US 20110035035A1

(19) **United States**

(12) **Patent Application Publication**

Khan et al.

(10) **Pub. No.: US 2011/0035035 A1**

(43) **Pub. Date: Feb. 10, 2011**

(54) **METHOD AND SYSTEM FOR ANALYZING DIGITAL AUDIO FILES**

(75) Inventors: **Rehan M. Khan**, Lafayette, CA (US); **George Tzanetakis**, Victoria (CA)

Correspondence Address:
FITZPATRICK CELLA HARPER & SCINTO
1290 Avenue of the Americas
NEW YORK, NY 10104-3800 (US)

(73) Assignee: **ROVI TECHNOLOGIES CORPORATION**, Santa Clara, CA (US)

(21) Appl. No.: **12/906,584**

(22) Filed: **Oct. 18, 2010**

Related U.S. Application Data

(63) Continuation of application No. 11/839,768, filed on Aug. 16, 2007, now Pat. No. 7,853,344, which is a continuation of application No. 09/695,457, filed on Oct. 24, 2000, now Pat. No. 7,277,766.

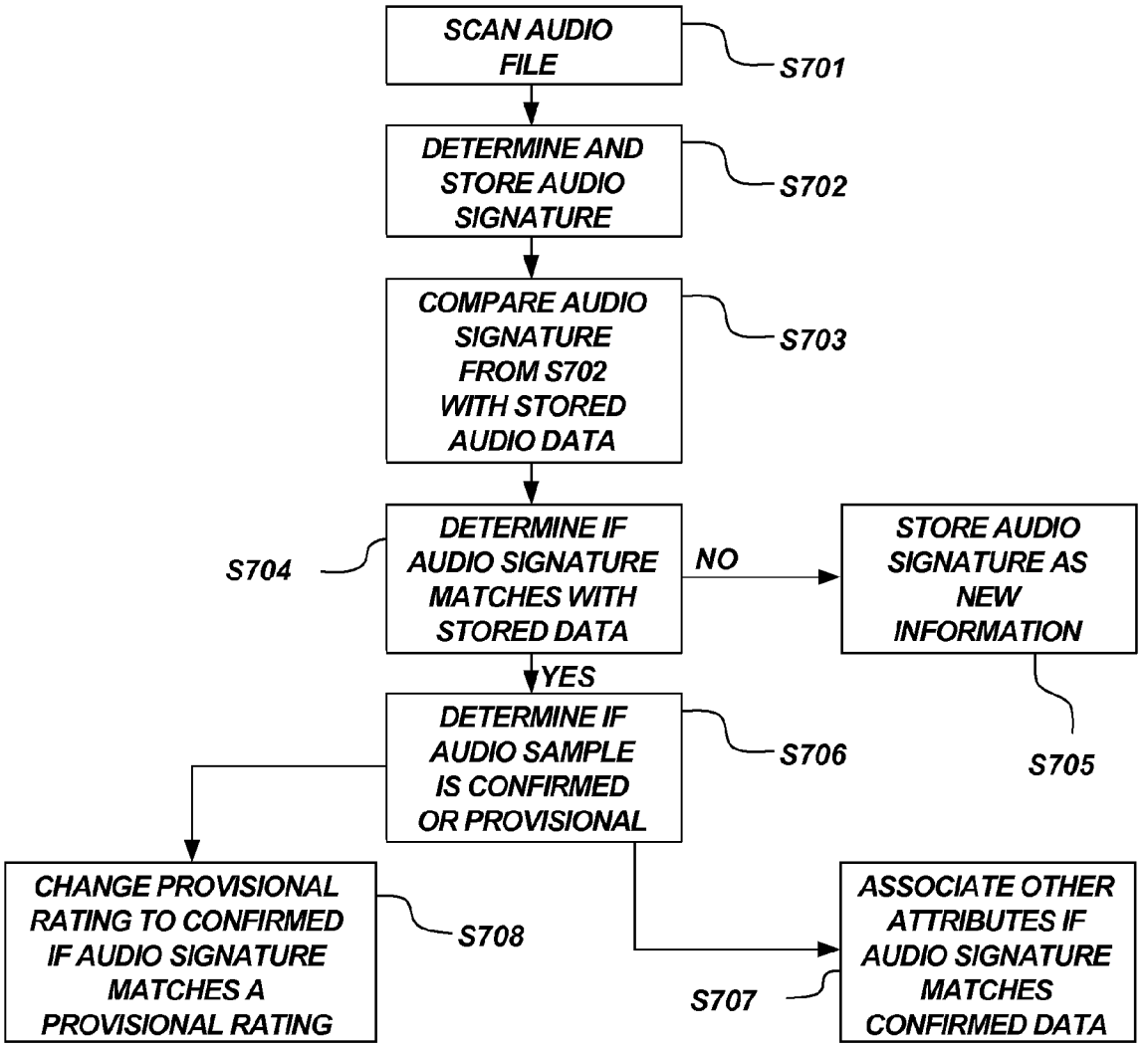
Publication Classification

(51) **Int. Cl.**
G06F 17/00 (2006.01)

(52) **U.S. Cl.** **700/94**

(57) **ABSTRACT**

A fingerprint is generated from an unknown audio signal by dividing the unknown audio signal into bins, where each bin includes points representing a feature space. Each of the points is mapped to one of a plurality of predetermined cluster centers based on the distance between each point and the plurality of cluster centers, each cluster center being associated with an element of a codebook. A string of elements is generated based on the mapping and compressed.



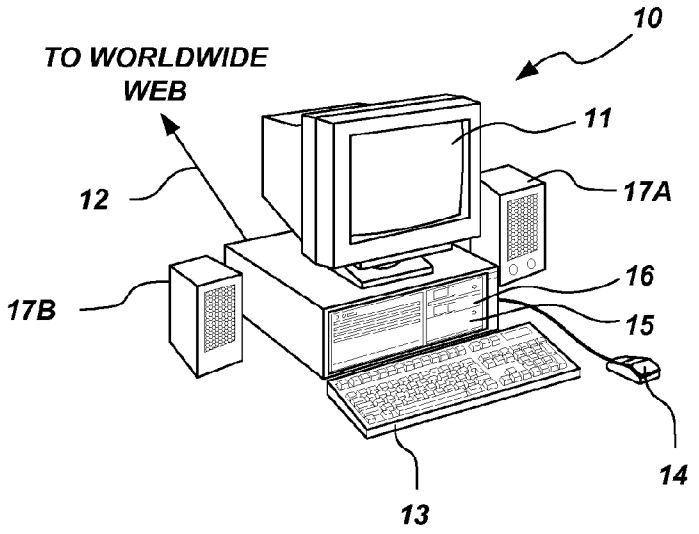

10
TO WORLDWIDE WEB
11
12
17A
16
17B
15
14
13

FIG. 1

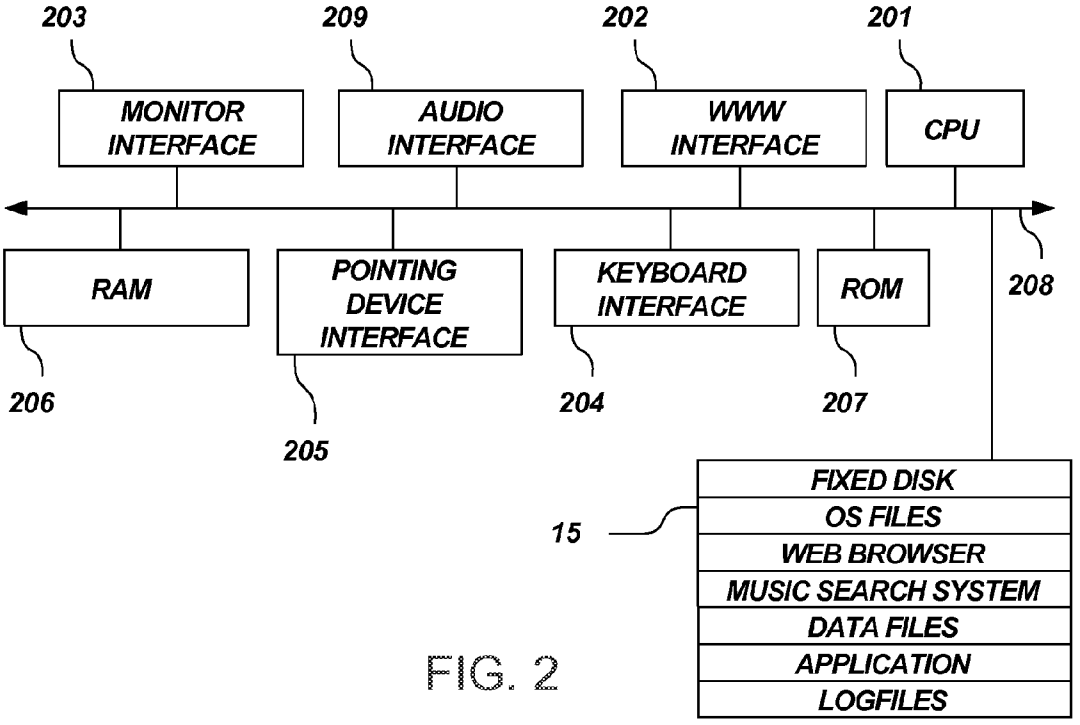

203
209
202
201
MONITOR INTERFACE
AUDIO INTERFACE
WWW INTERFACE
CPU
RAM
POINTING DEVICE INTERFACE
KEYBOARD INTERFACE
ROM
208
206
205
204
207
15
FIXED DISK
OS FILES
WEB BROWSER
MUSIC SEARCH SYSTEM
DATA FILES
APPLICATION
LOGFILES

