Content-Based Advertisement Detection

Patrick Cardinal, Vishwa Gupta, Gilles Boulianne

Centre de Recherche Informatique de Montréal (CRIM), Montréal, Canada
{patrick.cardinal,vishwa.gupta,gilles.boulianne}@crim.ca

Abstract

Television advertising is widely used by companies to promote their products among the public but it is hard for an advertiser to know if its advertisements are broadcast as they should. For this reason, some companies are specialized in the monitoring of audio/video streams for validating that ads are broadcast according to what was requested and paid for by the advertiser. The procedure for searching specific ads in an audio stream is very similar to the copy detection task for which we have developed very efficient algorithms. This work reports results of applying our copy detection algorithms to the advertisement detection task. Compared to a commercial software, we detected 18% more advertisements and the system runs at 0.003x of real-time.

Index Terms: Advertisement detection, fingerprint, GPU

1. Introduction

Television advertising is widely used by companies to promote their products among the public. Worldwide, the TV and radio advertisement market was valued at over 214 billion dollars in 2008. In the US alone, TV and radio advertisements amounted to over 82 billion dollars in 2008. With all that money at stake, the advertiser is entitled to ascertain that its television advertising campaign is broadcast as requested and paid for.

Currently, monitoring of ad campaigns is offered as a service by many companies worldwide. Some companies use watermarking for automated monitoring of ads. In watermarking, they embed a unique code in the audio or the image before it is broadcast. This code can then be retrieved by their watermark monitoring equipment. Watermarking every commercial for subsequent monitoring by specialized equipment is however expensive. In addition, watermarking only allows companies to monitor their own ads and they cannot follow the campaigns of their competitors for business intelligence.

Another approach is the use of a content-based method that allows ad detection without the aforementioned constraints imposed by watermarking. Several works have been published dealing with content-based commercial detection. Most of these use repetition of sequences and/or video and audio features such as black frames or change in energy to detect advertisements in the broadcast stream [1, 2]. These features do not however discriminate between specific commercials.

Lienhard et al. [3] use the color coherence vector as a fingerprint to find known advertisements. They report that all commercials were recognized with no false alarms. However, the process is slow since recognition time is 0.9 of real-time, without taking into account the computation of color coherence vector values. According to them, the whole process could run in real-time with a highly optimized version. Still, that is too slow for the purpose of searching ads on several channels in real-time.

Li et al. [4] search for unknown advertisements but use a color histogram-based fingerprint to detect those that have been seen before. They report pretty good results but the system is a bit slow (58% of real-time) for processing large amounts of data. No specific results are reported for the detection of stored advertisements.

In 2009, CRIM participated in the TRECVID 2009 evaluation [5]. The evaluated task involves searching for transformed audio queries in over 280 hours of test audio. The queries were transformed in seven different ways; three of these involved mixing unrelated speech to the original query, making it a much more difficult task than advertisement detection. The evaluation results show that using video stream for detecting specific segments is much less efficient both in terms of accuracy and processing time. Consequently, we worked only with the audio stream for the ads detection.

To make search fast enough, the matching process should use fingerprints allowing fast-matching computations. The 32 bit energy difference [6] is such a feature parameter. However, it does not occur frequently enough that we can rely on exact match counts. If we reduce the feature vector from 32 to 15 bits, then it occurs frequently enough that we can rely on the matching frame counts as a similarity measure between two segments.

This work was supported in part by the Natural Science and Engineering Research Council of Canada (NSERC).

For this reason, we experimented with the energy difference parameter used in audio copy detection [7]. We found that this parameter is very robust to various query transformations in the 2008 TRECVID copy detection evaluation [8] and leads to very fast computing.

We also experimented with a new fingerprint, called nearest-neighbor (NN), that maps frames of test audio to nearest query frames. We show that this mapping is more robust to query transformations than the energy difference fingerprint. It produces significantly fewer false alarms, and matching counts for correct matches are higher. It reduced the missed segments from 7.7% to 1.6% for the 2008 queries. This new fingerprint can also be efficiently computed in a graphic processing unit (GPU) resulting in an accurate and fast copy detection.

