Audio Content Identification by using Perceptual Hashing

R. Lancini, F. Mapelli and R. Pezzano
CEFRIEL – Politecnico di Milano
Via R.Fucini, 2 – 20133, Milano – ITALY
E-mail: {rosa,mapelli}@cefriel.it

Abstract

This paper presents a hashing method for automatic song recognition. This technique works analyzing the signal taking into account its nature. The goal of automatic recognition is obtained extracting different features, which are robust to signal processing and distinctive of the signal. They describe in a compact way the signal and can be efficiently stored in a database. The features are analyzed for short frames using a quantization approach, and a parameter for identification is proposed. Moreover we define also a confidence computed on the identification parameter. Combining the two parameters a better identification is obtained. The algorithm is tested in different situations: compression, cropping, noise addition, subsampling, stereo to mono conversion, etc. The results show that the identification can be performed also using a short excerpt of the song.

1. Introduction

In the last few years non-invasive techniques for automatic recognition have received an increasing interest. Among them one of the most important techniques is hashing [1], also known as (passive) fingerprinting. The non-invasive techniques can be used in a lot of applications such as information retrieval, automatic content identification, authentication and so on. In this paper we present a hashing technique as a method for automatic recognition of songs. Given a song or an excerpt we want to be able to identify which song it is with a good confidence. This has to be done without modifying the original signal but only analyzing it. Many examples can be found in literature, see [1,2,4,5,6,7,8].

Our algorithm works in the frequency domain extracting three different features [2,4], which are proved to be robust to different operations. Robust means that they should not – so much – be affected by signal processing operation – like mp3 compression, noise addition and so on. Moreover the features have to be distinctive of the signal in order to distinguish between different songs. Features are analyzed using adaptive quantization on short window of about two seconds. In this way we realize a compact representation of features that can be stored in a database. In the identification phase we define a parameter to measure the similitude between two hashes. Moreover we define a confidence parameter starting from the similitude in order to check the reliability of the identification.

The algorithm is tested for different signal processing operations and the obtained results show that this goal can be achieved with very high confidence even if we do not have the whole song but only a little excerpt of few seconds.

2. Hash extraction phase

Figure 1 shows the main phases of the algorithm that are described in the following section taking into account the main parts of a hashing algorithm according to [1]. In the following paragraphs we explain the different phases.

![Figure 1 - Hash extraction algorithm.](image)

2.1. Preprocessing

This first phase is important in order to improve the efficiency of the algorithm and to obtain a better modeling of the audio signal and features robust to
audio processing. First of all the audio is converted to a mono signal, if necessary. One of the most important problems to take into account is the loss of some frequencies. The first reason could be the resample – for example for lower bit-rate coding – that is solved resampling the signal to 44.1kHz. The second reason is the transmission over band limited network – e.g. phone network – so we limit our analysis to the most significant frequency range: 300-3400 Hz.

2.2. Framing and windowing

In this phase the original signal is split in shorter parts called frames following the approach in [2]. This is important for some different reasons. Firstly the whole signal could not be available. Moreover we know that the audio signal is not stationary, but the shorter is the frame the more stationary can be considered the signal. Lastly working on short frames allows faster operation reducing computational burden. The chosen frame length is 32768 – this is a power of 2 in order to allow the use of fast algorithm for the transform. Many signal processing operations – like shifting, time stretch and so on – can cause loss of alignment between frames in original and modified songs. This effect can be reduced introducing an overlap between frames: the frames are overlapped for 63/64 samples. To reduce discontinuities each frame is windowed using a Hamming window.

It is almost impossible to identify correctly each single frame, for this reason we have to choose the minimum identifiable shot. This shot is about 2 seconds long, includes 64 frames and is called hash window.

2.3. Transform

To choose the transform we have to decide which parameters model the fingerprint. We are interested in three parameters: the energy of each window, the SCF (Spectral Crest Factor) and SFM (Spectral Flatness Measure). They are computable in the frequency domain: we use the FFT. For this reason in the previous paragraph we have chosen the size of the frame as a power of 2.

2.4. Feature extraction

Many features have been proposed for robust modeling of an audio signal. In our work we focus the attention on three different parameters [2,3,5] with different characteristics in order to find a trade off between a robust fingerprint and a low complexity algorithm.