FIG. 2

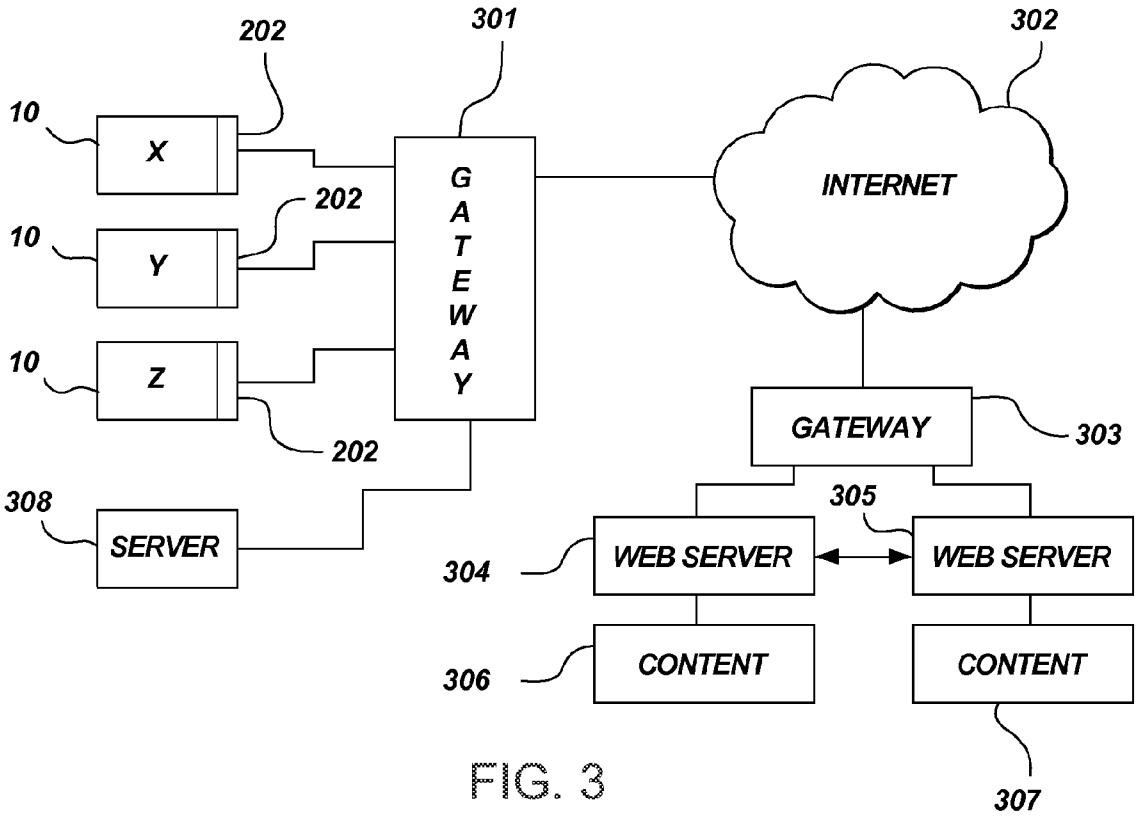

202
301
302
10
X
202
INTERNET
10
Y
GATEWAY
202
10
Z
202
GATEWAY
303
308
SERVER
305
304
WEB SERVER
WEB SERVER
306
CONTENT
CONTENT
307

FIG. 3

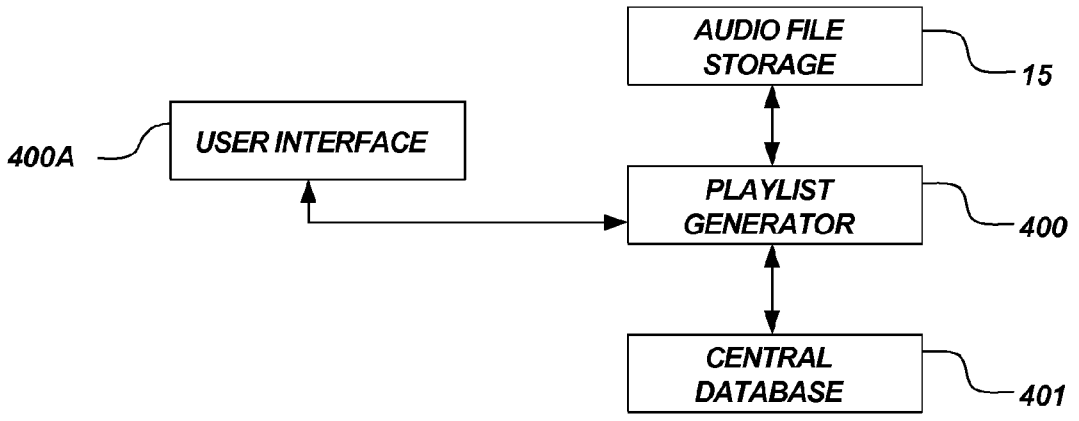

AUDIO FILE STORAGE
15
400A
USER INTERFACE
PLAYLIST GENERATOR
400
CENTRAL DATABASE
401

FIG. 4

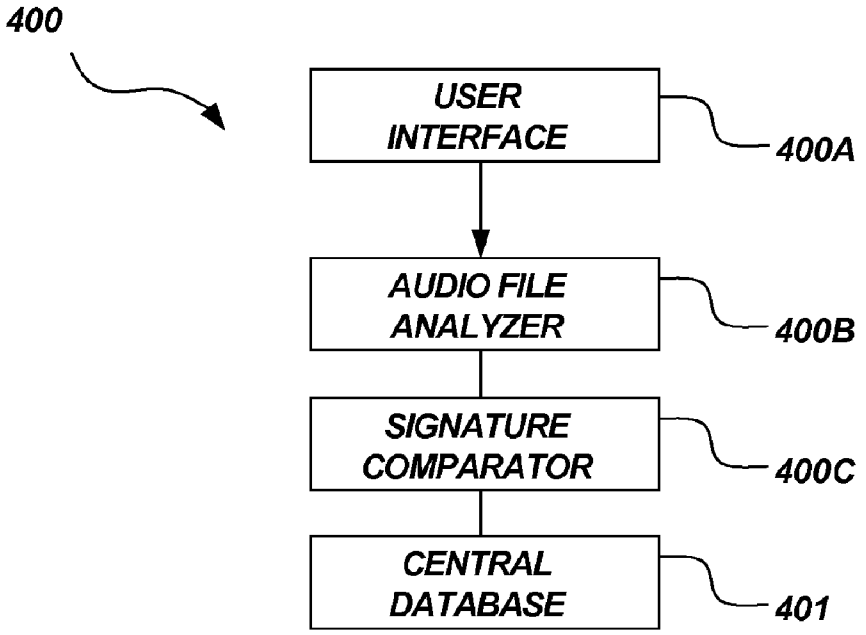

400
USER INTERFACE
400A
AUDIO FILE ANALYZER
400B
SIGNATURE COMPARATOR
400C
CENTRAL DATABASE
401

FIG. 5

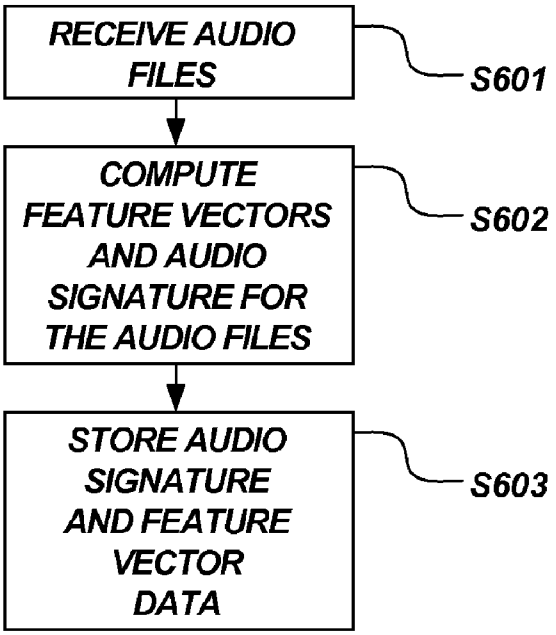

RECEIVE AUDIO FILES
S601
COMPUTE FEATURE VECTORS AND AUDIO SIGNATURE FOR THE AUDIO FILES
S602
STORE AUDIO SIGNATURE AND FEATURE VECTOR DATA
S603