The overall success of this approach prompted us to extend it to advertisement detection. The paper is organized as follows: Section 2 describes the fingerprints used and how they are used in the advertisement detector; section 3 presents results and we conclude in section 4.

2. Detection Process

The task is to find specific advertisements (for which we have an example) in a large amount of audio data. We present a fingerprint matching system and 2 specifics fingerprints.
2.1. Fingerprint Matching

Advertisement detection is accomplished by computing a fingerprint for each frame of the advertisement. The fingerprint is also computed for each frame of the audio to be analyzed (program). Basically, the search is done by moving the advertisement \((n)\) frames over the program \((m)\) frames and counting the number of fingerprint matches for each possible alignment of the advertisement and program, as illustrated in figure 1. In this example, in which the advertisement is aligned at frame 1, the match starts at frame 3 and ends at frame 7 with a score of 3, since there are 3 matching fingerprints.

![Advertisement fingerprints](figure1.png)

Figure 1: An example of matching an advertisement to a program.

From the \(m - n\) alignments, only those with a count greater than a fixed threshold are considered. In our case, we used a threshold optimized for the copy detection task. The remaining alignments are then filtered according to the following rules:

**Extension**

Consider two alignments \(a_1\) and \(a_2\). Both alignments are synchronized if

\[
| (pStart[a_1] - pStart[a_2]) - (aStart[a_1] - aStart[a_2]) | \leq 2
\]

where \(pStart[a]\) and \(aStart[a]\) are respectively the first matching frame in the program and the first matching frame in the advertisement for the alignment \(a\).

If two alignments are synchronized, the one with the lower count is eliminated and its count is added to the remaining one.

**Overlap**

Two alignments \(a_1\) and \(a_2\) overlap if one of the following conditions is met:

\[
\begin{align*}
pStart[a_2] & \leq pStart[a_1] & \text{and} & pEnd[a_2] & \geq pStart[a_1] \\
pStart[a_2] & \leq pEnd[a_1] & \text{and} & pEnd[a_2] & \geq pEnd[a_1] \\
pStart[a_2] & \geq pStart[a_1] & \text{and} & pEnd[a_2] & \leq pEnd[a_1]
\end{align*}
\]

where \(pStart[a]\) and \(pEnd[a]\) are respectively the first and last matching frame in the program for the alignment \(a\). When two alignments overlap, the one with lower count is eliminated.

2.2. Advertisement detector

The advertisement detector uses two types of fingerprints for accurately detecting advertisements in one or more programs. Figure 2 shows a global view of the detection process.

![Advertisement detector process](figure2.png)

2.2.1. Energy-Difference fingerprint

The first type of fingerprint is referred to as the energy-difference fingerprint. Basically, 15 bits/frame are extracted from the audio signal. In a first step, the audio signal is low-pass-filtered to 5.5 KHz; pre-emphasized with a coefficient of 0.97 and divided into 25 ms Hamming windows with 10 ms frame advance. The Fourier transform spectrum between 300 Hz and 5000 Hz is divided into 16 bands using mel-scale spaced triangular windows and the energy is computed in each band. The energy differences between the bands are used to compute the fingerprint. If \(EB(n, m)\) represents the energy value of the \(n^{th}\) frame at the \(m^{th}\) band, then the \(m^{th}\) bit \(F(n, m)\) of the 15-bit fingerprint is given by

\[F(n, m) = 1, \text{ if } EB(n, m) - EB(n, m + 1) > 0, \]

Otherwise, \(F(n, m) = 0\).

The search process is described in more detail in [8]. This method is very fast and produces good results. Moreover, the search algorithm is very easy to parallelize on multicore/distributed systems (Table 1) since each advertisement can be computed independently. The drawback of this algorithm is the number of false alarms it produces.

