ARGOS: Automatically Extracting Repeating Objects From Multimedia Streams

Cormac Herley, *Member, IEEE*

***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 de-construct them into their component objects. Our algorithm exploits dimension reduction techniques on the audio portion of a multimedia stream to make search and buffering feasible. Our architecture assumes no a priori knowledge of the streams, and does not require that the repeating objects (ROs) be known. Everything the system needs, including the position and duration of the ROs, is learned on the fly. We demonstrate that it is perfectly feasible to identify in real-time ROs that occur days or even weeks apart in audio or video streams. Both the compute and buffering requirements are comfortably within reach for a basic desktop computer. We outline the algorithms, enumerate several applications and present results from real broadcast streams.**

***Index Terms*—Audio fingerprint, low-dimension representation, multimedia, repeats, segmentation.**

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. Other examples are station call-signs and jingles, signature tunes of particular television programs, news footage of noteworthy events (e.g., a clip from a State of the Union speech in the U.S. will typically air many times in the days following the speech) and even entire radio or television programs (e.g., many NPR member stations broadcast syndicated programs more than once). These objects can vary from a few seconds in length to several hours. The gaps between successive copies of the same object can be as little as a few minutes or as great as weeks or even months. The objects may repeat in a very regular fashion (e.g., the signature tune of a news program occurring at defined times throughout the day) or with no apparent regularity (e.g., particular songs occurring in a radio broadcast at the discretion of the disk jockeys or playlist generators). Some objects may be repeated very frequently for a short period of time and then are seldom seen again (e.g., a clip of a current newsworthy event) others may be repeated over the spans of years (e.g., signature tunes of serial programs). Since there is such diversity in the length of repeating objects (ROs), and the frequency and regularity of their recurrence few useful generalizations about them can be drawn.

Much work has been reported on identifying, segmenting, and classifying particular classes of objects. For example, in applications such as television news summarization progress has been made in distinguishing between news-anchor scenes, live report scenes, weather reports, and so on [14], [22].

An interesting approach by Lienhart *et al.* [17] 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 [19], can be compared to detect recurrence of the already known commercials. However, the authors of [17] point out that any fingerprint which is tolerant of channel deformations has low dimension and discriminates well between different commercials will suffice. They also point out that the arrival of new, previously unknown commercials may be inferred, when, for example, an unknown 30-s sequence of frames occurs between two known commercials. Like [17], we will examine possible candidates for fingerprints. Unlike [17] however, we will not assume that a library of known ROs is available or make any assumptions about the duration or nature of new ROs as they appear in the stream. Another interesting approach along these lines is [16], which explores an efficient method to search for known objects in long streams or libraries. Both [17] and [16] have in common with our work that they are explicitly interested in very long sequences (i.e., days or weeks); the point of difference is that they take the set of sought objects to be known.

Considerable effort has addressed the question of identifying objects based on a set of features. In [8], for example, the authors describe a system to track multimedia content based on video features, while [18] uses a system of audio and visual cues. The authors of [1] report considerable success in distinguishing commercials from normal content by using an efficient vector based on the evolving color information. Many of the approaches to stream analysis make use of the growing body of work on audio analysis, querying and retrieval. See, for example, the object extraction work of [20], [24] and the similarity detection work of [2], [11]. Many of the approaches customizing multimedia streams have in common that they seek particular features of certain classes of objects [14], [22]. Video commercials are often preceded by two blank frames for example. Alternatively, they often work with a library of known objects [17]. An interesting example of searching for unknown ROs in a music stream is [10] though the authors there work

Manuscript received November 7, 2003; revised November 29, 2004. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. John R. Smith.

The author is with Microsoft Research, Redmond, WA 98052 USA (e-mail: c.herley@ieee.org).