FIG. 6

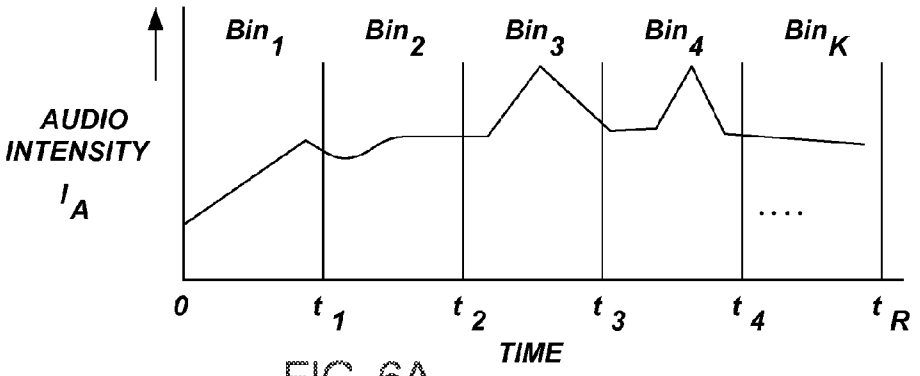

Bin 1
Bin 2
Bin 3
Bin 4
Bin K
AUDIO INTENSITY
I A
....
0
t 1
t 2
t 3
t 4
t R
TIME

FIG. 6A

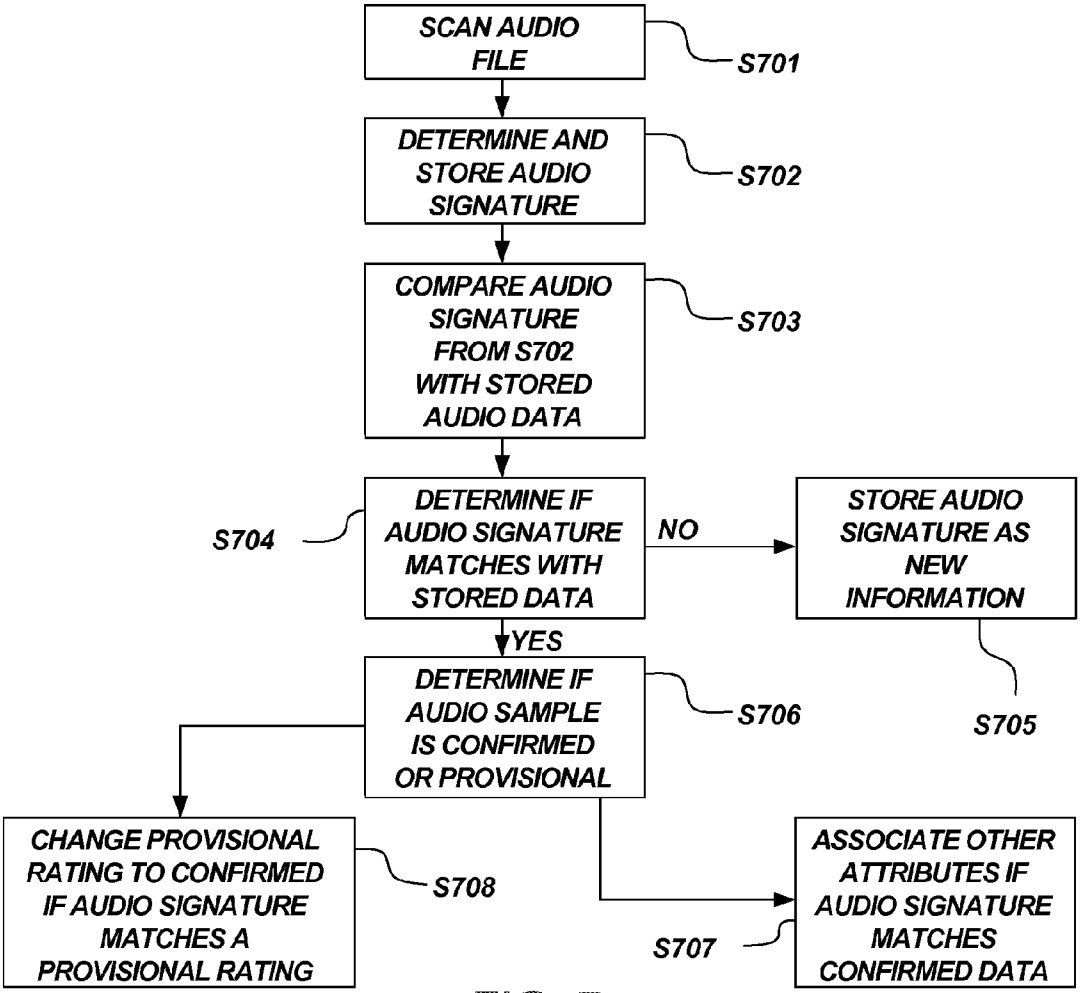

SCAN AUDIO FILE
S701
DETERMINE AND STORE AUDIO SIGNATURE
S702
COMPARE AUDIO SIGNATURE FROM S702 WITH STORED AUDIO DATA
S703
S704
DETERMINE IF AUDIO SIGNATURE MATCHES WITH STORED DATA
NO
STORE AUDIO SIGNATURE AS NEW INFORMATION
S705
YES
DETERMINE IF AUDIO SAMPLE IS CONFIRMED OR PROVISIONAL
S706
CHANGE PROVISIONAL RATING TO CONFIRMED IF AUDIO SIGNATURE MATCHES A PROVISIONAL RATING
S708
S707
ASSOCIATE OTHER ATTRIBUTES IF AUDIO SIGNATURE MATCHES CONFIRMED DATA

FIG. 7

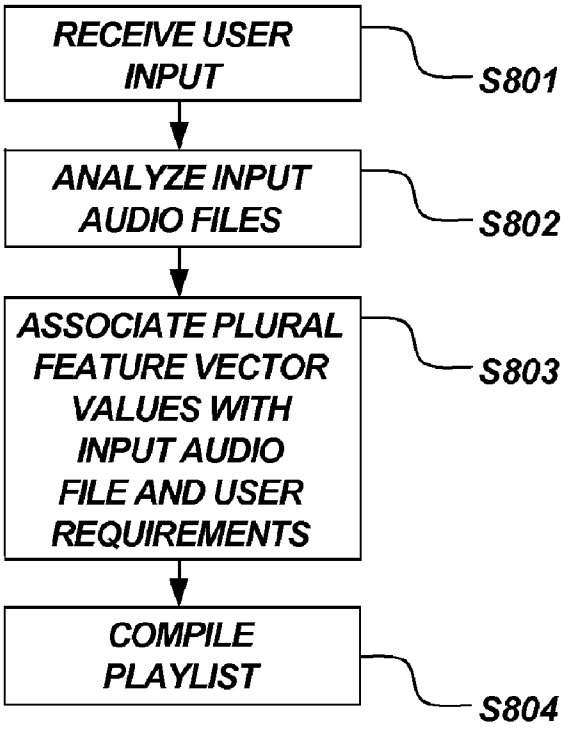

RECEIVE USER INPUT
S801
ANALYZE INPUT AUDIO FILES
S802
ASSOCIATE PLURAL FEATURE VECTOR VALUES WITH INPUT AUDIO FILE AND USER REQUIREMENTS
S803
COMPILE PLAYLIST
S804

FIG. 8

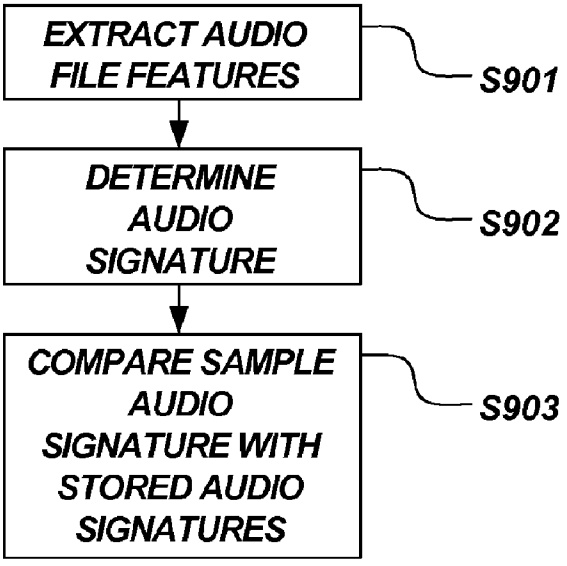

EXTRACT AUDIO FILE FEATURES
S901
DETERMINE AUDIO SIGNATURE
S902
COMPARE SAMPLE AUDIO SIGNATURE WITH STORED AUDIO SIGNATURES
S903

