Fast Philips Fingerprinting Algorithm with minimal artifacts

Dong-In Lee, Markus Cremer, Randall Cook
Gracenote, Inc.

ABSTRACT

In this paper we introduce a modified Philips fingerprinting algorithm which generates fingerprints as twice as faster than the original Philips algorithm. Our test results show promising music recognition rates from this new algorithm.

1. INTRODUCTION

Gracenote has been implementing Philips fingerprinting algorithm in C implementation only. After Google launched Android platform, there has been a need for JAVA implementation for Android phone support. As we already know, JAVA has some performance issues compared to C. Especially when it comes to numerical computation, JAVA is outperformed by C in almost every part. Through our initial investigation, we realized that Philips algorithm in JAVA took too much time to fingerprint. Thus, we added a simple modification on the original Philips algorithm to speed up the computation time. To do so, we had to sacrifice the fingerprint quality slightly.

2. ALGORITHM CONCEPTS

In this section the original Philips algorithm is described first, and modification on the algorithm is presented followed by generation of the weak bit.

2.1. Original Algorithm

The original Philips algorithm extracts 32-bit sub-fingerprints for every interval of 11.6 milliseconds. A fingerprint block consists of 256 subsequent sub-fingerprints, corresponding to a granularity of 3 seconds. The audio signal is segmented into overlapping frames. The overlapping frames have a length of 0.37 seconds and are weighted by a hanning window with an overlap factor of 31/32. Due to the large overlap subsequent sub-fingerprints have a large similarity and are slowly varying in time.

In order to extract a 32-bit sub-fingerprint value for every frame, 33 non-overlapping frequency bands are selected. If we denote the energy of band m of frame n by $E(n,m)$ and the m-th bit of the sub-fingerprint of frame n by $F(n,m)$, the bits of the sub-fingerprint are formally defined as

$$F(n,m) = \begin{cases} 1 \ \ if \ \ E(n,m) - E(n,m+1) - \big(E(n-1,m) - E(n-1,m+1)\big) > 0 \\ 0 \ \ if \ \ E(n,m) - E(n,m+1) - \big(E(n-1,m) - E(n-1,m+1)\big) \le 0 \end{cases}$$

The most computationally demanding operation is the Fourier transform of every audio frame. In the down-sampled audio signal, 2048 point FFT is carried for 256 times to extract one fingerprint block.

2.2. Modification on the Algorithm

As presented earlier, the overlap factor 31/32 is quite high. The proposed algorithmic modification is based on this factor. Since energy of each band is slowly changed over time, it is possible that we get the energy of even frame by interpolating energies of adjacent odd frames. If we follow this scheme, simple math can show that the bits of the sub-fingerprint $F(n,m)$ are repeated twice.

$$
\begin{aligned}
&if\ \ E(n,m)-E(n,m+1)-\big(E(n-2,m)-E(n-2,m+1)\big)>0\\
&then\\
&\quad E(n,m)-E(n,m+1)-\big(E(n-1,m)-E(n-1,m+1)\big)\\
&\quad = E(n,m)-E(n,m+1)-\left(\frac{E(n,m)+E(n-2,m)}{2}-\frac{E(n,m+1)+E(n-2,m+1)}{2}\right)\\
&\quad = \frac{E(n,m)-E(n,m+1)}{2}-\left(\frac{E(n-2,m)-E(n-2,m+1)}{2}\right)\\
&\quad = \frac{E(n,m)-E(n,m+1)-\big(E(n-2,m)-E(n-2,m+1)\big)}{2}>0
\end{aligned}
$$

$$
\begin{aligned}
&Also,\\
&E(n-1,m)-E(n-1,m+1)-\big(E(n-2,m)-E(n-2,m+1)\big)\\
&=\left(\frac{E(n,m)+E(n-2,m)}{2}-\frac{E(n,m+1)+E(n-2,m+1)}{2}\right)-\big(E(n-2,m)-E(n-2,m+1)\big)\\
&=\frac{E(n,m)-E(n,m+1)-\big(E(n-2,m)-E(n-2,m+1)\big)}{2}>0
\end{aligned}
$$

Thus, by the definition of sub-fingerprint $F(n,m)$, we can safely repeat the value without having to do all the calculation each time. In theory, this modification speeds up the fingerprinting as twice as before since we skip sub-fingerprinting every even frame.