Digital Object Identifier 10.1109/TMM.2005.861286

on MIDI data rather than a raw audio format. The literature on text string searching has numerous approaches [3], [23] that can serve as analogies for our work. Recent work in the database community [15] on finding matches in unbounded streams is also related, but lacks the media component. Searching multimedia databases is discussed in [6].

The Scene Transition Graph developed by Yeung *et al.* [25] identifies related scenes in a stream. Distinct scenes are identified by detecting shot boundaries and building a graph based on scene similarities. In [25], related scenes are clustered (e.g., closeups of the same person in a dialog scene). This differs from our work in that we are interested only in identifying segments of streams that are identical (other than channel deformations). Another interesting approach to building multimedia ontologies is [12].

A point of contrast between our work and many previous approaches is that we propose explicit detection of 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 nonrepeating portions of the stream, and we will only assume that ROs have some minimum length (e.g., they are at least 30 s 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. For fear of misunderstanding we emphasize that we seek only objects that repeat, and not "objects" in the sense that it is sometimes used in the literature: for example, a well-defined 30-s commercial that plays only once would never be found by ARGOS.

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 audio portion of the stream we can hugely reduce the computational complexity. Furthermore, by segmenting the objects we can form a library of objects as they are found and remove the need to buffer large amounts of the stream. Combining these improvements greatly reduces the computation and buffering required. In Section IV, we give an architecture of the system and in Section V we show the results of object detection and extraction performed on real broadcast radio and television streams. Numerous applications are enabled by the ability to decompose a stream into its component objects, most obviously the ability to customize a stream and make a library of objects for later viewing. A preliminary version of this work was reported in [9]. We call the system Automatic RepeatinG Object Segmentation (ARGOS), that also being the name of the dog who recognized Ulysses on his return, even though more than 20 years had passed.

A. Applications

Before demonstrating that a system capable of extracting repeat objects from streams can operate in real-time on a consumer level PC, it may be worthwhile to mention applications to motivate the work.

- *Commercial skipping:* in Section IV we give an architecture that allows skipping of objects in real-time
- *Building libraries of objects for later viewing:* the architecture of Section IV allows a user to automatically acquire copies of all of the ROs in a stream. Examples are shown in Figs. 7–10.
- *Customization:* streams can be adapted in real-time to user preferences by removing unwanted content and replacing with preferred content.
- *De-noising:* as more copies of an object are received, channel noise can be attenuated by de-noising techniques [21].
- *Querying and Indexing:* the library of ROs that is generated by our algorithm can be queried to determine if particular streams obtain particular objects.
- *Compression and archiving:* the library of ROs that is generated by our algorithm forms a dictionary that can be used to compress the stream efficiently. Since objects keep repeating, there is no need to compress each copy of each RO every time it appears.
- *Gathering statistics of object play frequencies on streams:* once we know how to decompose streams into their component objects reliably it is simple to then track the distributions of play frequencies. Examples on two FM radio streams are shown in Figs. 8 and 7 and on two broadcast television streams in Figs. 9 and 10. An application of these statistics to playlist generation is given in [26].
- *Broadcast monitoring:* since our algorithm builds a library of ROs and determines when they played, it is easy to verify, for example, that particular commercials played at particular times.

We elaborate on two of these applications.

1) Customization and Adaptation: Any stream that is made up of ROs can be customized if real-time identification of the objects is possible. The received stream $\mathbf{r}(t)$ can be delayed and altered so that the viewed stream $\mathbf{v}(t)$ corresponds closely to the desires and preferences. The algorithms and architecture we outline in this paper shows how this can be achieved in real-time with basic computing resources.