FIG. 9

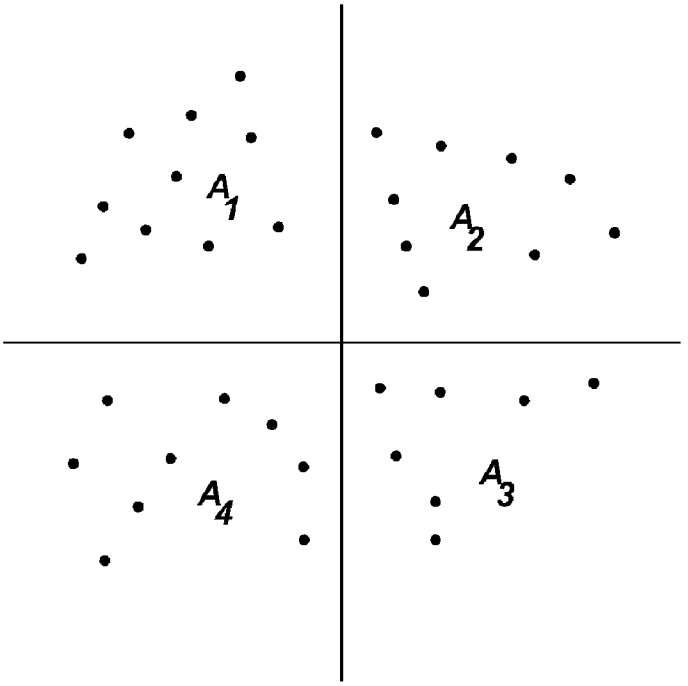


FIG. 10A

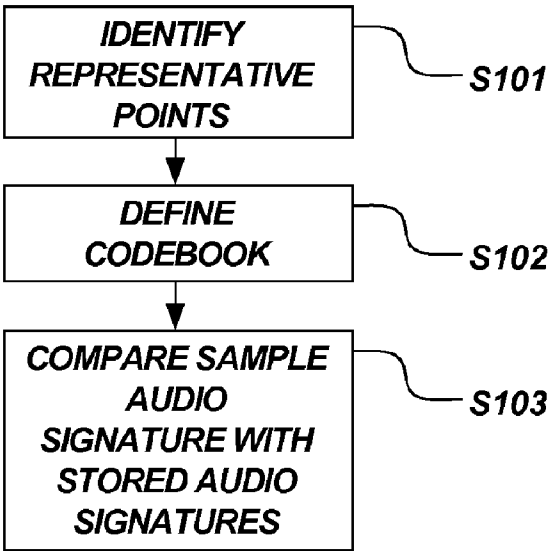


FIG. 10B

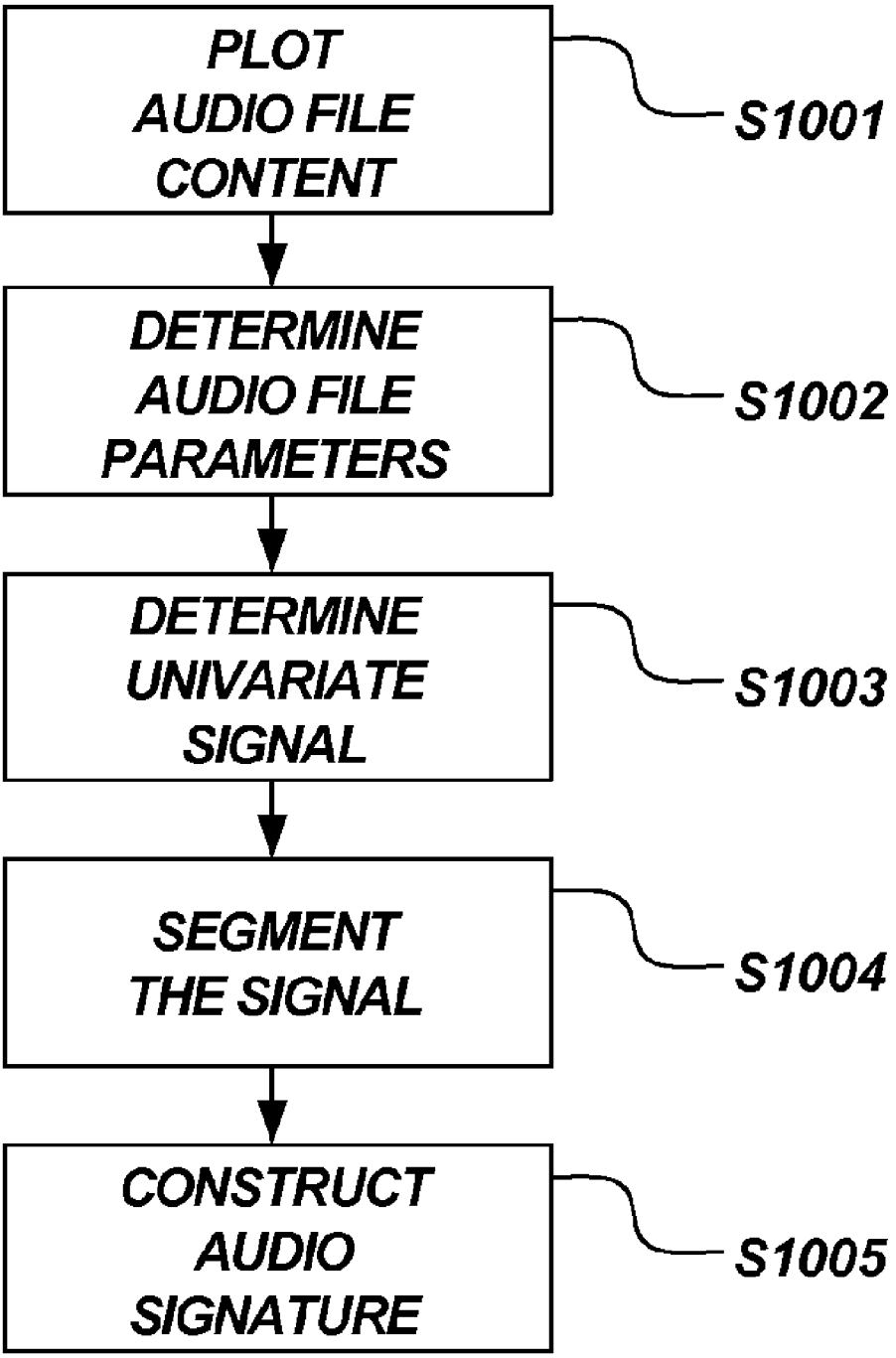

PLOT AUDIO FILE CONTENT
S1001
DETERMINE AUDIO FILE PARAMETERS
S1002
DETERMINE UNIVARIATE SIGNAL
S1003
SEGMENT THE SIGNAL
S1004
CONSTRUCT AUDIO SIGNATURE
S1005

FIG. 10C

METHOD AND SYSTEM FOR ANALYZING DIGITAL AUDIO FILES

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] The present application is a continuation of U.S. application Ser. No. 11/839,768, filed on Aug. 16, 2007, now U.S. Pat. No. _____, entitled "METHOD AND SYSTEM FOR ANALYZING DIGITAL AUDIO FILES", issued on _____, which is a continuation of U.S. patent application Ser. No. 09/695,457, filed on Oct. 24, 2000, now U.S. Pat. No. 7,277,766, entitled "METHOD AND SYSTEM FOR ANALYZING DIGITAL AUDIO FILES", issued on Oct. 2, 2007, both of which are incorporated herein by reference in their entirety.

BACKGROUND OF THE INVENTION

[0002] 1. Field of the Invention

[0003] The present invention relates to analyzing audio files and more particularly to presenting a playlist based upon listener preferences and audio file content.

[0004] 2. Background

[0005] The Internet connects thousands of computers world wide into a vast network using well-known protocols, for example, Transmission Control Protocol (TCP)/Internet Protocol (IP). Information on the Internet is stored world wide as computer files, mostly written in the Hypertext Mark Up Language ("HTML"). The collection of all such publicly available computer files is known as the World Wide Web (WWW).

[0006] The WWW is a multimedia-enabled hypertext system used for navigating the Internet and is made up of hundreds of thousands of web pages with audio, images, text and video files. Each web page can have connections to other pages, which may be located on any computer connected to the Internet.

[0007] A typical Internet user uses a client program called a "Web Browser" to connect to the Internet. A user can connect to the Internet via a proprietary network, such as America Online or CompuServe, or via an Internet Service Provider, e.g., Earthlink.

[0008] A Web Browser may run on any computer connected to the Internet. Currently, various browsers are available of which two prominent browsers are Netscape Navigator and Microsoft Internet Explorer. The Web Browser receives and sends requests to a web server and acquires information from the WWW. A web server is a program that, upon receipt of a request, sends the requested data to the requesting user.

[0009] A standard naming convention known as Uniform Resource Locator ("URL") has been adopted to represent hypermedia links and links to network services. Most files or services can be represented with a URL. URLs enable Web Browsers to go directly to any file held on any WWW server.

[0010] Information from the WWW is accessed using well-known protocols, including the Hypertext Transport Protocol ("HTTP"), the Wide Area Information Service ("WAIS") and the File Transport Protocol ("FTP"), over TCP/IP protocol. The transfer format for standard WWW pages is Hypertext Transfer Protocol (HTTP).

[0011] The advent and progress of the Internet has changed the way consumers buy or listen to music. Consumers today can download digital music via the Internet using MP3 or SDMI technology, with a click of a mouse. Audio delivery techniques have also made it easy to stream audio from a website to a consumer, upon demand. A typical music listener can download audio files from the WWW, store the audio files, and listen to music.

[0012] Currently music can be stored in various file formats. Generally there are two types of file formats: (1) self-describing formats, where device parameters and encoding are made explicit in a header, and (2) headerless formats, where device parameters and encoding are fixed.

[0013] The header of self-describing formats contain parameters of a sampling device and may also include other information (e.g. a human-readable description of sound, or a copyright notice etc.). Some examples of popular self describing formats are provided below:

File Extension Variable Parameters (Fixed; Comments)

[0014]

au or .snd	rate, #channels, encoding, info string
aif(f), AIFF	rate, #channels, sample width, lots of info
aif(f), AIFC	same (extension of AIFF with compression)
iff, IFF/8SVX	rate, #channels, instrument info (8 bits)
mp2, .mp3	rate, #channels, sample quality
.ra	rate, #channels, sample quality
.sf	rate, #channels, encoding, info
.smp	loops, cues, (16 bits/1 ch)
.voc	rate (8 bits/1 ch; can use silence deletion)
.wav, WAVE	rate, #channels, sample width, lots of info

[0015] Headerless formats define single encoding and usually allow no variation in device parameters (except sometimes for sampling rates). The following are a few examples of Headerless formats:

Extension	Parameters or name
.snd, .fssd	Variable rate, 1 channel, 8 bits unsigned
.ul	8 k, 1 channel, 8 bit "u-law" encoding
.snd	Variable rate, 1 channel, 8 bits signed

[0016] Although music listeners can store audio files, conventional music search techniques do not allow a music listener to search for music based upon audio file content. Conventional systems also do not allow a music listener to generate play lists based upon music listener preferences and/or audio file content.

