Extracting Repeats from Media Streams

Cormac Herley
Microsoft Research
One Microsoft Way
Redmond, WA 98052
Email: c.herley@ieee.org

Abstract—**Many media streams consist of distinct objects that repeat. For example broadcast television and radio signals contain advertisements, call sign jingles, songs and even whole programs that repeat. The problem we address is to explicitly identify the underlying structure in repetitive streams and deconstruct them into their component objects. Our architecture assumes no *a priori* knowledge of the streams, and does not require a pre-trained database. Everything the system needs is learned on the fly. We demonstrate that using a modestly capable computer it is perfectly feasible to identify in realtime repeating objects that occur days or even weeks apart in audio or video streams. We outline the algorithms, enumerate several applications and present results from real streams.**

I. INTRODUCTION

Rather than having infinite innovation many media streams are quite repetitive. They contain objects that recur with essentially no change. Advertisements on broadcast radio and television stations, and songs or video on music channels are some of the obvious examples. An interesting approach by Lienhart *et al* [7] is to detect a *known* set of commercials. That is, given a library of, say, K commercials (*i.e.* short sequences of frames that can be expected to recur in a stream) they calculate a fingerprint, which is a function of this sequence of frames. This fingerprint, which they base on the Color Coherence Vector [8], can be compared to detect recurrence of the already known commercials. Other interesting approaches are [5], [2].

By contrast we propose explicit detection of *unknown* repeating segments in the stream without making assumptions of the nature of the objects. For example, we will not need to assume that any of the objects we seek have different audio or video characteristics from the non-repeating portions of the stream, and we will only assume that repeating objects (ROs) have some minumum length (*e.g.* they are at least 30s in length). We will not need to assume the existence of a library of already labeled or recognized objects, but rather will learn the repeat patterns on the fly.

In the next section we show how repeats may be identified in a stream using a scheme that is conceptually simple, but requires considerable compute and memory resources. In Section III we show that using low dimension representations of the stream and segmenting objects we can hugely reduce the computational complexity and remove the need to buffer large amounts of the stream. We show results on broadcast streams in Section IV.

II. SEARCHING FOR UNKNOWN REPEATING OBJECTS

Suppose that our media stream is $s(t)$, and this contains embedded repeating objects. We assume that the objects undergo relatively minor copy-to-copy variation; i.e. successive repeats of an object in a stream separated by time will obviously experience different channel deformations, but will otherwise be almost identical. We further assume that objects are generally non-overlapping.

We'll model the stream as being made up of K repeating objects, which are played with relative frequencies determined by the probabilities p_i. That is, the stream is constructed by choosing an object from the library, and the $i - th$ object has independent probability p_i of being chosen when a decision is made. For generality we will also assume that only a fraction r of the stream consists of repeating objects; for example $r = 0.9$ would imply that one tenth of the stream was non-repeating content that separated repeating objects from each other. While a little simplistic, this model is sufficiently general to capture the behavior of real streams reasonably accurately, as we shall see from the experiments on real streams in Section IV.

A. Determining Repeats

First, observe that it is simple to *verify* that an object at t_i also occurs at t_j. This can be done, for example, by taking the cross-correlation of windowed sections of the stream centered at t_i and t_j. If the cross-correlation:

$$[w(t) \cdot s(t - t_i)] * [w(t) \cdot s(t - t_j)]$$

has a sufficiently large peak (compared with the autocorrelation of either section) we would determine that they are approximately the same. This is only the simplest example: the Color Coherence Vector used in [7], [8] or the fingerprint algorithms used in [1], [4] are among other functions that could be used.

B. Searching for Unknown Repeating Objects

Seeking K *known* objects (much as is done in [7]) would require comparing each incoming block against the K known objects. This would require $K/2$ comparisons per incoming block on average. Thus the complexity of determining whether *known* objects are present in a stream is directly related to the size of the database of sought objects.

