Shazam Application:

1. Introduction:-

The Shazam algorithm can be used in many applications besides just music recognition over a mobile phone. Due to the ability to dig deep into noise we can identify music hidden behind a loud voiceover, such as in a radio advert. On the other hand, the algorithm is also very fast and can be used for copyright monitoring at a search speed of over 1000 times realtime, thus enabling a modest server to monitor significantly many media streams. The algorithm is also suitable for content-based cueing and indexing for library and archival uses.

2. Basic principle of operation:-

Each audio file is “fingerprinted,” a process in which reproducible hash tokens are extracted. Both “database” and “sample” audio files are subjected to the same analysis.

The fingerprints from the unknown sample are matched against a large set of fingerprints derived from the music database. The candidate matches are subsequently evaluated for correctness of match. Some guiding principles for the attributes to use as fingerprints are that they should be temporally localized, translation invariant, robust , and sufficiently entropic. The temporal locality guideline suggests that each fingerprint hash is calculated using audio samples near a corresponding point in time, so that distant events do not affect the hash.

The translationinvariant aspect means that fingerprint hashes derived from corresponding matching content are reproducible independent of position within an audio file, as long as the temporal locality containing the data from which the hash is computed is contained within the file. This makes sense, as an unknown sample could come from any portion of the original audio track. Robustness means that hashes generated from the original clean database track should be reproducible from a degraded copy of the audio. Furthermore, the fingerprint tokens should have sufficiently high entropy in order to minimize the probability of false token matches at non-corresponding locations between the unknown sample and tracks within the database. Insufficient entropy leads to excessive and spurious matches at non-corresponding locations, requiring more processing power to cull the results, and too much entropy usually leads to fragility and non-reproducibility of fingerprint tokens in the presence of noise and distortion.

- Robust Constellations:-

  In order to address the problem of robust identification in the presence of highly significant noise and distortion, we experimented with a variety of candidate features that could survive GSM encoding in the presence of noise.

  We settled on spectrogram peaks, due to their robustness in the presence of noise and approximate linear superposability [1].

  A time-frequency point is a candidate peak if it has a higher energy content than all its neighbors in a region centered around the point.

Candidate peaks are chosen according to a density criterion in order to assure that the time-frequency strip for the audio file has reasonably uniform coverage.

The peaks in each time-frequency locality are also chosen according amplitude, with the justification that the highest amplitude peaks are most likely to survive the distortions listed above.

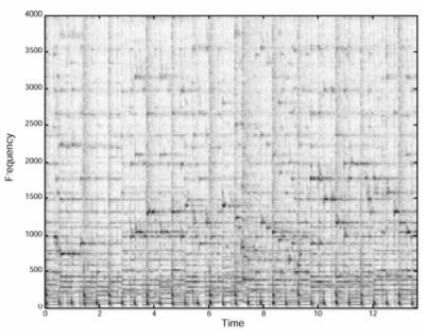


Fig. 1A - Spectrogram

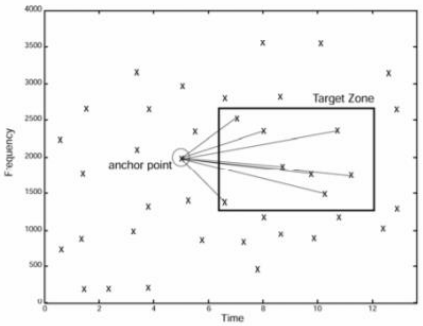


Fig. 1C - Combinatorial Hash Generation

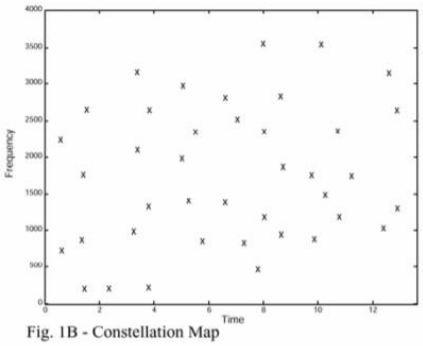


Fig. 1B - Constellation Map

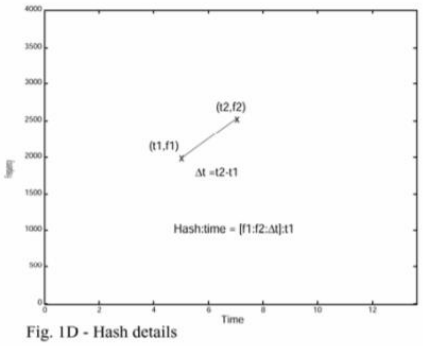


Fig. 1D - Hash details

Thus, a complicated spectrogram, as illustrated in Figure 1A may be reduced to a sparse set of coordinates, as illustrated in Figure 1B.