[0017] Hence what is needed is a method and system that can analyze audio file content and produce a play list based upon preferences defined by a music listener.

SUMMARY

[0018] The present invention solves the foregoing drawbacks by providing a method and system for analyzing audio files. Plural audio file feature vector values based on an audio file's content are determined and the audio file feature vectors are stored in a database, that also stores other pre-computed audio file features. The process determines if the audio file's feature vectors match the stored audio file vectors. The process also associates a plurality of known attributes to the audio file.

[0019] The present invention includes a system for analyzing audio files that includes a playlist generator that determines a plurality of audio file vectors based upon an audio file's content; and a signature comparator between the playlist generator and a database, wherein the database stores a plurality of audio file vector values of plural music samples. The signature comparator compares input audio samples with previously stored audio samples in the database. Also provided is a user interface that allows a music listener to input search request for searching music based upon attributes that define music content.

[0020] In another aspect, the present invention includes a method for determining audio signatures for input audio samples. The process extracts plural features representing the input audio samples, wherein the features are extracted by Fourier transform analysis. The process also identifies a set of representative points based upon the plural features, and determines a codebook of plural elements for mapping the representative points to the elements of the codebook.

[0021] In yet another aspect, the present invention includes a method for comparing input audio signatures with pre-computed stored audio signatures. The process determines a query signature based upon the input audio signature and divides the query signature into a string of characters; and compares the string of characters to stored pre-computed audio signatures.

[0022] In yet another aspect, the present invention divides an input audio sample into bins and determines a plurality of features describing the bins. Thereafter, the process determines a univariate signal based upon the plural features and computes an audio signature based upon the univariate signal.

[0023] One advantage of the foregoing aspects of the present invention is that unique audio signatures may be assigned to audio files. Also various attributes may be tagged to audio files. The present invention can generate a customized playlist for a user based upon audio file content and the attached attributes, hence making the music searching experience easy and customized.

[0024] This brief summary has been provided so that the nature of the invention may be understood quickly. A more complete understanding of the invention can be obtained by reference to the following detailed description of the preferred embodiments thereof in connection with the attached drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0025] FIG. **1** illustrates a computing system to carry out the inventive technique.

[0026] FIG. **2** is a block diagram of the architecture of the computing system of FIG. **1**.

[0027] FIG. **3** is a block diagram of the Internet Topology.

[0028] FIG. **4** is a block diagram of the architecture of the present system.

[0029] FIG. **5** is a block diagram showing the architecture of a playlist generator.

[0030] FIG. **6** is a flow diagram of computer executable process steps for analyzing an audio file.

[0031] FIG. **6**A is a graphical illustration of an audio file's content.

[0032] FIG. **7** is a flow diagram of computer executable process steps for comparing input audio files with stored audio data.

[0033] FIG. **8** is a flow diagram of computer executable process steps of generating a playlist, according to the present invention.

[0034] FIG. **9** is a flow diagram of computer executable process steps for determining audio signatures based upon one aspect of the present invention.

[0035] FIG. **10**A shows a set of representative points randomly scattered and used for vector quantization, according to another aspect of the present invention.

[0036] FIG. **10**B is a flow diagram of computer executable process steps for performing vector quantization, according to another aspect of the present invention.

[0037] FIG. **10**C is a flow diagram of computer executable process steps for determining audio signatures, according to yet another aspect of the present invention.

[0038] The use of similar reference numerals in different figures indicates similar or identical items.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0039] FIG. **1** is a block diagram of a computing system for executing computer executable process steps according to one embodiment of the present invention. FIG. **1** includes a host computer **10** and a monitor **11**. Monitor **11** may be a CRT type, a LCD type, or any other type of color or monochrome display. Also provided with computer **10** is a keyboard **13** for entering text data and user commands, and a pointing device **14** for processing objects displayed on monitor **11**.

[0040] Computer **10** includes a computer-readable memory medium such as a rotating disk **15** for storing readable data. Besides other programs, disk **15** can store application programs including web browsers by which computer **10** connects to the Internet and the systems according to the present invention as described below.

[0041] Computer **10** can also access a computer-readable floppy disk storing data files, application program files, and computer executable process steps embodying the present invention or the like via a floppy disk drive **16**. A CD-ROM interface (not shown) may also be provided with computer **10** to access application program files, audio files and data files stored on a CD-ROM.

[0042] A modem, an integrated services digital network (ISDN) connection, or the like also provides computer **10** with an Internet connection **12** to the World Wide Web (WWW). The Internet connection **12** allows computer **10** to download data files, audio files, application program files and computer-executable process steps embodying the present invention.

[0043] Computer **10** is also provided with external audio speakers **17**A and **17**B to assist a listener to listen to music either on-line, downloaded from the Internet or off-line using a CD (not shown). It is noteworthy that a listener may use headphones instead of audio speakers **17**A and **17**B to listen to music.

[0044] FIG. **2** is a block diagram showing the internal functional architecture of computer **10**. Computer **10** includes a CPU **201** for executing computer-executable process steps and interfaces with a computer bus **208**. Also shown in FIG. **2** are a WWW interface **202**, a display device interface **203**, a keyboard interface **204**, a pointing device interface **205**, an audio interface **209**, and a rotating disk **15**. Audio Interface **209** allows a listener to listen to music, On-line (downloaded using the Internet or a private network) or off-line (using a CD, not shown).

[0045] As described above, disk **15** stores operating system program files, application program files, web browsers, and other files. Some of these files are stored on disk **15** using an installation program. For example, CPU **201** executes computer-executable process steps of an installation program so that CPU **201** can properly execute application programs.

[0046] A random access main memory ("RAM") **206** also interfaces to computer bus **208** to provide CPU **201** with access to memory storage. When executing stored computer-executable process steps from disk **15** (or other storage media such as floppy disk **16** or WWW connection **12**), CPU **201** stores and executes the process steps out of RAM **206**.

[0047] Read only memory ("ROM") **207** is provided to store invariant instruction sequences such as start-up instruction sequences or basic input/output operating system (BIOS) sequences for operation of keyboard **13**.

[0048] The present invention is not limited to the computer architecture described above. Systems comparable to Computer **10**, for example, portable devices or hand held computing devices that can be connected to the Internet may also be used to implement the present inventive techniques.

[0049] FIG. **3** shows a typical topology of a computer network with computers similar to computer **10**, connected to the Internet. For illustration purposes, three computers X, Y and Z are shown connected to the Internet **302** via Web interface **202**, through gateway **301**, where gateway **301** can interface numerous computers. Web interface **202** may be a modem, network interface card or a unit for providing connectivity to other computer systems over a network using protocols such as X.25, Ethernet or TCP/IP, or to any device that allows direct or indirect computer-to-computer communications.

[0050] It is noteworthy that the invention is not limited to a particular number of computers. Any number of computers that can be connected to the Internet **302** or to any other computer network may be used to implement the present inventive techniques.

[0051] FIG. **3** further also shows a second gateway **303** that connects a network of web servers **304** and **305** to the Internet **302**. Web servers **304** and **305** may be connected with each other over a computer network. Web servers **304** and **305** can provide content including music samples and audio clips to a user from database **306** and/or **307**. Web servers **304** and **305** can also host the system according to the present invention. Also shown in FIG. **3** is a client side web server **308** that can be provided by an Internet service provider.

[0052] FIG. **4** shows a block diagram of a system used for analyzing audio files, according to the present invention. Audio files may be stored on rotating disk **15** in a music listener's computer **10**, or at any remote computer **10** connected to the Internet **302**.

[0053] A playlist generator **400** accesses audio files stored on rotating disk **15**. Playlist generator **400** is an application program that can be located on remote server **304** or on a music listener's computer **10**. Playlist generator **400** scans and analyzes audio files stored on rotating disk **15** and computes audio signatures that uniquely and compactly identify the content of the audio file. An audio signature is computed only once for each file and stored in a local database. The computed audio signature that compares the analyzed audio files with previously analyzed audio data is stored in a central database **401**. Central database **401** stores a plurality of feature vector values as discussed below. Central database **401** also includes data similar to the data stored in a production database that is described in U.S. patent application Ser. No. 09/533,045, entitled, "Method for Creating a Database for Comparing Music Attributes", incorporated herein by reference in its entirety. Customized play lists are generated after audio file content is compared to the data stored in central database **401**, as described below.

[0054] A user interface **400**A is also provided that allows a music listener to input preferences for generating play lists. User interface **400**A may be a separate component or integrated with playlist generator **400**. One such user interface is described in U.S. patent application Ser. No. 09/533,045, entitled "Method for Creating a Database for Comparing Music Attributes", incorporated herein by reference in its entirety.