<table>
<thead>
<tr>
<th>Number of threads</th>
<th>CPU time (min:sec)</th>
<th>Elapsed time (min:sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11:59</td>
<td>11:59</td>
</tr>
<tr>
<td>2</td>
<td>11:58</td>
<td>6:08</td>
</tr>
<tr>
<td>4</td>
<td>14:29</td>
<td>3:42</td>
</tr>
</tbody>
</table>

Table 1: Performances of energy difference fingerprint on a quad core CPU. The program searches for 1379 advertisements over 51 hours of audio.

2.2.2. Nearest-Neighbor Fingerprint

The second type of fingerprint maps each frame of the program to the closest frame of the advertisement. For computing this measure of closeness, we compute 12 cepstral coefficients and normalized energy and their first derivative for a total of 26 features. The distance between program frame and an advertisement frame is defined as \(\sum_{i=1}^{n} |a_i - p_i|\) where \(a_1, ..., a_n\) are the cepstral parameters for an advertisement frame and \(p_1, ..., p_n\) are the cepstral parameters for a program frame. To each program frame is associated its closest audio advertisement frame. This process is depicted by Algorithm 1. Once each program frame has been labeled with the closest advertisement frame, matching proceeds as in Figure 1.
Computing the closest advertisement frame for each program frame is computationally intensive. However, note that the search for the nearest advertisement frame can be done independently for each program frame. Consequently, an alternate processor, specialized in parallel computations, can be used to outperform the speed offered by modern CPU.

Modern graphic cards incorporate a specialized processor called Graphics Processing Unit (GPU). A GPU is mainly a Single Instruction, Multiple Data parallel processor that is computationally powerful, while being quite affordable. Table 2 shows the performances of GPU over CPU for the NN fingerprint task. The GPU improves speed by a factor of 70.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>464 hours</td>
</tr>
<tr>
<td>GPU</td>
<td>6.5 hours</td>
</tr>
</tbody>
</table>

Table 2: Performances of nearest-neighbor fingerprint on GPU. The program searches for 1379 advertisements over 51 hours of audio.

2.2.3. CUDA development framework

We have implemented the NN computation module in CUDA, a development framework for NVidia graphic cards [9]. The CUDA framework shows the graphic card as a parallel coprocessor for the CPU. The development language is C with some extensions.

A program in the GPU is called a kernel and many of them can be concurrently launched. A kernel is made up of configurable amounts of blocks, each of which consists of a configurable amount of threads. At execution time, each block is assigned to a multiprocessor in which 16 threads are concurrently executed. More than one block can be assigned to a given multiprocessor.

There are two kinds of memory. The first is the global memory which is accessible by all multiprocessors. Since this memory is not cached, it is important to ensure that the read/write memory accesses by a multiprocessor are coalesced in order to improve the performance. The second kind of memory is the shared memory which is internal to multiprocessors and is shared within a block. This memory, which is a lot faster than the global memory, can be seen as user-managed cache.

2.2.4. Nearest-Neighbor Kernel

Figure 3 shows how the computation of the NN is calculated in the GPU. In this figure, \( t_{ad} \) denotes the thread identifier for which the range is \([0, n]\), where \( n \) is the number of threads in the block. The value of \( blockId \) has the same meaning for the blocks. In this case, the number of blocks is the number of program frames divided by 128.1(304,115),(991,996)

![Figure 3: Nearest-Neighbor computation in the GPU](image)

Firstly, the program frames are divided into sets of 128 frames. Each set is associated with a multiprocessor running 128 threads. Thus, each thread computes the closest advertisement frame for its associated program frame.

Each thread in the multiprocessor downloads one advertisement frame from global memory. At this time, each thread can compute the distance between its program frame and all of the 128 advertisement frames now in shared memory. This operation corresponds to lines 4–11 of algorithm 1. When all threads are finished, the next 128 advertisement frames are downloaded and the process is repeated.