The case where the objects are *unknown* is more complex, since we must learn first what the objects are, and then find all

of their recurrences. A first approach, is to break a finite stream into N blocks and compare the current block with every other:

```
[found, jposn] = searchBuffer(iPosn){
    jposn = iPosn;
    while (jposn > iPosn - NL)
    if (ApproxEq(strm(iposn), strm(jposn))
        AddToRepeatsList(iposn, jposn);
        found = true;
        break;
    end
    jposn -= L;
end }
```

The blocks are spaced L apart. We assume that our Ap-proxEq() routine determines whenever two blocks contain the same object with almost no false positives or negatives. We assume that AddToRepeatsList() performs whatever actions are desired when the fact that an object occurs at both iposn and jposn has been verified. The complexity of the algorithm is determined by the $N-1$ calls to ApproxEqual(). So clearly, a buffer of duration NL can be searched in realtime only if $N-1$ calls to ApproxEqual() can be computed in time L.

C. Analysis of probability of objects repeating in the buffer

For an object to be found by searchBuffer() it must appear twice in the buffer. Assume objects have average duration E. Then the average time between repeat copies of object i will be $E/(rp_i)$. Suppose the length of our buffer is B. So all objects for which $E/(rp_i) < B$ have a good chance of occurring twice in the buffer. We now show that the probability that two copies of an object appear in the buffer increases significantly as the buffer evolves.

At any given time there will be $M = \lfloor B \cdot r/E \rfloor$ repeating objects in the buffer. Call $Pr(i, t, \text{found})$ the probability that two copies of object i have been in the buffer simultaneously at least once by time t (and thus the object has been found by routine searchBuffer()). Clearly, at $t = B$, when the buffer first fills,

$$Pr(i, B, \text{found}) = \sum_{j=2}^{M} \binom{M}{j} p_i^j \cdot (1 - p_i)^{M-j}.$$

We now wish to know how $Pr(i, t, \text{found})$ evolves for increasing t. Since this is a cumulative probability it can only increase. Recall that in time E/r one new object enters the buffer, and another drops out. The increase brought about is:

$$Pr(i, t + E/r, \text{found}) - Pr(i, t, \text{found}) =$$
$$[1 - Pr(i, t, \text{found})] \cdot Prob\{\text{i in buffer once|Not twice}\}$$
$$\cdot Prob\{\text{Incoming is i}\} \cdot Prob\{\text{Outgoing is not i}\}$$
$$= [1 - Pr(i, t, \text{found})] \cdot \frac{\binom{M}{1} p_i (1 - p_i)^{M-1}}{\binom{M}{1} p_i (1 - p_i)^{M-1} + (1 - p_i)^M}$$
$$\cdot p_i \cdot \frac{M - 1}{M}.$$

In Figure 1 we graph how $Pr(i, t, \text{found})$ evolves for various scenarios. Using $B = 1440$ minutes (*i.e.* one day) and $r =$
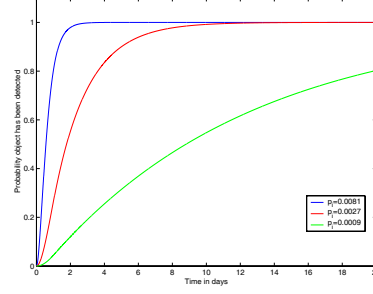

Fig. 1. Probability that two or more copies simultaneously appear in a one day buffer for objects with various probabilities p_i. Observe that even an object that repeats on average every $E/rp_i = 3$ days has a 50% probability of being detected after 10 days

0.9 and $E = 3.5$ minutes we show $Pr(i, t, \text{found})$ for $p_i = 0.0081, 0.0027$ and 0.0009. Objects with these probabilities would have average repeat intervals $E/(rp_i)$ of 8 hours, 1 day an 3 days respectively. As can be seen from Figure 1, even objects that occur on average with intervals larger than the searched buffer have high likelihood of being detected if we wait long enough.

III. REDUCING THE COMPLEXITY OF DETERMINING REPEATS

Calculating cross-correlations on video data, or even audio at 44.1k samples per second is computationally very expensive. But we can exploit the fact that both video and audio contain much redundancy to reduce the complexity.