[0055] FIG. **5** is a block diagram showing various components of playlist generator **400**. Playlist generator **400** includes an audio file analyzer **400**B and a Signature Comparator **400**C. Audio file analyzer **400**B receives and analyzes audio files as described below. Audio files may be stored on rotating disk **15** or may be acquired from a remote computer.

[0056] The Signature Comparator **400**C receives user requests from UI **400**A and obtains music or a list of music based upon user preferences, as described below. It is noteworthy that audio file analyzer **400**B and signature comparator **400**C may be integrated into a single module to determine audio signature and implement the various aspects of the present invention.

[0057] FIG. **6** is a process flow diagram of computer executable process steps for analyzing audio files according to the present invention.

[0058] In step S**601** audio analyzer **400**B receives audio files ("input audio files"). Audio files may be stored on a user's disk **15** or at webserver **304** connected to the Internet or on any private network. Audio analyzer **400**B may seek audio files based upon user input in UI **400**A or may receive audio files from a designated source.

[0059] In step S**602**, audio analyzer **400**B analyzes input audio file content and computes a set of parameters ("audio file vectors"). Audio file vectors can uniquely identify audio files and can also be used to assign a unique "audio signature" for an input audio file. Details of determining audio signatures are described below. These audio signatures need be computed only once for a given audio file and can thereafter be stored either locally or remotely and referred to as necessary.

[0060] In step S**603**, input audio file vector values for specific bins and the audio signature are stored in a database. One such database is a central database **401**. Audio file signatures are sets or vectors of values based upon the audio segments as shown in FIG. **6**A.

[0061] FIG. **7** is a flowchart of computer executable process steps that allows playlist generator **400** to compare audio file content with stored and analyzed audio file data. Such audio files may be stored on rotating disk **15** or on a remote computer.

[0062] In step S**701**, playlist generator **400** scans an audio file(s) stored on disk **15**. Audio files may be pulled by playlist generator **400** from disk **15**, or a music listener may send audio files to playlist generator **400** via the Internet **302** or a private network.

[0063] In step S**702**, playlist generator **400** determines the audio signature of input audio file(s). An audio signature may be determined by the process described below.

[0064] In step S**703**, playlist generator **400** transfers the audio signature and audio file vector values ($V_{t1} \ldots V_{tk}$) to

Signature Comparator **400**C. Playlist generator **400** also commands the Signature Comparator **400**C to compare the audio signature and feature vectors values determined in step S**702** with historical audio signatures and audio file vector values stored in central database **401**. Thereafter, the Signature Comparator **400**C compares input audio signature and audio file vector values with historical audio signatures and audio file vector values, stored in database **401**.

[0065] In step S**704**, Signature Comparator **400**C determines whether the input audio signature and audio file vectors values match stored audio signatures and vector values. If the input audio file vector values do not match with any stored audio data (audio signature and feature vector), then in step S**705** input audio file vector values are stored in central database **401**.

[0066] If the input audio file signature matches with stored audio signatures and vector values, then in step S**706**, the process determines if the stored entries are confirmed or provisional. A confirmed entry in database **401** is an entry that has been ratified by multiple sources.

[0067] A provisional entry in database **401** is one that is not confirmed.

[0068] If the input audio file matches a confirmed entry, then in step S**707** other feature values stored in database **401** are associated with the audio file. Examples of such feature vectors are provided in U.S. patent application Ser. No. 09/533,045, entitled "Method for Creating a Database for Comparing Music Attributes", filed on Mar. 22, 2000, assigned to the assignee herein, and incorporated by reference in its entirety. Associating a plurality of feature vectors allows a listener to search for music based upon content. For example, feature values associated with a given audio file may indicate that it is Rock music with a slow tempo and a smooth, female singer accompanied by a band featuring a prominent saxophone.

[0069] Some of the features that can be associated with the audio files are:

(a) Emotional quality vector values that indicate whether an audio file content is Intense, Happy, Sad Mellow, Romantic, Heartbreaking, Aggressive or Upbeat.

(b) Vocal vector values that indicates whether the audio file content includes a Sexy voice, a Smooth voice, a Powerful voice, a Great voice, or a Soulful voice.

(c) Sound quality vector values that indicate whether the audio file content includes a strong beat, is simple, has a good groove, is fast, is speech like or emphasizes a melody.

(d) Situational quality vector values that indicate whether the audio file content is good for a workout, a shopping mall, a dinner party, a dance party, slow dancing or studying.

(e) Ensemble vector values indicating whether the audio file includes a female solo, male solo, female duet, male duet, mined duet, female group, male group or instrumental.

(f) Genre vector values that indicate whether the audio file content belongs to a plurality of genres including Alternative, Blues, County, Electronics/Dance, Folk, Gospel, Jazz, Latin, New Age, Rhythm and Blues (R and B), Soul, Rap, Hip-Hop, Reggae, Rock and others.

(g) Instrument vectors that indicate whether the audio file content includes an acoustic guitar, electric guitar, bass, drum, harmonica, organ, piano, synthesizer, horn or saxophone.

[0070] If the input audio file matches a provisional rating, then in step S**708**, the process converts the provisional rating to a confirmed rating.

[0071] FIG. **8** is a flow diagram of computer executable process steps for generating a customized music list ("playlist") based upon user defined parameters.

[0072] In step S**801**, a music listener inputs a request for a playlist. UI **400**A (FIG. **4**) may be used by a music listener to input user preferences. UI **400**A is described in U.S. patent application Ser. No. 09/533,045, entitled "Method for Creating a Database for Comparing Music Attributes", filed on Mar. 22, 2000, and incorporated herein by reference. A user may request a playlist by specifying emotional quality, voice quality, instrument or genre vector, tempo, artist, album title and year of release, etc.

[0073] In step S**802**, playlist generator **400** scans audio files stored on user's disk **15**. One methodology of analyzing audio files is provided above in FIG. **7**. Audio files may also be acquired form a remote computer connected to the Internet and analyzed as shown above.

[0074] In step S**803**, Signature Comparator **400**C searches for music based upon analyzed audio file data.

[0075] In step S**804**, playlist generator **400** compiles a playlist based upon user preferences and the compared data.

Determining Audio Signatures

[0076] Audio signatures identify audio files and are based upon the signal characteristics of a particular audio file sample. If the audio signatures of a group of audio files is known then a new audio file may be compared with the known group of audio signatures. Audio signature is a representation of an audio file that assists in comparing audio samples. Audio signature may be developed for any audio file and whenever two audio samples overlap, the audio signatures of the samples will also overlap. It is this property that assists comparison of audio samples.

[0077] FIG. **9** is a flow diagram showing process steps for determining audio signatures according to one aspect of the present invention.

[0078] In step S**901**, an audio sample is represented as a signal. The signal is then used to determine a set of parameters or features that describe the signal. FIG. **6**A shows an example of representing an audio file over time.

[0079] In step S**902**, the determined features from step S**901** are used to compute an audio signature for the audio sample.

[0080] In step S**903**, the audio signature determined in step S**902** is compared to pre-computed audio signatures stored in database **401**.

[0081] The following describes different techniques under the present invention that may be used to implement the process steps of FIG. **9**.

Vector Quantization Methodology

[0082] Under this methodology, an audio file is transformed into a raw intensity signal and the transformed signal is used to compute a set of features or parameters. The features are computed from sequential (possibly overlapping) sets of raw intensity signal and transforms the intensity signal into a time series of feature values. This multivariate time series of feature vectors is then compressed into a 1-dimensional vector of elements. This compression is achieved using vector quantization of the multivariate features, so that each point in the feature space can be mapped onto an element in a finite codebook. Thereafter, the computed 1-dimensional

vector (1-d string) is matched against known values stored in a database using fast string matching algorithms to retrieve a match.

Pre-Processing

[0083] As discussed in step S**901**, pre-processing is used to convert an audio file whose signature is to be calculated, to a standard reference format. More specifically mono audio files of 16-bit samples at 22050 Hz sampling rate are used as a reference format. Standard commercially available software converters and decoders may be used for this purpose. More specifically Sox sound converting utility that can be downloaded from the Internet address www.spies.com/Sox/ and Xaudio mp3 decoder downloaded from a website located at www.xaudio.com may be used.

Feature Extraction

[0084] As discussed in step S**901**, certain features or parameters are extracted from an audio file signal. The features of this methodology are based on Short Time Fourier Transform (STFT) analysis. The STFT is a signal processing analysis technique that processes audio file signals in samples designated as bins or windows (FIG. **6**A) and then a Discrete Fourier Transform (DFT) is determined for the bins or windows. This technique is disclosed in A. V. Oppenheim and A. S. Wilsky, "Signals and Systems" (2nd Ed.) Prentice-Hall (1997), which is herein incorporated by reference in its entirety for all purposes. The signal consists of a time series of values. The time series is divided into smaller units or bins (that may be overlapping). The STFT and related features are computed on each of these bins. Thus the original signal time series is transformed into a vector time series of features.

[0085] It is noteworthy that because overall level (volume) effects vary across different encodings of the same audio file and can not be assumed to be uniform, STFT features are normalized to eliminate level (volume) effects by dividing out the dc component of the STFT at each sample point before further processing.

[0086] The following STFT-based features may be extracted in step S**901**:

[0087] Spectral Centroid is the balancing point of the spectrum (The spectrum is the representation of audio intensity over time. (FIG. **6**A)) magnitude and is defined as:

$$C = \frac{\sum i * M_i}{\sum M_i} \quad (1)$$

where $i = 1, 2, \ldots, N$

where N is the FFT window size and the Mi's are the frequency bins of the spectrum magnitude.

