indicate a clear result with regard to other sections of the audio. Although the "best match" shown in Table IV shows an average match ratio of only around 0.7 in relation to the nearest other matches, this is considerably close.

Also given in Table IV is the number of matches that have been found to be below 0.85. Although the best score in the table is .6931, it can be clearly seen that other sections have been identified as similar, which gives an indication of the repetitiveness of the audio. However, as the initial query string length increases, either the "score" decreases or the number of matches found decreases, giving a reduction in accuracy while determining the best match. Adding to this, the need to replace sections of audio when dropouts greater than 1 second occur eliminates the choice of using 1- or 2-second-length queries as sample criteria while searching for matches. In graphs (a) and (b) of Figure 15, a very close match can be seen marginally to the left of the original query time point. In theory, this could cause problems when trying to repair bursty errors, as it is too close to the live stream time point and the media application will have only a very limited time frame in which to recover the network. Although more possible choices appear, the accuracy is reduced; balance between the extreme lengths of queries as shown in Figure 15 can be seen by using a 5-second length as

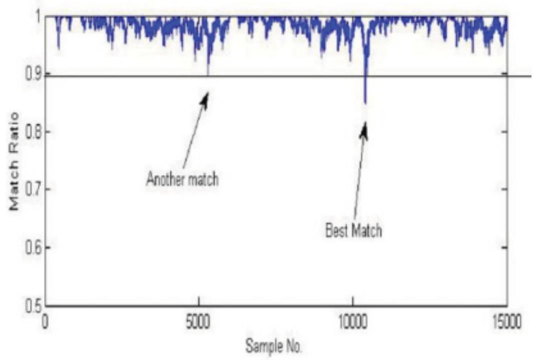


Fig. 16. A 5-second query on only preceding sections.

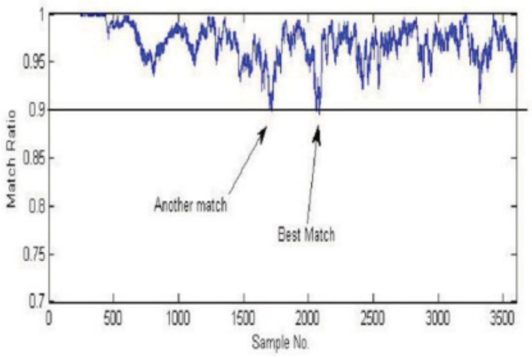


Fig. 17. A 5-second query from only 30 seconds of audio.

shown in part (e) of the figure. The purpose of graphs (a) to (h) in Figure 15 is to show results of matching sections of audio found throughout the audio file. For the purpose of repairing bursty dropouts, the media section of SoFI can use only previously received portions of the live stream up to the point at which network bandwidth/through-put becomes unstable. Figure 16 shows another random time point chosen near the end of a different song than the song used in Figure 15. Using 5 seconds as a query length, a match success can clearly be seen as identified by the best match indicator in Figure 16. Only one other possible match can be seen, and this match has a relatively low match ratio of 0.87. All other comparisons resulted in near and above 0.95—a match at this level would be considered almost unusable.

Figure 17 represents a worst-case scenario where a network dropout occurs near the beginning of a song and also shows the query result of a dropout occurring after 30 seconds of audio have been received. The best match ratio is now only just below 0.89, and only marginally better than any of the other samples. It is this level of a best match that the phrase "best effort" can be used in pattern matching to its fullest. Using this portion of audio as a starting point to replace the break in the live stream will simply mask the error from the listener, although it will be apparent. At this level, the attempted repair is merely to replace a complete loss of signal to minimise the level of distraction caused to the listener. A subjective test by listeners is more a measurement of the success of the replacement than the actual values displayed. Table V shows the average match for cluster string lengths between 1 and 20 seconds. As the time span increases, the accuracy of the match decreases. The interesting point to note in

Table V. Average Match Ratio across All Song Segments

Time (seconds)		Average Match
1		0.6534
2		0.6994
...		....
10		0.7918
12		0.8007
15		0.8121
20		0.8365