Notice that at this point the amplitude component has been eliminated. This reduction has the advantage of being fairly insensitive to EQ, as generally a peak in the spectrum is still a peak with the same coordinates in a filtered spectrum (assuming that the derivative of the filter transfer function is reasonably small—peaks in the vicinity

of a sharp transition in the transfer function are slightly frequency-shifted). We term the sparse coordinate lists “constellation maps” since the coordinate scatter plots often resemble a star field.

The pattern of dots should be the same for matching segments of audio.

If you put the constellation map of a database song on a strip chart, and the constellation map of a short matching audio sample of a few seconds length on a transparent piece of plastic, then slide the latter over the former, at some point a significant number of points will coincide when the proper time offset is located and the two constellation maps are aligned in register.

The number of matching points will be significant in the presence of spurious peaks injected due to noise, as peak positions are relatively independent; further, the number of matches can also be significant even if many of the correct points have been deleted.

Registration of constellation maps is thus a powerful way of matching in the presence of noise and/or deletion of features.

This procedure reduces the search problem to a kind of “astronavigation,” in which a small patch of time-frequency constellation points must be quickly located within a large universe of points in a strip-chart universe with dimensions of bandlimited frequency versus nearly a billion seconds in the database.

- Fast Combinatorial Hashing:-

  Finding the correct registration offset directly from constellation maps can be rather slow, due to raw constellation points having low entropy.

  For example, a 1024-bin frequency axis yields only at most 10 bits of frequency data per peak. We have developed a fast way of indexing constellation maps.

  Fingerprint hashes are formed from the constellation map, in which pairs of time-frequency points are combinatorially associated. Anchor points are chosen, each anchor point having a target zone associated with it.

  Each anchor point is sequentially paired with points within its target zone, each pair yielding two frequency components plus the time difference between the points (Figure 1C and 1D). These hashes are quite reproducible, even in the presence of noise and voice codec compression. Furthermore, each hash can be packed into a 32-bit unsigned integer.

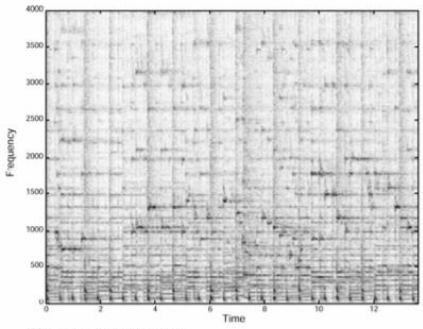


Fig. 1A - Spectrogram

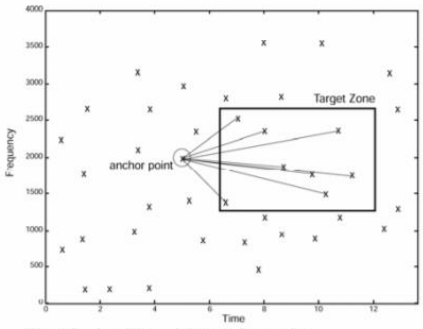


Fig. 1C - Combinatorial Hash Generation

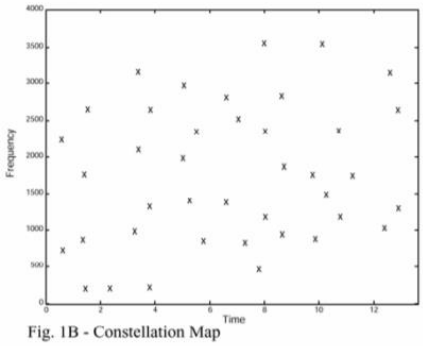


Fig. 1B - Constellation Map

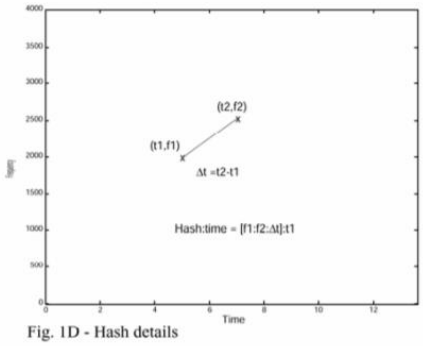


Fig. 1D - Hash details

To create a database index, the above operation is carried out on each track in a database to generate a corresponding list of hashes and their associated offset times. Track IDs may also be appended to the small data structs, yielding an aggregate 64-bit struct, 32 bits for the hash and 32 bits for the time offset and track ID.