![Energy with quantization levels](image1)

![Histogram of energy values](image2)

**Figure 2 - Feature plot with linear quantization levels (above) and histogram of values distribution (below).**

The three features are:

- Mean energy:
  \[
  E = \frac{1}{N} \sum_{1}^{N} |S(f)|^2 ;
  \]

- Spectral Flatness Measure:
  \[
  SFM = 10 \log\left(\frac{Mg[S(f)]^2}{Ma[S(f)]^2}\right) ;
  \]

- Spectral Crest Factor:
  \[
  SCF = 10 \log\left(\frac{Max[S(f)]^2}{Ma[S(f)]^2}\right) ;
  \]

where \(Ma\) is the arithmetic mean, \(Mg\) the geometrical mean, \(s(t)\rightarrow E\rightarrow S(F)\), \(s(t)\) the signal in the time domain and \(S(f)\) the signal in the Fourier domain.

2.5. Fingerprint modeling

Different approaches are proposed to analyze the extracted features in literature like feature difference [2]. We use a different approach: each parameter is quantized using 4 bit per level. In this way we reduce the “noise” due to signal processing and the data to be stored. The problem is the choice of the quantization – e.g. Figure 2 shows a feature and its value distribution using a linear quantization. The distribution varies a lot with modification on the signal – for example if a high peak is inserted the quantization tends to quantize the whole signal with low value. This reduces the identification capabilities of our system. This reason leads us to develop a different adaptive quantization. We analyze the distribution of the values and we consider non-uniform interval. We order in increasing way the values, than we take the middle and split them...
in two groups. Then, in each group, we repeat the procedure, and so on until we obtain 16 intervals. In this way, we highly reduce the dependence on signal modification. Figure 3 shows the quantization levels obtained for the same feature of Figure 2. Obviously, in this case we have the same number of samples for each quantization level, so the histogram is constant.

![Figure 3 - Feature plot and adaptive quantization levels.](image)

In this way, we quantize each feature in each hash window – i.e. 64 frames. We use 4 bits for the quantized values, this means that we have a hash window – 2 seconds of song – represented by 768 bits – 64 (values per hash window) * 4 (bits per values) * 3 (features).

3. Identification parameter

As the identification parameter we use the following similarity parameter computed between the two hashes we are comparing:

\[ d = \frac{1}{N_O} \sqrt{\frac{1}{N_F} \sum (H_2 - H_1)^2}; \]

where \( H_1 \) and \( H_2 \) are the hash values to compare, \( N_F = 64 \) the number of values in each hash, and \( N_O = 16 \) the number of quantization levels. Obviously, the lower is \( d \), the higher is the similarity between the two hashes. The definition entails \( d \in [0,1] \).

It is important to determine the threshold \( T \) for the identification. If \( d > T \) we decide that the two compared hashes belong to different songs, otherwise if \( d < T \) we have the same song. Experimentally, we found that a good value for the threshold is \( T = 0.35 \).

4. Songs identification

The identification process is presented in this section. The first frame – i.e. 32768 samples, about 0.74 seconds – is extracted and analyzed. Then, the window analysis is moved 1/64 of frame – i.e. 512 samples – ahead on the next frame, and so on. Once analyzed the first 64 frames, we have a complete hash window, so we quantize the coefficients and compute the first distance in order to identify which is the song.

For the database retrieval, the process is similar to the one in [2]. The retrieval is performed not on the whole fingerprint but using only few bits – in particular, the number of bits obtained from one frame. In this way, it is simplified the comparison and it allows the retrieval even if time shifts have occurred.

At the first step, the retrieved song is the one with the lowest \( d \). Then, the algorithm goes on analyzing next frames and windows. And the identification is performed again. We compute also the confidence of the results considering two parameters: \( d \) and the number of consecutive windows where the song has \( d < T \). For example, take the following values of \( d \) for two songs S1 and S2:

<table>
<thead>
<tr>
<th>Song</th>
<th>0.1</th>
<th>0.08</th>
<th>0.15</th>
<th>0.07</th>
<th>0.05</th>
<th>0.04</th>
<th>0.2</th>
<th>0.11</th>
<th>0.13</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0.5</td>
<td>0.48</td>
<td>0.6</td>
<td>0.35</td>
<td>0.4</td>
<td>0.1</td>
<td>0.5</td>
<td>0.6</td>
<td>0.47</td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It can be noticed that S1 has hash \( d \) below threshold except the 6th value, while S2 is always above threshold except the 6th value. Checking each single hash without considering previous results, we should say that we have: S1 for 5 times, than S2, than S1 for other 4 times. But checking the confidence, we can say that the 6th result is S2 with very low confidence.