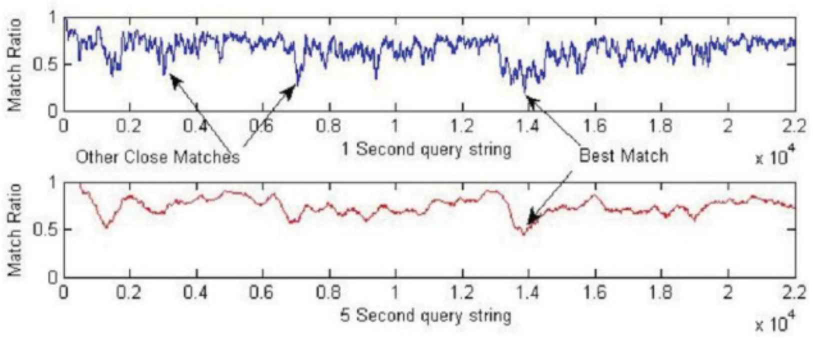


Fig. 18. A comparison of 1- and 5-second query strings.

Table V is the jump between 0.6534 for a 1-second query string and 0.6994 for a 2-second query string. This is due to too many false positives of a match result for such a short query string.

The problem of too many successful matches can be seen more clearly in Figure 18. Both the 1- and 5-second queries returned the same time point as the best possible match for the starting time point of the query. However, additional matches below the best match result using 5 seconds were found with only 1 second of audio. This can lead to sections of audio that may be used, which are not an accurate replacement for dropouts of more than 1 second in length. Using a 5-second length reduces this possibility while increasing the likelihood of the audio following on from the query string time point still being correct.

Of the songs tested, one of the most problematic was by the artist Enya. Although the structure of the songs by Enya are repetitive in principle, they do not strictly adhere to the western tonal format definition. For example, the song "Orinoco Flow" follows the verse/chorus/bridge/verse/chorus structure and yet repeats of the chorus are not composed exactly the same each time they are repeated. This presents problems when matching "chorus" sections as well as verses and bridges. The match ratio expected for a verse is expected to be lower than for a chorus, where the verse usually contains the same underlying music (guitar, drums, piano) in the same repeated manner for different sections. The lyrics, however, can and do change for each verse throughout the song, thereby leading to a lower match percentage. In the case of work by Enya, however, not only do the verses change but the chorus is different as well. To add to the complexity of an uncertain structure, Enya changes the underlying music, although

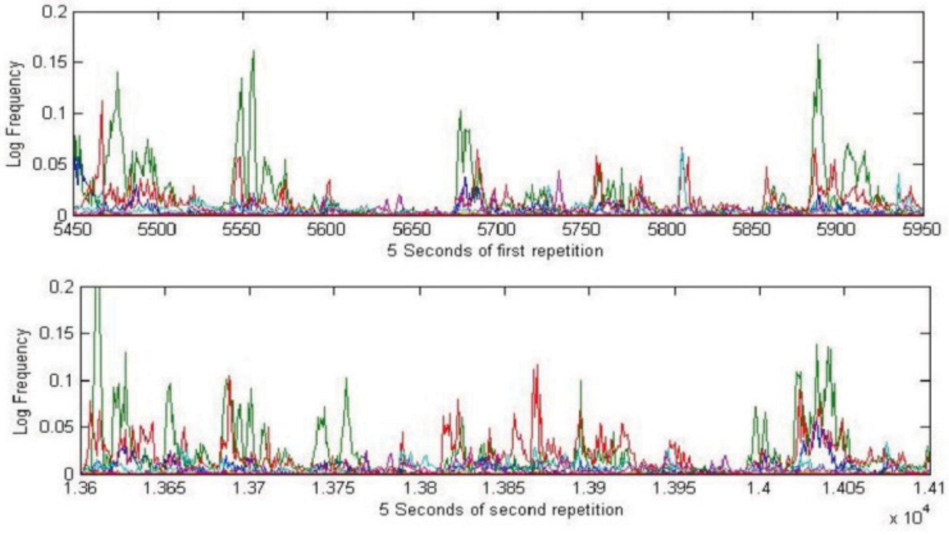


Fig. 19. Two “similar” 5-second segments.