To facilitate fast processing, the 64-bit structs are sorted according to hash token value.

The number of hashes per second of audio recording being processed is approximately equal to the density of constellation points per second times the fan-out factor into the target zone.

For example, if each constellation point is taken to be an anchor point, and if the target zone has a fanout of size F=10, then the number of hashes is approximately equal to F=10 times the number of constellation points extracted from the file.

By limiting the number of points chosen in each target zone, we seek to limit the combinatorial explosion of pairs.

The fan-out factor leads directly to a cost factor in terms of storage space.

By forming pairs instead of searching for matches against individual constellation points we gain a tremendous acceleration in the search process.

For example, if each frequency component is 10 bits, and the Δt component is also 10 bits, then matching a pair of points yields 30 bits of information, versus only 10 for a single point.

Then the specificity of the hash would be about a million times greater, due to the 20 extra bits, and thus the search speed for a single hash token is similarly accelerated.

On the other hand, due to the combinatorial generation of hashes, assuming symmetric density and fan-out for both database and sample hash generation, there are F times as many token combinations in the unknown sample to search for, and F times as many tokens in the database, thus the total speedup is a factor of about $1000000/F^2$, or about 10000, over token searches based on single constellation points.

Note that the combinatorial hashing squares the probability of point survival, i.e. if p is the probability of a spectrogram peak surviving the journey from the original source material to the captured sample recording, then the probability of a hash from a pair of points surviving is approximately p^2 .

This reduction in hash survivability is a tradeoff against the tremendous amount of speedup provided.

The reduced probability of individual hash survival is mitigated by the combinatorial generation of a greater number of hashes than original constellation points.

For example, if F=10, then the probability of at least one hash surviving for a given anchor point would be the joint probability of the anchor point and at least one target point in its target zone surviving.

If we simplistically assume IID probability p of survival for all points involved, then the probability of at least one hash surviving per anchor point is $p*[1-(1-p)^F]$.

For reasonably large values of F, e.g. F>10, and reasonable values of p, e.g. p>0.1, we have approximately

$$p \approx p*[1-(1-p)^F]$$

so we are actually not much worse off than before.

We see that by using combinatorial hashing, we have traded off approximately 10 times the storage space for approximately 10000 times improvement in speed, and a small loss in probability of signal detection.

Different fan-out and density factors may be chosen for different signal conditions.

For relatively clean audio, e.g. for radio monitoring applications, F may be chosen to be modestly small and the density can also be chosen to be low, versus for the somewhat more challenging mobile phone consumer application.

The difference in processing requirements can thus span many orders of magnitude.

- Searching and Scoring:-

  To perform a search, the above fingerprinting step is performed on a captured sample sound file to generate a set of hash:time offset records.

  Each hash from the sample is used to search in the database for matching hashes.

  For each matching hash found in the database, the corresponding offset times from the beginning of the sample and database files are associated into time pairs.

The time pairs are distributed into bins according to the track ID associated with the matching database hash.

After all sample hashes have been used to search in the database to form matching time pairs, the bins are scanned for matches.

Within each bin the set of time pairs represents a scatterplot of association between the sample and database sound files.

If the files match, matching features should occur at similar relative offsets from the beginning of the file, i.e. a sequence of hashes in one file should also occur in the matching file with the same relative time sequence.

The problem of deciding whether a match has been found reduces to detecting a significant cluster of points forming a diagonal line within the scatterplot.

Various techniques could be used to perform the detection, for example a Hough transform or other robust regression technique.

Such techniques are overly general, computationally expensive, and susceptible to outliers.

Due to the rigid constraints of the problem, the following technique solves the problem in approximately N*log(N) time, where N is the number of points appearing on the scatterplot.

For the purposes of this discussion, we may assume that the slope of the diagonal line is 1.0.