[0088] Spectral Rolloff like the spectral centroid is another measure of the shape of the spectrum. It is defined as:

$$R=r \text{ such that } \Sigma_1^r M_i = 0.8 * \Sigma_r^N M_i, \quad (2)$$

where N is the FFT window size, the Mi's are the frequency bins of the spectrum magnitude.

[0089] Spectral Flux is the 2-norm of the difference between the magnitude of the short time Fourier transform (STFT) spectrum evaluated at two successive analysis windows. Note that the signal is first normalized for energy, i.e. all segments on which the STFT is calculated are constrained to have equal energy (the same dc value of the STFT).

[0090] Peak ratio is the ratio of the magnitude of the highest peak in the magnitude spectrum to the average magnitude.

[0091] Subband energy vector is a vector of sub-band energies calculated by grouping STFT frequency bins to logarithmically spaced sub-bands. The STFT is normalized in energy.

[0092] Subband flux is the 2-norm of the difference between the subband energy vectors evaluated at two successively analyzed windows.

[0093] Subband Energy Ratios are formed by analyzing the energy component of the signal in each of a defined set of frequency bands. The ratios of the energy in these sub-bands provide a set of measures of the spectral composition of the signal. For example, if energies in 5 sub-bands are determined then there are 10 (5 choose 2) distinct ratios between the 5 sub-bands. These 10 numbers define a vector, which characterizes the spectral energy distribution. A related set of statistics that may be used is the logarithmic ratios of the sub-band energies. The logarithmic ratios make the numbers more stable and robust.

Signature Calculation (Step S**902**, FIG. 9)

Vector Quantization

[0094] Feature extraction in step S**901** provides a time series of feature vectors. Vector Quantization (VQ) is a technique that maps a large set of vectors to a smaller representative indexed set of vectors (code words) designated as the code-book. VQ is used to compress vector information. VQ reduces multidimensional vectors comprising of the computed features onto a single value. FIG. **10**A is a flow diagram showing computer executable process steps to perform VQ, according to one aspect of the present invention.

[0095] In step S**101**, identify a set of representative points. FIG. **10**B shows a distribution of a set of points. Each point may include a plurality of features defined above. Generally, the representative points are based upon music samples across many genres of music. The representative points specify location in a multidimensional feature space. After identifying a set of representative points, a c-means clustering algorithm is applied to the data set. The c-means clustering algorithm determines a set of cluster centers (A**1**, A**2**, A**3** and A**4** in FIG. **10**B). 20 and 40 cluster centers may be used for a 5-10 dimensional feature space. The invention is not limited to any particular numbers of cluster centers. The iterative c-means algorithm based upon relative Mahalonobis distance determines the cluster centers. Details of the foregoing techniques are provided in "Multivariate Observations" by G. A. F. Seber, published by John Wiley & Sons (1984), incorporated herein by reference in its entirety.

[0096] In step S**102**, the process defines a code-book and each of the representative points derived in step S**101** above is mapped to an element in the codebook, where each element corresponds to a unique ASCII character.

[0097] In step S**103**, the process defines a rule that can map any point in a feature space onto an element of the codebook. Cluster centers are assigned to the elements of the codebook. Points that are not cluster centers are mapped to a cluster center, and then assigned to an element of the codebook corresponding to that cluster center. To map an arbitrary point onto a cluster center Mahalonobis distance metric and a near-

est neighbor algorithm is used. Every point is mapped onto the closest cluster center, using a Mahalonobis distance or a similar metric.

[0098] VQ provides a string of characters. These characters may be compressed by using a logarithmic run length compression scheme. In this scheme a run of the same character is compressed to a shorter run with length equal to the logarithm of the original run length. For example, the string of characters aaaabccccccccc will be compressed to aabccc ($Log_2 8=3$ so the 8 c's are compressed to 3). The compressed string provides compact audio signatures that can be efficiently transmitted and compared. Also, logarithmic run length compression highlights the regions when the signal is changing.

Base Overshifting

[0099] The process described above depends upon STFT which in turn depends on the bin or window size of the signal. In particular, to compute STFT efficiently, the raw audio signal is partitioned into bins of a certain size (for example bins size 512 of sample points). However, where the bins should start and end affects the accuracy of the computed audio signature. For example, if a raw signal has 10,000 points and the signal is divided into bins of length **512**, the computation could start at the beginning of every 512 points or the computation could start at the second, or tenth, or hundredth point etc. Since not every audio sample starts at the "beginning" of some canonical version of the exact choice of bins or windows the signature extraction procedure should be robust to average out the arbitrariness in the computation.

[0100] The present solution provides a solution to the arbitrariness of bin selection by choosing features that are intrinsically robust to shifts in binning Also, multiple signatures may be computed for each audio sample based on different bin shifts. For example, one signature is computed based on starting the first bin at the beginning of the audio file. A second signature is then computed based on bins starting at the Nth data point. A third signature might be computed based on the first bin starting at the N+Kth data point and so on. Plural number of shifted audio signatures is computed for each audio sample. Each audio signature is a short list of strings (typically of length **8** or **16**) where each string corresponds to a slightly shifted bin analysis.

[0101] Audio signatures determined by the foregoing process allow signature comparison of new audio samples (query signature) with stored pre-computed audio signatures (base signature)(FIG. **7**, step S**704**). In order to compare a query signature with a base signature, the query signature is divided into overlapping sub-strings. The sub-strings are then compared to the base signature. The following example illustrates the matching process.

[0102] Assume that a query signature is denoted as AABBCADAABC . . . .

[0103] The above signature is broken into sub-strings of defined length (e.g. 11 characters). The sub-string of 11 characters is then compared to the stored base signature stored in central database **401**.

[0104] Hierarchical matching may also be used to obtain the best match. In this process a sub-string of X characters is used and thereafter, a subset of the sub-string comprised of X' characters where X' is less than X is used to match a query signature with the base algorithm. A specific confidence level or value may be assigned to a match for each sub-string. A match index value may be used to enhance the accuracy of matching where match index value is given by:

$$MI=\Sigma W_i * N_i, \quad (3)$$

where W_i is a weight (real number) for matches of length i and N_i is the number of substring matches of length i. Thus, Match Index (MI) is a weighted sum of the number of substring matches of different length substrings. Matches for longer substrings are less likely by chance and should thus be assigned a higher weight than matches of shorter length. Thus, the magnitude of the weight W_i is a measure of the confidence assigned to matches of a given substring length.

[0105] The foregoing process can be iterated to improve performance. For example, after a particular code-book is defined to match a small set of characters out of a large set, the process may define a second code-book for the subset of songs matched with the first codebook to expedite matching. Alternatively, many independent codebooks may be defined and used in conjunction. The best matching songs will match across the most codebooks. The advantage of this approach is that errors in matching in one codebook will be compensated for by matches in another codebook.

[0106] The following specific values may be used for the foregoing process:

sampling_rate=22050 Hz

sample size (bits)=16

fft_size=512

window_size=512

code_book_size=14

overshifting=8

query substring length=14

query overlap=13

[0107] It is noteworthy that the foregoing values are merely to illustrate the foregoing process and is not to limit the invention.

Segmentation Technique

[0108] Another method for determining audio signatures according to the present invention involves the following: 1) an audio sample is segmented into disjoint, contiguous regions in a robust and deterministic manner, 2) a robust description of the audio characteristics of each segment is determined. This description may be any parameterization of the audio file, or may consist of nonparametric samples, and 3) an audio signature is developed which reflects the segmentation structure of the original audio sample. One method to determine audio for segmenting audio files is described in an article "Multifeature Audio Segmentation for Browsing and Annotation", by George Tzanetakis, one of the present inventors, and Perry Cook ("Tzanetakis and Cook"), published Oct. 17-20, 1999, in "IEEE Workshop on Applications of Signal Processing to Audio and Acoustics", incorporated herein by reference in its entirety:

[0109] FIG. **10**C is a flow diagram of process steps to implement the Segmentation techniques.

[0110] In step S**1001**, an input audio file's is plotted over time. FIG. **6**A shows an audio sample where audio intensity (I_A) is plotted against time.

[0111] In step S**1002**, the process determines audio sample feature vectors or parameters. A set of parameters for I_A is computed for a set of short contiguous bins (or short overlapping bins). Hence, I_A is transformed into a set of parametric descriptions (over time), designated for the purpose of illustration as Pa_1, Pa_2, . . . , PaN, for N parameterizations. Pa1, Pa2, . . . , PaN, is a time series of feature vectors that describe

the audio file intensity I_A. Examples of such parameters are described above under vector quantization methodology and also include the following:

(a) Zero Crossings: Number of time domain zero crossings, i.e., the number of times the signal intensity changes from positive to negative or negative to positive in the sample file.

(b) Root Mean Square (RMS) energy (intensity) within a given bin.

(c) Nth percentile frequency of the spectrum (e.g., N=25, 50, 75, etc.)

[0112] The Nth percentile frequency of the spectrum of a specific bin is the largest frequency fk such that Sum(A_i)/Sum (A_j)<=N*100 where i=1, 2, . . . , fk and j=1, 2, . . . , fM, where fM is the highest sample frequency.