A. Low Dimension Representations of Audio

Since media streams primarily consist of audio alone or video accompanied by audio, we restrict attention to low dimension representations for audio objects. For an RO to occur in a video stream it is necessary, though not sufficient that an RO occur in the associated audio component of the stream. Hence we'll concentrate on audio streams; this spares involving the video in search process.

A commonly used tool for audio analysis is to split the audio into critical bands [6] (sometimes known as Bark bands). For example a commonly used set for audio splits the signal into 25 bands with band centers at $\{100, 200, 300, 400, 510, 630, 770, 920, 1080, 1270, 1480, 1720,$ $2000, 2320, 2700, 3150, 3700, 4400, 5300, 6400, 7700, 9500,$ $12000, 15500, 22200\}$ Hz. These are narrow frequency selective channels, and can of course be sampled at a much lower frequency than the overall audio signal. In fact, we take the energy of the signal in the bark band centered at 770 Hz, lowpass filter it, and then sample at 11 samples/sec. We call this waveform BB7. This of course is no longer sufficient to give even a crude representation of the signal; however it suffices for recognition.

By way of example in Figure 2 we show two sections of the energy of BB7 overlaid. Each segment represents slightly greater than 10 minutes of audio. They are different copies of the same object recorded from an FM radio station at different

times. As can be seen in the center portion both copies are approximately equal, while at the beginning and end they differ.

1) Experimental: To test the hypothesis that BB7 forms a good low dimension representation of audio we performed the following experiment. We recorded an FM radio station for a period of seven days, and calculated BB7 for the entire stream. We randomly selected 500 6 minute segments; some of the segments were voice, some music, some a mixture of voice and music. We then formed the cross-correlation of each with every 6 minute segment from the entire stream. When the cross-correlation indicated a match we examined the two segments manually to determine whether or not they actually matched. Some of the objects had no matches (other than the segment of their own occurrence in the stream) and others occurred as many as 20 times. In no case was a match indicated where data did not correspond to the same object. This suffices to indicate that the false positive rate of using cross-correlations of BB7 is very low.

Determining the false negative rate is harder. To accomplish this we hand parsed a 48 hour section of the stream; i.e. hand labeled every repeating object greater than 2 minutes in length. We randomly selected 50 objects from the first 24 hours of labeled stream, and for each of them: cross-correlated a 6 minute segment of BB7 centered on the object with the entire second 24 hours of labeled stream. No false negatives were found in the labeled stream.

Thus, we find that BB7 is a suitable low dimension representation of the signal. In going from 44.1 k samples/sec to 11 samples/sec we have achieved a 4000-fold reduction in the data rate, with no meaningful loss in recognition ability.

B. Identifying the boundaries of found objects

We have seen that repeating objects can be identified in a number of ways. Cross-correlations, Audio fingerprints [1], [4] or Color Coherence Vectors [8], [7] (in the case of video) are among the approaches that work. Any of these methods will determine that the data in the stream at locations t_i and t_j are approximately similar. Of course to carry out any action on an RO, *e.g.* delete it from the stream, copy it to disk and so on, we need to know its precise endpoints. We need to know, for example, whether its 10s or 3 minutes long before any action can be carried out.

Determining the endpoints becomes simple however if we have access to the stream. Conceptually, if we align the two portions of the waveform we can trace backward toward the beginning and forward toward the end to determine where the two copies diverge, giving the boundaries of the object. Recall, in Section III-A, we found that using the full rate stream was unnecessary to determine when matches occurred. In Figure 2 we align, normalize and overlay two segments of a stream containing the same object. As can be seen there is substantial (though not precise) overlap between the two occurrences. This allows us to state with reasonable accuracy that the segments coincide approximately between samples 5800 and
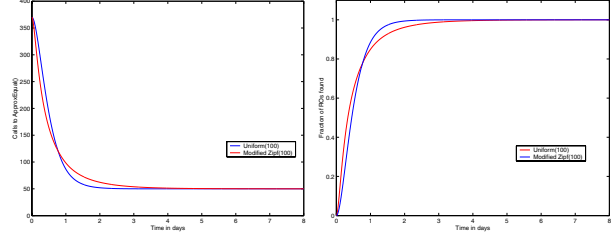