Then corresponding times of matching features between matching files have the relationship:-

$$t_k' = t_k + \text{offset},$$

where t_k' is the time coordinate of the feature in the matching (clean) database soundfile and t_k is the time coordinate of the corresponding feature in the sample soundfile to be identified. For each (t_k', t_k) coordinate in the scatterplot, we calculate:-

$$\delta t_k = t_k' - t_k$$

Then we calculate a histogram of these δt_k values and scan for a peak.

This may be done by sorting the set of δt_k values and quickly scanning for a cluster of values.

The scatterplots are usually very sparse, due to the specificity of the hashes owing to the combinatorial method of generation as discussed above.

Since the number of time pairs in each bin is small, the scanning process takes on the order of microseconds per bin, or less.

The score of the match is the number of matching points in the histogram peak. The presence of a statistically significant cluster indicates a match.

Figure 2A illustrates a scatterplot of database time versus sample time for a track that does not match the sample.

There are a few chance associations, but no linear correspondence appears.

Figure 3A shows a case where a significant number of matching time pairs appear on a diagonal line.

Figures 2B and 3B show the histograms of the δt_k values corresponding to Figures 2A and 3B.

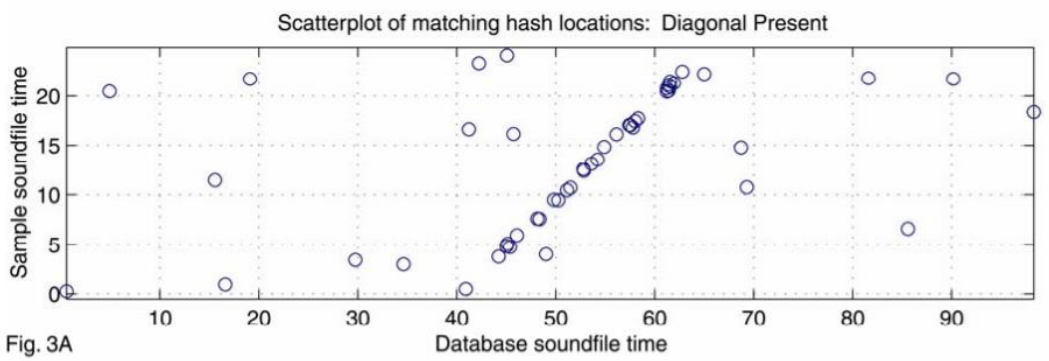


Fig. 3A

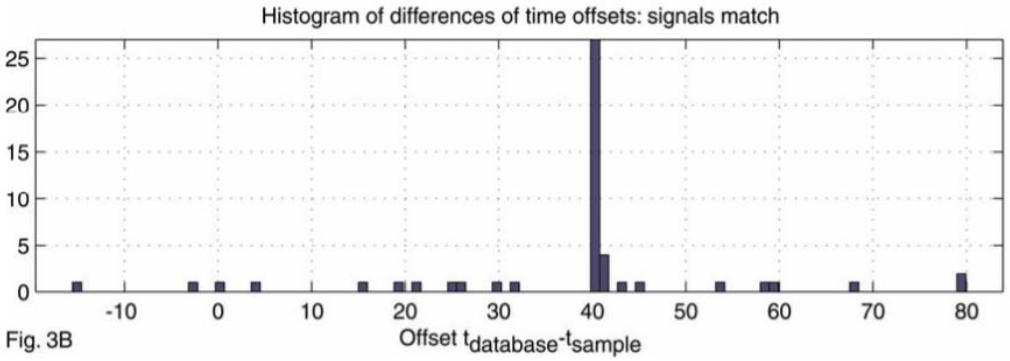


Fig. 3B

This bin scanning process is repeated for each track in the database until a significant match is found.

Note that the matching and scanning phases do not make any special assumption about the format of the hashes.

In fact, the hashes only need to have the properties of having sufficient entropy to avoid too many spurious matches to occur, as well as being reproducible.

In the scanning phase the main thing that matters is for the matching hashes to be temporally aligned.

- Significance:-

  As described above, the score is simply the number of matching and time-aligned hash tokens.

The distribution of scores of incorrectly-matching tracks is of interest in determining the rate of false positives as well as the rate of correct recognitions.

To summarize briefly, a histogram of the scores of incorrectly-matching tracks is collected.

The number of tracks in the database is taken into account and a probability density function of the score of the highestscoring incorrectly-matching track is generated.