2.3. Weak Bit Analysis

Similar procedure can be applied to weak bit generation.

$$if\ \ \left|E(n,m)-E(n,m+1)-\big(E(n-2,m)-E(n-2,m+1)\big)\right|\ \ is\ \text{Minimum},$$

$$
\begin{aligned}
&\left|E(n,m)-E(n,m+1)-\big(E(n-2,m)-E(n-2,m+1)\big)\right|<\\
&\left|E(n,k)-E(n,k+1)-\big(E(n-2,k)-E(n-2,k+1)\big)\right|\ \ for\ \forall k\neq m
\end{aligned}
$$

$$
\begin{aligned}
&Now,\\
&\left|E(n,m)-E(n,m+1)-\big(E(n-1,m)-E(n-1,m+1)\big)\right|
\end{aligned}
$$

$$= \left| \frac{E(n,m) - E(n,m+1) - \left(E(n-2,m) - E(n-2,m+1)\right)}{2} \right|$$

$$\text{Thus,}$$

$$\left|E(n,m) - E(n,m+1) - \left(E(n-1,m) - E(n-1,m+1)\right)\right| <$$

$$\left|E(n,k) - E(n,k+1) - \left(E(n-1,k) - E(n-1,k+1)\right)\right| \quad \text{for } \forall k \neq m$$

$$\text{Similarly,}$$

$$\left|E(n-1,m) - E(n-1,m+1) - \left(E(n-2,m) - E(n-2,m+1)\right)\right| <$$

$$\left|E(n-1,k) - E(n-1,k+1) - \left(E(n-2,k) - E(n-2,k+1)\right)\right| \quad \text{for } \forall k \neq m$$

Eventually, the weak bits can be used twice without having to find them each time.

2.4. Error Conditions

Let's say that we are interested in F(n, m) and F(n+1, m). Then, there are four possible error conditions when we adopt the new algorithm.

	Correct Values		Error Values	
	F(n, m)	F(n+1, m)	F(n, m)	F(n+1, m)
Error Case 1	0	1	0	0
Error Case 2	1	0	0	0
Error Case 3	0	1	1	1
Error Case 4	1	0	1	1

For simple explanations, we can plot the following table.

G1	G2		…	…				G3	G4

---→time

Each cell consists of 64 (=2048/32) samples. The shaded cells in each row means one frame (2048 samples). 1st and 2nd rows will be used to calculate F(n, m), and 2nd and 3rd rows will be used to get F(n+1, m) in then original algorithm. Now, let's define the following variables.

a_1 : Energy of band m at G1. b_1 : Energy of band (m+1) at G1.

a_2 : Energy of band m at G2. b_2 : Energy of band (m+1) at G2.

a_3 : Energy of band m at G3. b_3 : Energy of band (m+1) at G3.

a_4 : Energy of band m at G4. b_4 : Energy of band (m+1) at G4.

Now, it is possible to show that correct F(n, m) and F(n+1, m) are set by the following formulas:

$$F(n,m):(a_3 - b_3) - (a_1 - b_1)$$
$$F(n+1,m):(a_4 - b_4) - (a_2 - b_2)$$

We can also show that the modified algorithm will set F(n, m) and F(n+1, m) as the arithmetic mean of the above two. Thus, if the sign of the two are different, then the error will occur.

3. EXPERIMENTAL RESULT

The modified algorithm produced quite promising recognition results compared to the original algorithm. Two types of sound samples were used in the experiments. High quality sound samples were picked from various American pop songs.

mp3 file → PC speaker → PC microphone → wave file → test app

<Low Quality Sound Generation Steps>

Low quality sound samples were made by recording original pop songs played through PC speaker.

<Averaged Bit Error Rate from Production Service>

	Original Algorithm	Fast Algorithm
High quality sound samples (438 samples)	4.1 (%)	5.2 (%)
Low quality sound samples (904 samples)	22.3 (%)	22.6 (%)

With high quality sounds, the modified, fast algorithm showed 0.9% higher bit error rate. The bit error rate difference was less noticeable with low quality sounds.

4. DISCUSSION

In this paper, modified and fast Philips fingerprinting algorithm is presented. It shows that the modified algorithm produces fingerprints as twice as faster than the original algorithm by sacrificing within 1% fingerprint quality. The differences in the fingerprint quality are quite negligible with real world sound samples.