(h) Spectral energy correlation of contiguous bins (the correlation between bins i and i+N (where N is a positive integer representing the correlation lag computed for the spectral energy across frequencies).

[0113] The foregoing list of parameters is not exhaustive. Other mathematical variables may also be used to determine audio file vectors.

[0114] In step **S1003**, the process determines a univariate signal based upon the foregoing audio file parameters. Pa1, Pa2, . . . , PaN of I_A, are combined into a single, univariate signal d_t. One method to compute d_t is to compute the distance between successive time frames of the feature vector of parameterizations. Various distance metrics may be used as discussed in Tzanetakis and Cook. Mahalonobis distance is one such metric and is used between successive values of feature vectors Pa1, Pa2, . . . , PaN. Other distance metrics, such as Minkowski metrics, may also be used.

[0115] An example of determining d_t is provided below. Assume that t_1 and t_2 are adjacent bins of I_A which have parameterization feature vectors V_1 and V_2 respectively.

[0116] Hence,

$$V_1=[Pa1_1 Pa2_1 \ldots PaN_1], \text{ and} \quad (4)$$

$$V_2=[Pa1_2 Pa2_2 \ldots PaN_2]. \quad (5)$$

$$d_t=\|V_2-V_1\|, \quad (6)$$

where $\| \|$ indicates computation of the Mahalonobis distance between V_1 and V_2.

[0117] Therafter Δt_i is determined by $\|V_t-V_{t-1}\|$ for all adjacent bins.

[0118] After determining the univariate distance signal Δt, in step **S1004**, the process segments the audio sample. Regions of high change in the signal define segmentation. A derivative,

$$\frac{d\Delta t}{dt} \quad (7)$$

is computed and compared to a predefined threshold value. Points at which the derivative value is greater than the threshold value indicate sudden transitions in the audio signal and define segmentation points.

[0119] In step **S1005**, the audio signature for a sample audio file is determined, as illustrated by the following example.

[0120] For a given audio sample S, segment points are determined at t_1, t_2, . . . , t_N using the procedure described above. A set of N−1 segments may then be constructed, having lengths $L_1, L_2, \ldots, L_{N-1}$ where $L_k=t_{k+1}-t_k$ and t_k is the time point (in the sample) of the kth segment. For each audio segment a set of robust statistics are computed. Numerous statistical values may be used to define and describe specific segments for example, spectral centroid, spectral rolloff; intensity variance, skewness, kurtosis, and sub-band energy ratios etc.

[0121] Hence, having determined a set of audio segments $L_1, L_2, \ldots, L_M$ and a set of k robust statistics for each segment $R_{11}, R_{12}, \ldots, R_{1k}$, an audio signature, ("AS") may be determined as follows:

$$AS=[L_1 R_{11} R_{12} \ldots R_{1k} L_2 R_{21} R_{22} \ldots R_{2k} \ldots L_M R_{M1} R_{M2} \ldots R_{Mk}] \quad (8)$$

[0122] AS is a vector or string which concatenates the segment length and robust summary for each of the segments.

[0123] It is noteworthy that AS is definable for any audio sample of sufficient length to contain at least 1 segment, and if two different audio samples contain overlapping audio segments, then the audio signatures contain overlapping elements.

[0124] It is noteworthy that instead of segmenting an audio file into variable length segments as described above, the audio file is first segmented into fixed-length segments, and then robust statistics are calculated within each segment.

[0125] It is also noteworthy that any or all of these techniques described above may be combined in a multistage procedure.

[0126] One advantage of the foregoing aspects of the present invention is that unique audio signatures may be assigned to audio files. Also various attributes may be tagged to audio files. The present invention can generate a customized playlist for a user based upon audio file content and the attached attributes, hence making the music listening experiences easy and customized.

[0127] Although the present invention has been described with reference to specific embodiments, these embodiments are illustrative only and not limiting. Many other applications and embodiments of the present invention will be apparent in light of this disclosure and the following claims.

1. A method of generating a fingerprint from an unknown audio signal, the method comprising:

dividing the unknown audio signal into bins, each bin including a plurality of points representing a feature space;

mapping each of the plurality of points to one of a plurality of predetermined cluster centers based on the distance between each point and the plurality of cluster centers, each cluster center being associated with an element of a codebook;

generating a string of elements based on the mapping; and

compressing the string of elements.

2. The method according to claim **1**, further comprising:

performing a Fourier transform on the plurality of bins to generate a frequency representation of each bin; and

computing at least one feature from the frequency representation of each bin.

3. The method according to claim **2**, wherein the at least one feature includes at least one of:

(a) a spectral centroid;

(b) a spectral rolloff;

(c) a spectral flux;

(d) a peak ratio;

(e) a subband energy vector;
(f) a subband flux; and
(g) a subband energy ratio.

4. The method according to claim **1**, wherein each point representing the feature space specifies a location in a multidimensional feature space.

5. The method according to claim **1**, wherein the plurality of cluster centers are determined by using a c-means clustering algorithm.

6. The method according to claim **1**, wherein the distance between each point and the plurality of cluster centers is based on a Mahalonobis distance.

7. The method according to claim **1**, wherein each element of the codebook corresponds to a unique ASCII character.

8. The method according to claim **1**, further comprising:
parsing the compressed set of elements into a plurality of sub-strings of a predefined length of characters.
matching each substring to a database of precomputed signatures to obtain at least one base signature;
assigning a match index value (MI) to each match obtained by the matching given by:

$$MI=\Sigma W_i*N_i,$$

where W_i is a weight for matches of length i and N_i is a number of substring matches of length i; and
selecting a best match based on the match index value (MI).

9. A non-transitory computer-readable medium storing a computer instructions, which when executed by one or more processors, causes the one or more processors to execute the steps of:
dividing the unknown audio signal into bins, each bin including a plurality of points representing a feature space;
mapping each of the plurality of points to one of a plurality of predetermined cluster centers based on the distance between each point and the plurality of cluster centers, each cluster center being associated with an element of a codebook;
generating a string of elements based on the mapping; and
compressing the string of elements.

10. The non-transitory computer-readable medium according to claim **9**, further comprising the steps of:
performing a Fourier transform on the plurality of bins to generate a frequency representation of each bin; and
computing at least one feature from the frequency representation of each bin.

11. The non-transitory computer-readable medium according to claim **10**, wherein the at least one feature includes at least one of:
(a) a spectral centroid;
(b) a spectral rolloff;
(c) a spectral flux;
(d) a peak ratio;
(e) a subband energy vector;
(f) a subband flux; and
(g) a subband energy ratio.

12. The non-transitory computer-readable medium according to claim **9**, wherein each point representing the feature space specifies a location in a multidimensional feature space.

13. The non-transitory computer-readable medium according to according to claim **9**, wherein the plurality of cluster centers are determined by using a c-means clustering algorithm.

14. The non-transitory computer-readable medium according to according to claim **9**, wherein the distance between each point and the plurality of cluster centers is based on a Mahalonobis distance.

15. The non-transitory computer-readable medium according to according to claim **9**, wherein each element of the codebook corresponds to a unique ASCII character.

16. The non-transitory computer-readable medium according to according to claim **9**, further comprising the steps of:
parsing the compressed set of elements into a plurality of sub-strings of a predefined length of characters.
matching each substring to a database of precomputed signatures to obtain at least one base signature;
assigning a match index value (MI) to each match obtained by the matching given by:

$$MI=\Sigma W_i*N_i,$$

where W_i is a weight for matches of length i and N_i is a number of substring matches of length i; and
selecting a best match based on the match index value (MI).

17. An apparatus for generating a fingerprint from an unknown audio signal, comprising:
at least one processor operable to perform:
dividing the unknown audio signal into bins, each bin including a plurality of points representing a feature space;
mapping each of the plurality of points to one of a plurality of predetermined cluster centers based on the distance between each point and the plurality of cluster centers, each cluster center being associated with an element of a codebook stored in a memory;
generating a string of elements based on the mapping; and
compressing the string of elements.

18. The apparatus according to claim **17**, the processor further operable to perform a Fourier transform on the plurality of bins to generate a frequency representation of each bin and compute at least one feature from the frequency representation of each bin.

19. The apparatus according to claim **18**, wherein the at least one feature includes at least one of:
(a) a spectral centroid;
(b) a spectral rolloff;
(c) a spectral flux;
(d) a peak ratio;
(e) a subband energy vector;
(f) a subband flux; and
(g) a subband energy ratio.

20. The apparatus according to claim **17**, wherein each point representing the feature space specifies a location in a multidimensional feature space.

21. The apparatus according to claim **17**, wherein the plurality of cluster centers are determined by using a c-means clustering algorithm.

22. The apparatus according to claim **17**, wherein the distance between each point and the plurality of cluster centers is based on a Mahalonobis distance.

23. The method according to claim **17**, wherein each element of the codebook corresponds to a unique ASCII character.

24. The apparatus according to claim **17**, wherein the at one processor is further operable to perform the following steps:
parsing the compressed set of elements into a plurality of sub-strings of a predefined length of characters.

matching each substring to a database of precomputed signatures to obtain at least one base signature;

assigning a match index value (MI) to each match obtained by the matching given by:

$$MI=\Sigma W_i*N_i,$$

where W_i is a weight for matches of length i and N_i is a number of substring matches of length i; and

selecting a best match based on the match index value (MI).

* * * * *