- Parents might wish to delete objects that they deem unsuitable for their children. For example, on a channel which plays music videos, parents might wish to prevent certain videos being played. A complete library of the ROs on the channel can be built as described in Section V. Each of the videos can be quickly viewed in turn and tagged as permitted or not. Of course, objects can not be tagged until they have repeated. However, as we show in Section V, many streams are extremely repetitive; in particular, for the FM and video music stations analyzed in Figs. 7, 8, and 10, almost all ROs had been found in a matter of five days or so, while those ROs continue to play for weeks or months before they drop from fashion. The ARGOS system can be used as a parental control to collect objects before they are seen by children. That ARGOS will have seen an object twice does not mean that the child has.
- Listeners and viewers might wish to store copies of favorite objects, or delete objects they do not like. For example, a listener to a Jazz radio station might alter the mix

of music played by specifying that some objects are never to be played, or by decreasing the frequency of play. Exactly such a scenario is described in Section IV and Fig. 6.
- Broadcasters might wish to substitute commercials suitable for local markets or tuned to the preferences of the viewer.

The method of associating an action with an object can vary. For example, in the parental control application a suitable UI might be as simple as a button to enable a parent while viewing an object to indicate that all future occurrences are to be deleted. Commercial substitution could be accomplished by the broadcaster associating a substitution action with an object based on the viewer's location and/or preferences.

2) Gathering Statistics: The ability to recognize ROs allows us to build a statistical picture of the composition of a stream. Basic examples are shown in Figs. 7 and 8, but various other statistics on the streams can be gathered. We can compile histograms of relative frequencies of objects, and compile a "Top 10" lists of the most popular objects.

TABLE I
DEFINITION OF TERMS USED

$\mathbf{s}(t)$	Media stream
$\mathbf{r}(t)$	Received stream $\mathbf{s}(t)+\mathbf{n}(t)$
$\mathbf{n}(t)$	Noise
$\mathbf{O}_i(t)$	Repeating Object $i=0,1\cdots K-1$
$L[\mathbf{O}_i(t)]$	Length of Repeating Object $\mathbf{O}_i(t)$
K	Number of Repeating Objects
r	Fraction of repeating content in $\mathbf{s}(t)$
$w(t)$	Length L window used for correlations
L	Length of block used for correlations
E	$\frac{1}{K}\sum_{i=0}^{K-1} L[\mathbf{O}_i(t)]$ Average Length of Repeating Objects
$R_{\mathbf{b}_i\mathbf{b}_j}(\tau)$	Correlation $<\mathbf{b}_i(t),\mathbf{b}_j(t-\tau)>$
ρ	$\max_{\tau,i\neq j} R_{\mathbf{b}_i\mathbf{b}_j}(\tau)$
T	Threshold used for correlations
NL	Buffer length
$Pr(i,t,\text{found})$	Probability that $\mathbf{O}_i$ has occurred twice in buffer by time t
p_i	Probability of choosing $\mathbf{O}_i$ when object added to stream

II. SEARCHING FOR UNKNOWN REPEATING OBJECTS

A. Model of Multimedia Streams

Without loss of generality,we will model our media stream $\mathbf{s}(t)$, as containing embedded ROs $\mathbf{O}_i(t)$ for $i=0,1,2\cdots,K-1$. The objects are played with relative frequencies determined by the probabilities p_i, and have lengths $L[\mathbf{O}_i(t)]$. That is, the stream is constructed by choosing an object from the library, and the ith 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 ROs; for example, $r=0.9$ would imply that one tenth of the stream was nonrepeating content that separated ROs from each other. 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.

The stream can be one dimensional such as audio, or multi-dimensional such as a combined audio and video stream. We further assume that objects are generally nonoverlapping, so the stream is created by concatenating objects rather than superimposing them. An example of such a stream might be written

$$\mathbf{s}(t)=\{\cdots|\mathbf{O}_{17}(t)|\mathbf{e}_d(t)|\mathbf{O}_9(t)|\mathbf{O}_{103}(t)|\mathbf{e}_d(t)|\mathbf{O}_{37}(t)|\cdots\}$$

where the symbol $|$ denotes concatenation of objects, and $\mathbf{e}_d(t)$ denotes a nonrepeating segment of duration d seconds. Clearly, if an object $\mathbf{O}_i(t)$ is repeated at times t_0 and t_1 in the stream, we will have

$$\mathbf{s}(t_0+t)=\mathbf{s}(t_1+t),\ \text{for}\ 0<t<L\left[\mathbf{O}_i(t)\right]. \tag{1}$$