The confidence is computed as:

\[ C = \frac{1}{2} \left( \frac{1}{N_T} \sum_{N_F} (1 - d_T) - \frac{1}{N_{NT}} \sum_{N_{NT}} (1 - d_{NT}) \right) \]

where \( N_T \) is the number of hash values with \( d \leq T \) and \( d_T \) is the value of \( d \) for these hashes; \( N_{NT} \): the number of hash values with \( d > T \) and \( d_{NT} \) is the value of \( d \) for these hashes. Obviously \( C \in [0,1] \). Typical plot of \( C \) has peaks in presence of consecutive values with \( d < T \), and rapidly decreases when \( d > T \) – an example is in Figure 4. The threshold – named \( T_C = 0.75 \) – represents a good experimental value to avoid false positive.

5. Test results

Different tests are performed in order to check the efficiency of our algorithm. In our test, we take more than 100 different songs both of same and different genre. The reported results are intended as a mean for
each window analyzed — e.g. \( d = 0.1 \) means that the mean of all \( d \) computed for each window is 0.1.

Figure 4 – Confidence \( C \) (upper graph) computed on parameter \( d \) (lower graph).

First of all we test the original songs in order to verify that different songs provide different hashes. In this test we have obviously \( d = 0 \) comparing the song with itself, while we have about \( d = 0.73 \) comparing different songs. The confidence is very high, always above 0.9 for any song.

The second part of tests is mainly focused on compression. In this phase we test the identification after compression down to 32kbps. The compression can be combined with subsampling and/or conversion from stereo to mono. As an example in Table 1 is reported an excerpt of the results obtained with compression to 64 kbps after mono conversion. The reported number are the parameter \( d \) obtained comparing the compressed version \( AS^* \) with the original \( S^* \). The obtained confidence is high also in this case, always above 0.8.

The last part of tests is focused on different attacks like cropping, noise addition and subsampling. These attacks do not cause problems to our algorithm. In any case the song can be identified with confidence higher than 0.75.

It is difficult a comparison with other hashing techniques due to the fact that each algorithm has its own parameters. The main features of our algorithm are the short shot of song needed for identification and the computation of a confidence parameter.

### 6. Conclusions

In this paper we presented an audio fingerprinting algorithm. This is based on a quantization approach of different features in the frequency domain. This technique provided good results in term of identification of attacked songs. The confidence as a function of the identification parameter is also studied and it provides an additional control on the identification. The work is mainly focused on improves the results part increasing the number of tested song and tests – following specification in [9]. We are also studying the use of error correcting codes to improve the performance of the database search.

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>S8</th>
<th>S9</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS1</td>
<td>0.19</td>
<td>0.50</td>
<td>0.61</td>
<td>0.56</td>
<td>0.70</td>
<td>0.62</td>
<td>0.60</td>
<td>0.67</td>
</tr>
<tr>
<td>AS2</td>
<td>0.54</td>
<td>0.14</td>
<td>0.39</td>
<td>0.83</td>
<td>0.70</td>
<td>0.51</td>
<td>0.52</td>
<td>0.53</td>
</tr>
<tr>
<td>AS3</td>
<td>0.61</td>
<td>0.48</td>
<td>0.16</td>
<td>0.75</td>
<td>0.56</td>
<td>0.63</td>
<td>0.43</td>
<td>0.39</td>
</tr>
<tr>
<td>AS4</td>
<td>0.45</td>
<td>0.72</td>
<td>0.71</td>
<td>0.11</td>
<td>0.62</td>
<td>0.33</td>
<td>0.73</td>
<td>0.89</td>
</tr>
<tr>
<td>AS5</td>
<td>0.64</td>
<td>0.71</td>
<td>0.50</td>
<td>0.55</td>
<td>0.24</td>
<td>0.46</td>
<td>0.56</td>
<td>0.63</td>
</tr>
<tr>
<td>AS6</td>
<td>0.56</td>
<td>0.58</td>
<td>0.52</td>
<td>0.43</td>
<td>0.45</td>
<td>0.19</td>
<td>0.61</td>
<td>0.74</td>
</tr>
<tr>
<td>AS7</td>
<td>0.67</td>
<td>0.67</td>
<td>0.41</td>
<td>0.71</td>
<td>0.51</td>
<td>0.66</td>
<td>0.25</td>
<td>0.46</td>
</tr>
<tr>
<td>AS8</td>
<td>0.73</td>
<td>0.57</td>
<td>0.41</td>
<td>0.90</td>
<td>0.66</td>
<td>0.77</td>
<td>0.45</td>
<td>0.19</td>
</tr>
<tr>
<td>AS9</td>
<td>0.44</td>
<td>0.38</td>
<td>0.51</td>
<td>0.57</td>
<td>0.56</td>
<td>0.46</td>
<td>0.61</td>
<td>0.67</td>
</tr>
</tbody>
</table>

Table 1 – Parameter \( d \) between original song \( (S^*) \) and attacked version \( (AS^*) \).

### 7. References