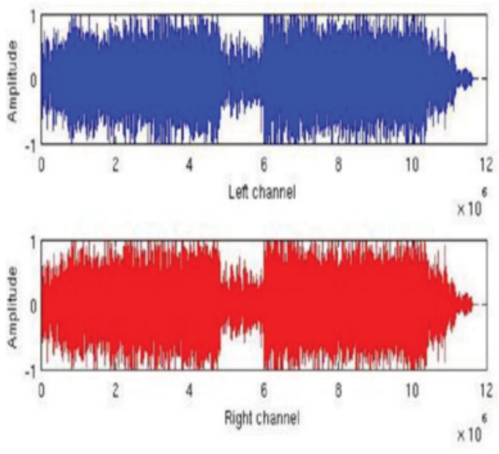


Fig. 20. A two-channel wave audio file.

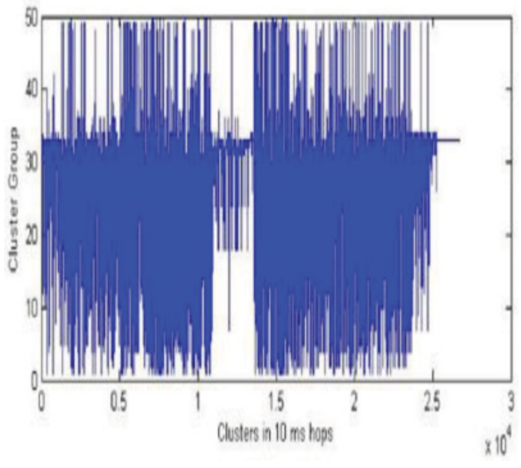


Fig. 21. Figure 18 as a cluster representation.

not the lyrics for each repetition of the chorus. For example, the drum rhythm and guitar rhythm appear “out of sync” compared to other repetitions of the chorus.

In Figure 19, a 5-second segment of the ASE representation of the first chorus of “Orinoco Flow” can be seen. The time point of which it starts is relative to the start of the first lyric in the chorus. When compared to the next time, the same lyrics are repeated, as shown in the lower plot of Figure 19. An overall difference of the audio composition for the equivalent section can be seen. Figures 20 and 21 show the full audio file as a wave representation and clustered format, respectively. Similarities of the overall structure of Enya’s music can be clearly seen. The bridge section is clearly visible in both figures, as well as similarities between the start and end of the song in the way the overall strength of the wave representation is somewhat similarly

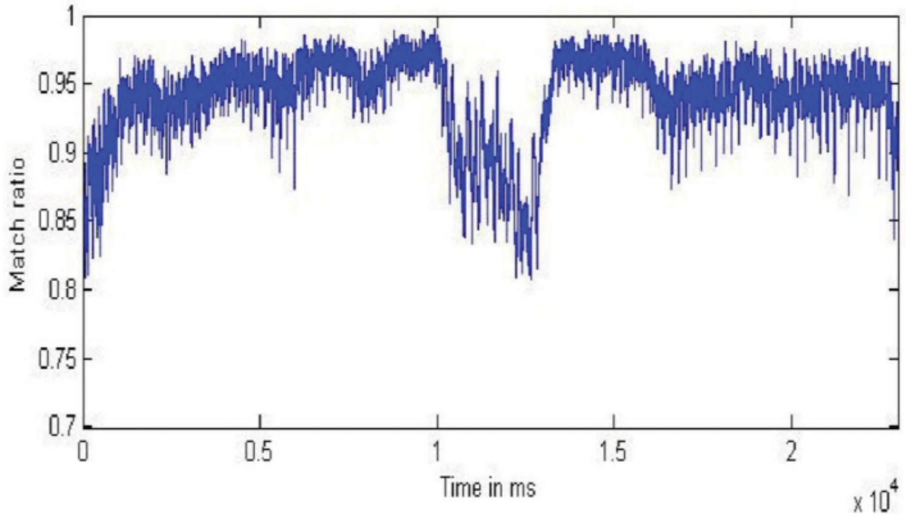


Fig. 22. Match ratio for one 5-second segment from Figure 19.