We would like to clarify at the outset that for a multidimensional stream, following (1) an object repeats only if it repeats along all dimensions. That is if, for video, the same visual content is replayed, but with different audio accompaniment this is not a RO. For example, if on a bilingual TV broadcast, a 30-s commercial occurs with video content unchanged, but sometimes with audio in English and sometimes in Spanish, this will be regarded as two separate ROs rather than one. Similarly, if the audio repeats but the video is different this is not regarded as a RO. The stream received by the user will of course undergo channel deformations, which we model by additive noise

$$\mathbf{r}(t)=\mathbf{s}(t)+\mathbf{n}(t).$$

We assume that $|\mathbf{s}|\gg|\mathbf{n}|$. Thus, 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 the noise is small relative to the signal

$$\mathbf{r}(t_0+t)\approx\mathbf{r}(t_1+t),\ \text{for}\ 0<t<L\left[\mathbf{O}_i(t)\right]. \tag{2}$$

For convenience, the definition of terms used in this paper is given in Table I.

B. Determining Repeats by Correlations

We now address the problem of finding the ROs in a stream constructed as above. First, observe that it is simple to *verify* that an object at t_0 also occurs at t_1. This can be done, for example, by taking the time correlation of windowed sections of the stream centered at t_0 and t_1. Let the window $w(t)$ be defined by

$$w(t)=\begin{cases}1, & 0<t<L-1\\ 0, & \text{otherwise.}\end{cases}$$

Denote a windowed block of the received stream

$$\mathbf{b}_0(t)=w(t)\cdot\mathbf{r}(t-t_0). \tag{3}$$

We expect that if $\mathbf{b}_0(t)$ and $\mathbf{b}_1(t)$ contain the same object, there should be a strong peak in their correlation along time. We define the time correlation as

$$R_{\mathbf{b}_0\mathbf{b}_1}(\tau)=\left\langle\frac{\mathbf{b}_0(t)}{|\mathbf{b}_0(t)|},\frac{\mathbf{b}_1(t-\tau)}{|\mathbf{b}_1(t)|}\right\rangle. \tag{4}$$

There are many way of treating the boundary issues that arise when correlating finite blocks such as we do here (see, for example, [21]). Also, note that readers should not confuse the time correlation of a section of a video stream with the full multidimensional cross-correlation. $R_{\mathbf{b}_0\mathbf{b}_1}(\tau)$ is a function of only one variable, while the cross-correlation of an audio-visual stream would be three dimensional. If $\mathbf{b}_0(t)$ and $\mathbf{b}_1(t)$ contain the same object, we expect $\max_\tau R_{\mathbf{b}_0\mathbf{b}_1}(\tau)$ to be larger than if they do not. We now show under what conditions this is so.

If the block length L is greater than the average object length E, a block containing an RO will have the form

$$\mathbf{b}_0(t) = [\mathbf{a}(t)|\mathbf{O}_i(t)|\mathbf{b}(t)] \tag{5}$$

where $\mathbf{a}(t)$ and $\mathbf{b}(t)$ are the portions of the stream preceding and following the object. These may of course themselves be portions of other ROs or nonrepeating content. When we time correlate with another block

$$\mathbf{b}_1(t) = [\mathbf{c}(t)|\mathbf{O}_j(t)|\mathbf{d}(t)] \tag{6}$$

we require that

$$\max_{\tau, i=j} R_{\mathbf{b}_0\mathbf{b}_1}(\tau) > \max_{\tau, i\neq j} R_{\mathbf{b}_0\mathbf{b}_1}(\tau). \tag{7}$$

That is, the peak of the correlation when $i = j$ (the two blocks contain the same object) should be larger than any value when $i \neq j$ (the two blocks do not contain the same object).