Then an acceptable false positive rate is chosen (for example 0.1% false positive rate or 0.01%, depending on the application), then a threshold score is chosen that meets or exceeds the false-positive criterion.

3. Performance:-

- Noise resistance:-

The algorithm performs well with significant levels of noise and even non-linear distortion.

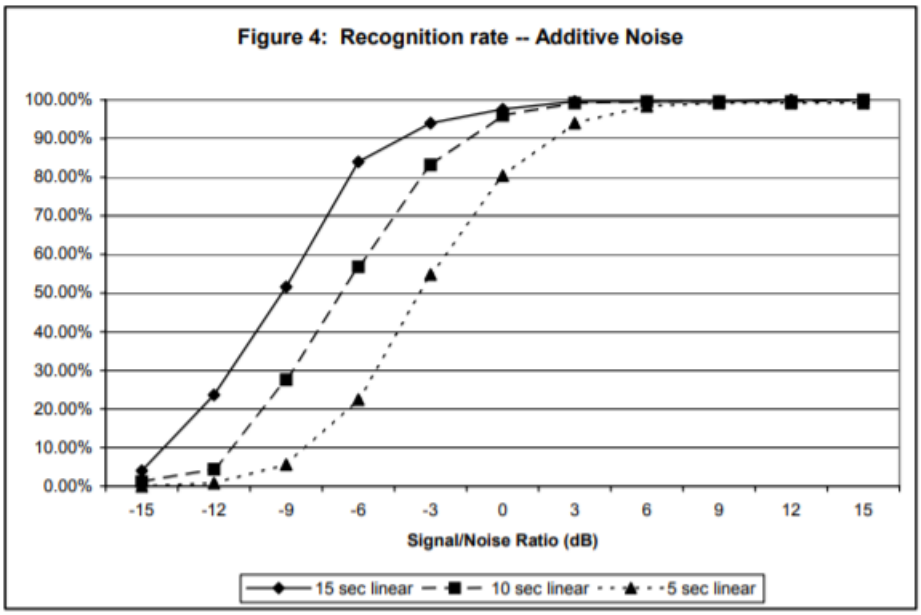
It can correctly identify music in the presence of voices, traffic noise, dropout, and even other music.

To give an idea of the power of this technique, from a heavily corrupted 15 second sample, a statistically significant match can be determined with only about 1-2% of the generated hash tokens actually surviving and contributing to the offset cluster.

A property of the scatterplot histogramming technique is that discontinuities are irrelevant, allowing immunity to dropouts and masking due to interference.

One somewhat surprising result is that even with a large database, we can correctly identify each of several tracks mixed together, including multiple versions of the same piece, a property we call “transparency”.

Figure 4 shows the result of performing 250 sample recognitions of varying length and noise levels against a test database of 10000 tracks consisting of popular music.



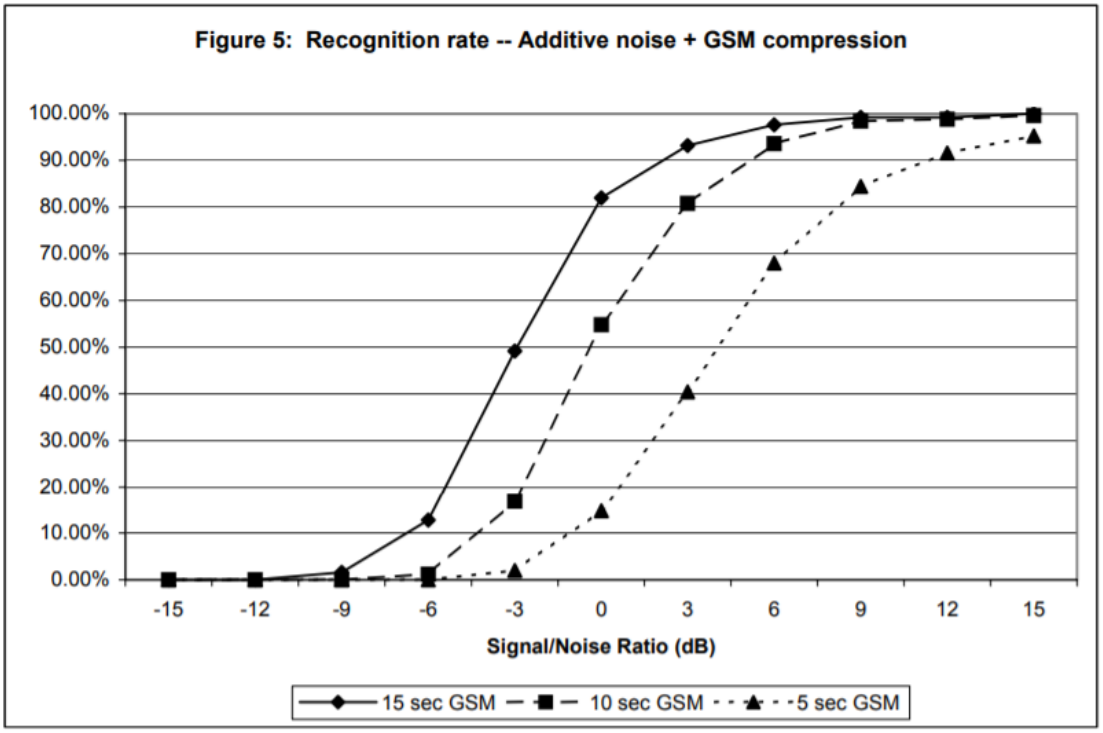
A noise sample was recorded in a noisy pub to simulate “real-life” conditions.