represented in the clustered representation. It can be implied from this that best-effort results would be similar to previous examples. However, shown in Figure 22 is the match ratio result for the time segments used in Figure 19, and the corresponding best match is not at the most optimal position. The correct time point is actually 10 seconds following on from this point. It can also be seen as a high match ratio at the beginning of the audio, where no lyrics are performed. This could possibly lead to the conclusion that the underlying music has more influence than the vocals in the song, although this would need to be investigated further. To explain the reason for the "misclassification" occurring in the case of this song, the music is timed differently for each different repetition of the lyrics in further sections.

Throughout almost the entirety of "Orinoco Flow," a repetitive music pattern is played; as lyrics change, the music remains the same during both verse and chorus. The only deviation from this pattern occurs during the bridge section. The result of this continuous repetition produces a best match because the music frequencies are more dominant than the lyrics. This leads to a false-positive match where the underlying music is the same but the section matched is not correct.

Table VI shows a comparison of the match ratio for Enya's "Orinoco Flow" alongside the difference between the average match ratio for durations of 1 to 10 seconds. The results "indicate" a better match for time lengths of over 2 seconds, but many of these matches may be false positives. Table VI and Figure 20 show how music that is not strictly in western tonal format can produce what appear to be good match sections but in reality are poor substitutes when better sections should have been identified.

9. CONCLUSION

Evaluation of SoFI using both subjective and objective techniques was performed. Results show a close correlation between similarly identified segments when compared to nonsimilar sections. Subjective evaluations with test subjects from simulated example scenarios show a greater level of acceptance of the audio repair when compared to alternative approaches. Using a Likert scale, the average score across all listening tests gave an acceptance score of 7.09/10, with the maximum score of 8.44/10 achieved when the longest dropout and subsequent repair was presented. A rank of preferred

Table VI. Comparison of Match Ratio across All Song Segments with "Orinoco Flow"

Search String (seconds)	Match Result	Difference from Average
1	0.6737	-0.0194
2	0.7260	-0.0203
3	0.7523	+0.0413
4	0.7680	+0.0697
5	0.7795	+0.0790
6	0.7958	+0.0952
8	0.8145	+0.0729
10	0.8339	+0.0677

"repairs" showed that acceptable repairs can be of a low match depending on the content of the audio. Combining the ASE features with k-means clustering enables similar audio frames to be classified. The MPEG–7 ASE representation enables pertinent data to be retained while reducing the overall data being analysed to a minimum. Clustering the audio into similarly identified groups enables large sections of audio to be analysed and compared. Integrating categorical measurement of distance to determine a best possible match in previous sections within an audio file similarly enables automatic swapping between live audio streams and previously received portions of the audio stored locally in a time- dependent manner. FEC can be performed using previously received audio broadcast across bursty networks without adversely increasing bandwidth, unlike other approaches that attempt to repair network dropouts using "synthesised" or greatly reduced signal quality as replacements. We have provided a novel approach to FEC that combines the latest metadata representation with a classical clustering and string matching technique in an attempt to minimise large dropouts in an audio broadcast on wireless bursty networks.