To increase performance, it is possible to concurrently process several programs. The search algorithm is described in more detail in [8, 10]. Note that to ensure that program frames can be stored in GPU registers (which allows considerably much faster access than shared memory), only the first 22 features have been used.

The NN fingerprint is more accurate than the energy-difference fingerprint. Comparing Table 1 and 2, it can be seen that the NN is much slower when a large set of data is considered. Another approach is to combine both fingerprints in a two pass system.

2.2.5. Combining Both Fingerprints

As shown in Figure 2, the first step is advertisement detection using the energy-difference fingerprint. This step outputs, for each advertisement in the database, a list of programs in which it was found. Then, the NN fingerprint is used to rescore each program. For efficiency, all programs are processed by the GPU concurrently. The result is the list of advertisements found in all the programs.

Algorithm 1: Nearest Neighbor computation

```
foreach \( prg \) \in program do
min \( \leftarrow \infty \)
foreach \( ad \) \in advertisement do
    \( d \leftarrow 0 \)
    for \( coeff \leftarrow 1 \) to \( n \) do
        \( d \leftarrow d + |f_{prg}[coeff] - f_{ad}[coeff]| \)
    end
    if \( d < min \) then
        \( min \leftarrow d \)
        \( results[prg] \leftarrow ad \)
    end
end
```

```
3. Experimental Results

3.1. Experimental setup

The test data for advertisement detection contains 51 hours of Canadian broadcast (French and English) divided in one hour segments. The advertisement database contains 1379 advertisements with an average length of 25.8 seconds.

Experiments have been conducted on a Intel Core 2 quad at 2.6MHz. The GPU used is the Nvidiag GeForce GTX 280 which contains 240 stream processors and has 1024 MB of RAM.

3.2. Results

In this section, we compare results yielded by fingerprints previously described and a proprietary fingerprint used by an advertisement monitoring company (our baseline). The threshold value used in the search algorithms has not been optimized; we directly used those of the TRECVID evaluation. It is also possible to discard a detected advertisement when the match begins too late in it. We did not use this threshold in this experiment. Table 3 summarizes the detection accuracies and execution times in seconds per hour of analyzed audio.

Unfortunately, we don’t know the actual number of ads in the test data. However, in copy detection, the NN fingerprint finds 99.3% of queries [8] in a clean signal environment. To confirm these results, we have processed a one hour TV show and the NN fingerprint found all the advertisements.

Table 3: Performances of advertisement detector.

<table>
<thead>
<tr>
<th>Fingerprint</th>
<th>Ads Detected</th>
<th>False Alarms</th>
<th>Subst.</th>
<th>Processing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>329</td>
<td>11</td>
<td>3</td>
<td>180 s/h</td>
</tr>
<tr>
<td>Energy diff.</td>
<td>393</td>
<td>22</td>
<td>2</td>
<td>4.4 s/h</td>
</tr>
<tr>
<td>NN</td>
<td>401</td>
<td>20</td>
<td>0</td>
<td>458 s/h</td>
</tr>
<tr>
<td>Combined</td>
<td>393</td>
<td>7</td>
<td>2</td>
<td>9.5 s/h</td>
</tr>
</tbody>
</table>

Table 4: No false alarms advertisement detection

When running our experiments, we noted that the audio quality of advertisements in the database and audio stream was different. The database advertisements were often of higher quality than the analyzed audio. Our results show that our fingerprints perform robustly towards differences in sound quality.

4. Conclusion and Future Work

We presented an advertisement detector based on our copy detection system. Our system detected 18% more ads than the commercial software we used as a baseline and runs at 0.003x the real-time. The results show that in the event a GPU is not available, results produced by the energy difference fingerprint method are good enough to be used in a commercial system.

The next step is the detection of all advertisements in an audio stream. This system could be used to monitor publicity campaigns of competitors or to remove advertisements in order to produce a transcription suitable for indexing the essential audio content.

5. References