Audio excerpts of 15, 10, and 5 seconds in length were taken from the middle of each test track, each of which was taken from the test database.

For each test excerpt, the relative power of the noise was normalized to the desired signal-to-noise ratio, then linearly added to the sample.

We see that the recognition rate drops to 50% for 15, 10, and 5 second samples at approximately -9, -6, and -3 dB SNR, respectively Figure 5 shows the same analysis, except that the resulting music+noise mixture was further

subjected to GSM 6.10 compression, then reconverted to PCM audio.



In this case, the 50% recognition rate level for 15, 10, and 5 second samples occurs at approximately -3, 0, and +4 dB SNR. Audio sampling and processing was carried out using 8KHz, mono, 16-bit samples.

- Speed:-

  For a database of about 20 thousand tracks implemented on a PC, the search time is on the order of 5-500 milliseconds, depending on parameters settings and application.

  The service can find a matching track for a heavily corrupted audio sample within a few hundred milliseconds of core search time.

  With “radio quality” audio, we can find a match in less than 10 milliseconds, with a likely optimization goal reaching down to 1 millisecond per query.

- Specificity and False Positives:-

  The algorithm was designed specifically to target recognition of sound files that are already present in the database.

  It is not expected to generalize to live recordings.

  That said, we have anecdotally discovered several artists in concert who apparently either have extremely accurate and reproducible timing (with millisecond precision), or are more plausibly lip synching. The algorithm is conversely very sensitive to which particular version of a track has been sampled.

  Given a multitude of different performances of the same song by an artist, the algorithm can pick the correct one even if they are virtually indistinguishable by the human ear.

  We occasionally get reports of false positives. Often times we find that the algorithm was not actually wrong since it had picked up an example of “sampling,” or plagiarism.

  As mentioned above, there is a tradeoff between true hits and false positives, and thus the maximum allowable percentage of false positives is a design parameter that is chosen to suit the application.

4. References:-

1. Avery Li-Chun Wang and Julius O. Smith, III., WIPO publication WO 02/11123A2, 7 February 2002, (Priority 31 July 2000).

2. http://www.musiwave.com .

3. Jaap Haitsma, Antonius Kalker, Constant Baggen, and Job Oostveen., WIPO publication WO 02/065782A1, 22 August 2002, (Priority 12 February, 2001).

4. Jaap Haitsma, Antonius Kalker, "A Highly Robust Audio Fingerprinting System", International Symposium on Music Information Retrieval (ISMIR) 2002, pp. 107-115.

5. Neuros Audio web site: http://www.neurosaudio.com/ .

6. Sean Ward and Isaac Richards, WIPO publication WO 02/073520A1, 19 September 2002, (Priority 13 March 2001) .

7. Audible Magic web site: http://www.audiblemagic.com/ .

8. Erling Wold, Thom Blum, Douglas Keislar, James Wheaton, "Content-Based Classification, Search, and Retrieval of Audio", in IEEE Multimedia, Vol. 3, No. 3: FALL 1996, pp. 27-36 .

9. Clango web site: http://www.clango.com/ .

10. Cheng Yang, "MACS: Music Audio Characteristic Sequence Indexing For Similarity Retrieval", in IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2001.