Most of the related work within the field of MIR uses one of the following forms of signal representation: SVD, PCA, or pitch/frequency detection. However, SoFI utilises as much of the original signal in a condensed form through the MPEG–7 ASE. By retaining the important content across the spectrum, a more informed comparison is made. String matching and clustering within the field of pattern identification and matching is shown to be a reliable and accepted approach. Within the field of MIR systems, similarity classifiers need a sensor that gathers the observations, a feature extraction mechanism that computes numeric or symbolic information from the observations, and a classification scheme that performs the actual classifying. SoFI conforms to these steps through the use of the ASE representation as the sensor, the clustering into groups as a symbolic representation, and by performing string matching comparisons classifies the measures of similarity for each frame. A union between the similarity MIR approach to identifying similar sections and packet loss with network critical–level identification when dropouts occur provides a unique approach to a problem that has until now only been approached on a packet level. Audio is dynamically changed based on the characteristics of network flow, but listeners are provided with a "best possible" alternative with the aim of minimising disruption. SoFI does not use compression, but through implementation of swapping between sources implements a minimal jitter effect at the start and end time points of the swap. This jitter effect is an associated problem within audio playback. However, we have shown how SoFI can be used to reduce potentially large dropouts down to smaller and less noticeable effects. Depending on the accuracy of the replacement, and whether either the start or end segments contain lyrics, jitter can be more apparent. However, through the subjective testing of SoFI, and a comparison with approaches that use this effect, it can be seen that SoFI produces a much more acceptable level even under poor match conditions. With the shift to IP-based streaming across different networks or platforms, the ability to meet audience expectations on quality of service and deliver content continuously will be a key requirement. The bursty nature of IP networks along with variations of performance due to contention will always play against delivering a high quality of service for streamed content to the destination. Any solution that improves the audience experience must add to the overall development of streaming audio over IP as the future of radio. It is in this arena that SOFI has the potential to improve the service to the listener and thus become commercially valuable.

REFERENCES

S. Abdallah, K. Noland, M. Sandler, M. Casey, and C. Rhodes. 2005. Theory and evaluation of a Bayesian music structure extractor. In *Proceedings of the 6th International Conference on Music Information Retrieval*. 420–425.

M. Bartsch and G. Wakefield. 2001. To catch a chorus: Using chroma-based representations for audio thumbnailing. In *Proceedings of the IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*. 15–18.

J. Bilmes and C. Bartels. 2005. Graphical model architectures for speech recognition. *IEEE Signal Processing Magazine* 22, 5, 89–100.

J. Bolot, S. Fosse-Parisis, and D. Towsley. 1999. Adaptive FEC-based error control for Internet telephony. In *Proceedings of INFOCOM'99: The 18th Annual Joint Conference of the IEEE Computer and Communications Societies*. 1453–1460.

R. Boyer and J. Moore. 1977. A fast string searching algorithm. *Communications of the ACM* 20, 10, 762–772.

P. Bradley and U. Fayyad. 1998. Refining initial points for k-means clustering. In *Proceedings of the 15th International Conference on Machine Learning*. 91–99.

J. Burred and A. Lerch. 2003. A hierarchical approach to automatic musical genre classification. In *Proceedings of the 6th International Conference on Digital Audio Effects*. 308–311.

S. Bush. 2000. *Active Jitter Control*. Intelligence in Services and Networks (ISN), London, UK.

A. Cater and N. O'Kennedy. 2000. You hum it, and I'll play it. In *Proceedings of the 11th Conference on Artificial Intelligence and Cognitive Science*.

C. Charras and T. Lecroq. 2004. *Handbook of Exact String Matching Algorithms*. King's College Publications.

Y. Cho and S. Choi. 2005. Nonnegative features of spectro-temporal sounds for classification. *Pattern Recognition Letters* 26, 9, 1327–1336.

C. Chuan and E. Chew. 2004. Polyphonic audio key finding using the spiral array CEG algorithm. In *Proceedings of the International Conference on Multimedia and Expo*.

M. Crochemore and W. Rytter. 1994. *Text Algorithms*. Oxford University Press, New York, USA.

R. Dannenberg and N. Hu. 2003. Pattern discovery techniques for music audio. *Journal of New Music Research* 32, 2, 153–163.

S. Doraisamy and S. Ruger. 2004. A polyphonic music retrieval system using n-grams. In *Proceedings of the International Conference on Music Information Retrieval*. 204–209.