Fig. 3.   Improving the efficiency of searching for unknown repeating objects. A buffer of length one day is assumed, for the two distributions shown. (a) Number of calls to ApproxEq(). Observe that the number of calls begins as N, but rapidly converges to $K/2$. (b) Fraction of the ROs played in a stream found. After two days almost every RO has already been found.

8300 and thus (since BB7 is sampled at 11 samples/sec) the object is approximately 227 seconds in duration.

1) Experimental: To test the accuracy of our segmentation we randomly selected 100 of the repeating objects from the hand-labelled 48 hour stream referred to in Section III-A.1. We exhaustively searched for all matches in the following 48 hours. Of the 100 objects, 88 had matches, and some of them multiple matches. We first manually examined the stream and decided where the endpoints were and then calculated the end points using our tracing algorithm. In most cases the calculated endpoints corresponded accurately with those determined by hand. In only 6 of the 88 cases was the difference greater than 2s.

C. Increasing efficiency of search for Repeating Objects

The search strategy we introduced in Section II-B was essentially brute force: break a buffer of the stream into blocks and compare the current block with all past blocks in the buffer. Once an object is found however it can be added to a library of known objects. Subsequently, we search this library first, and search the remainder of the buffer only if we find no match in the library. The advantage is that after its second appearance each repeating object will be in the library and will be identified from there without having to search the buffer. So long as the library is smaller than the buffer this improves the search, and for repetitive streams this is the case. In addition the library of found repeating objects can be ordered by frequency of repetition, so that most common object are checked first; this further improves the efficiency. The algorithm is straightforward extension of the one in Section II-B and is explained in detail in [3].

We now analyze the improvement of this algorithm over the brute force approach in Section II-B. If the current block contains the $i-th$ object the number of calls to ApproxEqual() will on average be $K/2$ if that object is already in the library, and it will be less than $K + N - 1$ otherwise. Thus, it is less than

$$Pr(i, t, \text{found}) \cdot K/2 + [1 - Pr(i, t, \text{found})] \cdot (K + N - 1).$$

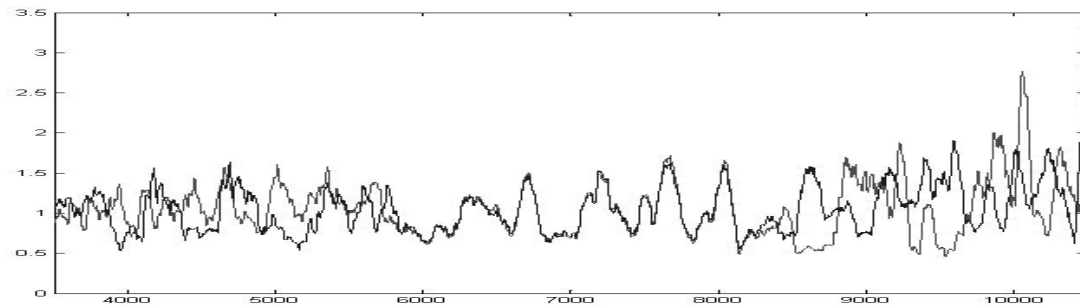Thus the average number of comparisons will be upper

Fig. 2. Example indicating how the boundaries of an object can be calculated once two or more copies have been found. Overlay of the first and second instance of the object once they have been aligned. The points where the two streams diverge at the beginning and end are the endpoints of the object.

bounded by

$$\sum_{i=0}^{K-1} p_i \cdot \big(Pr(i,t,\text{found}) \cdot K/2 + [1 - Pr(i,t,\text{found})] \cdot (K+N-1) \big).$$
(1)