R. Duda, P. Hart, and D. Stork. 2000. *Pattern Classification*. Wiley-Interscience.

T. Eerola and P. Toiviainen. 2004. MIR in MATLAB: The MIDI toolbox. In *Proceedings of the International Conference on Music Information Retrieval*. 22–27.

J. Foote and M. Cooper. 2003. Media segmentation using self-similarity decomposition. *Proceedings of SPIE* 50, 21, 167–175.

W. Frakes and R. Baeza-Yates. 1992. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, Upper Saddle River, NJ.

A. Ghias, J. Logan, D. Chamberlin, and B. Smith. 1995. Query by humming: Musical information retrieval in an audio database. In *Proceedings of the 3rd ACM International Conference on Multimedia*. 231–236.

M. Good. 2001. MusicXML: An Internet-friendly format for sheet music. In *Proceedings of the XML Conference and Expo*. 3–4.

D. Hermes. 1988. Measurement of pitch by subharmonic summation. *Journal of the Acoustical Society of America* 83, 257.

P. Herrera, V. Sandvold, and F. Gouyon. 2004. Percussion-related semantic descriptors of music audio files. In *Proceedings of the AES 25th International Conference*.

N. Hu and R. Dannenberg. 2002. A comparison of melodic database retrieval techniques using sung queries. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*. 301–307.

Humdrum. 2008. The Humdrum Toolkit: Software for Music Research. http://dactyl.som.ohio-state.edu/Humdrum.

Icecast. 2008. Icecast Home Page. Retrieved February 18, 2015, from http://www.icecast.org.

R. Jackendoff. 1987. *Consciousness and the Computational Mind*. MIT Press, Cambridge, MA.

R. Jackendoff. 2002. *Foundations of Language: Brain, Meaning, Grammar*. Oxford University Press.

I. Jackson. 2008. Song forms and terms - a quick study. http://irenejackson.com/songblog/song-forms-and-terms-a-quick-study/.

W. Jiang and H. Schulzrinne. 2002. Comparison and optimization of packet loss repair methods on VoIP perceived quality under bursty loss. In *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. Florida, USA, 73–81.

I. Jolliffe. 1986. *Principal Component Analysis*. Springer-Verlag, New York, NY.

D. Jurafsky and J. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, Upper Saddle River, NJ.

H. Kim, N. Moreau, and T. Sikora. 2004. Audio classification based on MPEG-7 spectral basis representations. *IEEE Transactions on Circuits and Systems for Video Technology* 14, 5, 716–725.

H. Kriegel, P. Kunath, M. Pfeifie, and M. Renz. 2005. Approximated Clustering of Distributed high-dimensional data. In *Advances in Knowledge Discovery and Data Mining*. Lecture Notes in Computer Science, Vol. 3518. 432–441.

F. Kurth, A. Ribbrock, and M. Clausen. 2002. Efficient fault tolerant search techniques for full-text audio retrieval. In *Proceedings of the AES Convention*.

G. Lakoff. 1988. Cognitive semantics. In *Meaning and Mental Representations*, U. Eco (Ed.). Indiana University Press, Bloomington, IN, 119–154.

Y. Lamdan, J. Schwartz, and H. Wolfson. 1988. Object recognition by affine invariant matching. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*. 335–344.

K. Lee and S. Chanson. 2004. Packet loss probability for bursty wireless real-time traffic through delay model. *IEEE Transactions on Vehicular Technology* 53, 3, 929–938.

M. Leman, L. Clarisse, B. De Baets, H. De Meyer, M. Lesaffre, G. Martens, J. Martens, and D. Van Steelant. 2002. *Tendencies, Perspectives, and Opportunities of Musical Audio-Mining*. Forum Acusticum Sevilla.

K. Lemstrom and E. Ukkonen. 2000. Including interval encoding into edit distance based music comparison and retrieval. In *Proceedings of the AISB Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*. 53–60.

K. Lemstrom, V. Makinen, A. Pienimaki, M. Turkia and E. Ukkonen. 2003. The C-BRAHMS project. In *Proceedings of the 4th International Conference on Music Information Retrieval.* 237–238.

F. Lerdahl and R. Jackendoff. 1983. *A Generative Theory of Tonal Music*. MIT Press, Cambridge, MA.

Y. Liang, N. Farber, and B. Girod. 2003. Adaptive playout scheduling and loss concealment for voice communication over IP networks. *IEEE Transactions on Multimedia* 5, 4, 532–543.

R. Likert. 1932. A technique for the measurement of attitudes. *Archives of Psychology* 22, 140, 1–55.

S. Lin, D. Costello, and M. Miller. 1984. Automatic-repeat-request error-control schemes. *IEEE Communications Magazine* 22, 12, 5–17.

B. Logan. 2000. Mel frequency cepstral coefficients for music modelling. In *Proceedings of the International Symposium on Music Information Retrieval*.

B. Logan and A. Salomon. 2001. A music similarity function based on signal analysis. In *Proceedings of the IEEE International Conference on Multimedia and Expo*.

J. Lukasiak, D. Stirling, N. Harders, and S. Perrow. 2003. Performance of mpeg-7 low level audio descriptors with compressed data. In *Proceedings of the International Conference on Multimedia and Expo.* 273–276.

R. Lyons. 2004. *Understanding Digital Signal Processing*. Prentice Hall, Upper Saddle River, NJ.

A. Mahanti, D. Eager, M. Vernon, and D. Sundaram-Stukel. 2003. Scalable on-demand media streaming with packet loss recovery. *IEEE/ACM Transactions on Networking* 11, 2, 195–209.

J. Mao, A. Jain, I. Center, and C. San Jose. 1996. A self-organizing network for hyperellipsoidal clustering (HEC). *IEEE Transactions on Neural Networks* 7, 1, 16–29.

J. Martinez, R. Koenen, and F. Pereira. 2002. MPEG-7: The generic multimedia content description standard, part 1. *IEEE MultiMedia* 9, 2, 78–87.

R. McNab, L. Smith, D. Bainbridge, and I. Witten. 1997. The New Zealand digital library MELody inDEX. *D-Lib Magazine* 3, 5, 4–15.

E. Menin. 2002. *The Streaming Media Handbook*. Pearson Education.

D. Meredith, G. Wiggins, and K. Lemstrom. 2001. Pattern induction and matching in polyphonic music and other multidimensional datasets. In *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI'01)*. London, UK, 22–25.

D. Meredith, K. Lemstrom, and G. Wiggins. 2002. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research* 31, 4, 321–345.

G. Navarro and M. Raffinot. 2002. *Flexible Pattern Matching in Strings: Practical On-Line Search Algorithms for Texts and Biological Sequences*. Cambridge University Press.

G. Navarro, M. Raffinot, and M. Farach-Colton. 1998. A bit-parallel approach to suffix automata: Fast extended string matching. In *Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching*. 14–33.

A. Ockelford. 1991. The role of repetition in perceived musical structures. In *Representing Musical Structure*, P. Howell, R. West, and I. Cross (Eds.). Academic Press, London, England, 129–160.

D. Parsons. 1975. Directory of Tunes and Musical Themes. Oxford Press, UK.

S. Pauws. 2002. Cubyhum: A fully operational query by humming system. In *Proceedings of the 3rd International Conference on Music Information Retrieval*. 187–196.

M. Pearce and G. Wiggins. 2002. Aspects of a cognitive theory of creativity in musical composition. In *Proceedings of the ECAI Workshop on Creative Systems*. 17–24.

C. Perkins, O. Hodson, and V. Hardman. 1998. A survey of packet loss recovery techniques for streaming audio. *IEEE Network* 12, 5, 40–48.

L. Prechelt and R. Typke. 2001. An interface for melody input. *ACM Transactions on Computer-Human Interaction* 8, 2, 133–149.

L. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77, 2, 257–286.