A tighter bound is derived in [3]. As shown on Figure 1 $Pr(i,t,\text{found})$ tends to one for increasing time; hence, the average number of comparisons tends to $K/2$. That is, as the RO library fills most objects are found from the library and the necessity of searching the buffer becomes rarer and rarer. This is shown in Figure 3 (a). Recall from Section II that the complexity of searching for *known* objects was linear in the size of the library K, while searching for *unknown* objects was linear in the size of the buffer N. Now we have shown that the complexity is initially proportional to N, but quickly converges to $K/2$. Thus, for repetitive streams, the complexity of identifying *unknown* repeating objects converges to the complexity of identifying a collection of *known* repeating objects.

We take the example of a stream composed of $K = 100$ objects drawn from a uniform and a Zipf distribution. We choose $r = 0.9$ and the average object is of length $E = 210s$, and a buffer of length $B = 24 * 60 * 60s$. In Figure 3 (a) we show how the average number of calls to ApproxEq() as predicted by Equation (1) evolves over time. In Figure 3 (b) we show the fraction of the K objects that comprise the stream that have been found as a function of time. Using a buffer of length one day, most objects have been found within two days.

IV. VERIFICATION ON REAL STREAMS

We implemented the algorithm above and ran on several broadcast streams. Using a Pentium III 750 MHz PC with 512 MBytes of RAM we were able to search a buffer of 3 days in real-time using only approximately 20% of CPU. An FM receiver was connected to the line-in of the PC. Our low dimension representation was used to find repeats and detect endpoints of objects once found. An example is shown in Figure 4 of listening to a Seattle pop music radio station for 6 days. We plot the rate at which unique new objects were found, and the distribution of frequencies. Clearly objects are found rapidly at first and then tail off. Also observe that the
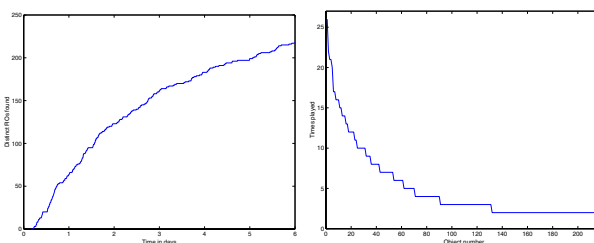

Fig. 4. The Seattle pop radio station KNDD FM 107.7 analyzed for a period of 6 days. (a) Number of unique ROs found as a function of time. Many objects are found initially, but the rate slows as fewer objects remain to be found. (b) Number of times that unique ROs were played in stream. A small number of popular objects are played very frequently and the stream is very repetitive.

graph of play frequencies is quite peaked: a small number of popular objects are played very often, with a larger number of objects being played relatively infrequently.

A. Acknowledgments

REFERENCES

[1] C. J. C. Burges, J. C. Platt and S. Jana. Distortion descriminant analysis for audio fingerprinting. *IEEE Trans. on Speech and Audio Processing*, 11:165–174, 2003.

[2] A. Hampapur and R. Bolle. Feature based indexing for media tracking. *Proc. ICME*, 2000.

[3] C. Herley. Argos: Automatically extracting repeating objects from multimedia streams. *IEEE Trans. on Multimedia*, 2003. Submitted. Available as MSR-TR-2004-02.

[4] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system. *Proc. Intl Conf on Music Information Retrieval*, 2002.

[5] A. Jaimes and J. R. Smith. Semi-automatic, data-driven construction of multimedia ontologies. *Proc. IEEE ICME*, 2003.

[6] N. S. Jayant and P. Noll. *Digital Coding of Waveforms*. Prentice Hall, Englewood Cliffs, NJ, 1984.

[7] R. Lienhart, C. Kuhmuench, and W. Effelsberg. On the detection and recognition of television commercials. *Proc. Intl. Conf. on Multimedia Computing and Systems*, pages 509–516, June 1997.

[8] G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. *Proc. ACM Multimedia*, pages 65–73, Nov. 1996.