F. Salzer. 1962. *Structural Hearing: Tonal Coherence in Music*. Dover Publications.

H. Sapp and B. Aarden. 2008. Themefinder. Available at http://www.themefinder.org.

M. Schedl, T. Pohle, P. Knees, and G. Widmer. 2011. Exploring the music similarity space on the Web. *ACM Transactions on Information Systems* 29, 3, Article No. 14.

E. Schubert, J. Wolfe, and A. Tarnopolsky. 2004. Spectral centroid and timbre in complex, multiple instrumental textures. In *Proceedings of the 8th International Conference on Music Perception and Cognition.*

J. Seo, M. Jin, S. Lee, D. Jang, S. Lee, and C. Yoo. 2005. Audio fingerprinting based on normalized spectral subband centroids. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. 213–216.

Semantic. 2008. Semantic Interaction with Audio Contents. Retrieved August 6, 2008, from http://www.semanticaudio.co.uk/.

M. Slaney, I. Center, and C. San Jose. 2002. Semantic-audio retrieval. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing.* 4108–4111.

T. Socolofsky and C. Kale. 2008. A TCP/IP Tutorial. Retrieved February 18, 2015 from http://www.ietf.org/rfc/rfc1180.txt.

W. Stevens. 1993. TCP/IP Illustrated Volume 1: The Protocols. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

S. Stevens, J. Volkmann, and E. Newman. 1937. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America* 8, 1, 185–194.

H. Sze, S. Liew, and Y. Lee. 2001. A packet-loss-recovery scheme for continuous-media streaming over the Internet. *IEEE Communications Letters* 5, 3, 116–118.

A. Tanguiane. 1993. *Artificial Perception and Music Recognition*. Springer, Berlin.

D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. 2007. Towards musical query-by-semantic-description using the CAL500 data set. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 439–446.

D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. 2008. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech, and Language Processing* 16, 2, 467–476.

G. Tzanetakis and P. Cook. 2002. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing* 10, 5, 293–302.

G. Tzanetakis, A. Ermolinskyi, and P. Cook. 2003. Pitch histograms in audio and symbolic music information retrieval. *Journal of New Music Research* 32, 2, 143–152.

S. Varadarajan, H. Ngo, and J. Srivastava. 2002. Error spreading: A perception-driven approach to handling error in continuous media streaming. *IEEE/ACM Transactions on Networking* 10, 1, 139–152.

B. Wah and D. Lin. 2005. LSP-based multiple-description coding for real-time low bit-rate voice over IP. *IEEE Transactions on Multimedia* 7, 1, 167–178.

R. Walker. 1997. Visual metaphors as music notations for sung vowel spectra in different cultures. *Journal of New Music Research* 26, 4, 315–345.

H. Wallach. 2004. Evaluation metrics for hard classifiers. Unpublished Note. Available at http://www.inference.phy.cam.ac.uk/hmw26/papers/evaluation.ps.

Y. Wang, A. Ahmaniemi, D. Isherwood, and W. Huang. 2003. Content-based UEP: A new scheme for packet loss recovery in music streaming. In *Proceedings of the 11th ACM International Conference on Multimedia*. 412–421.

E. Wold, T. Blum, D. Keislar, and J. Wheaten. 1996. Content-based classification, search, and retrieval of audio. *IEEE Multimedia* 3, 3, 27–36.

G. Wiggins. 1998. Music, syntax, and the meaning of "meaning." In *Proceedings of the 1st Symposium on Music and Computers*.

G. P. Williams. 1997. *Chaos Theory Tamed*. Taylor and Francis, London, England.

H. Zha, X. He, C. Ding, M. Gu, and H. Simon. 2002. Spectral relaxation for k-means clustering. *Advances in Neural Information Processing Systems* 2, 1, 1057–1064.

B. Zhang, J. Shen, Q. Xiang, and Y. Wang. 2009. CompositeMap: A novel framework for music similarity measure. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 403–410.

Received July 2013; revised August 2014; accepted August 2014