Recall that the received stream $\mathbf{r}(t)$ is corrupted by noise so that two instances of an object $\mathbf{O}_i$ will differ slightly. Normalizing, and taking the time correlation

$$\begin{aligned}\max_{\tau, i=j} R_{\mathbf{b}_0\mathbf{b}_1}(\tau) &= \max_{\tau, i=j} \left\langle \frac{\mathbf{b}_0(t)}{|\mathbf{b}_0(t)|}, \frac{\mathbf{b}_1(t-\tau)}{|\mathbf{b}_1(t)|} \right\rangle \\ &\geq \left\langle \frac{\mathbf{O}_i(t)+\mathbf{n}_0(t)}{|\mathbf{O}_i+\mathbf{n}_0|}, \frac{\mathbf{O}_i(t-\tau)+\mathbf{n}_1(t-\tau)}{|\mathbf{O}_i+\mathbf{n}_1|} \right\rangle \\ &\geq \frac{|\mathbf{O}_i|}{|\mathbf{O}_i|+|\mathbf{n}_0|}. \end{aligned} \tag{8}$$

Here, $\mathbf{n}_0(t) = w(t)\mathbf{n}(t-t_0)$ and $\mathbf{n}_1(t) = w(t)\mathbf{n}(t-t_1)$. We have assumed that the noise is uncorrelated with the signal (i.e., $\langle \mathbf{O}_i, \mathbf{n}_1 \rangle = 0$) and is stationary (i.e., $|\mathbf{n}_0| = |\mathbf{n}_1|$). Intuitively, (8) makes sense: the peak of the correlation is slightly less than 1 due to noise. The worse the SNR, the further (8) will fall below 1.

When $i \neq j$ $R_{\mathbf{b}_0\mathbf{b}_1}(\tau)$ is the correlation of unrelated signals. From (8), we know that the correlation of blocks containing the same object is close to 1. If the blocks were orthogonal, it would be 0. For unrelated signals, we expect something in between. Let us define the worst case correlation between length E nonmatching signals as

$$\rho = \max_{\tau, i\neq j} R_{\mathbf{O}_i\mathbf{O}_j}(\tau). \tag{9}$$

We will discuss in Section III-B how to estimate ρ. Then, since our length L blocks are longer than E

$$\max_{\tau, i\neq j} R_{\mathbf{b}_0\mathbf{b}_1}(\tau) \leq \rho \cdot \frac{L}{E}. \tag{10}$$

Putting (8) and (10) into (7), we get that to reliably distinguish between a match (where $i = j$) and no match (where $i \neq j$), we require

$$\rho \cdot \frac{L}{E} < \frac{|\mathbf{O}_i|}{|\mathbf{O}_i|+|\mathbf{n}_0|}. \tag{11}$$

This establishes that a relation between the appropriate block length, the length of ROs, the SNR, and the worst-case time correlation of different objects. To take a concrete example, assume that we seek objects of average length 180 s in length on an audio broadcast where the reception SNR is 20 dB (i.e., $20\log_{10}(|\mathbf{O}_i|/|\mathbf{n}_0|) = 20$). We will estimate ρ in Section III-B, but for now assume $\rho = 0.1$. Then (11) gives that we can detect matches so long as $L < 9.095E$. To make detection reliable and robust, we might build in a factor of two margin for error. In other words choosing a block length of less than 4.5 times the length of the average RO will give reliable detection. Thus, a very simple routine for determining equality of segments is:

- $\mathrm{ApproxEqual}(\mathbf{b}_i, \mathbf{b}_j)$;
- if $\max_\tau R_{\mathbf{b}_i\mathbf{b}_j}(\tau) > 0.5$ ApproxEqual(i,j)==true else false.

We have chosen the threshold to be $T = 0.5$ here. We defer until Section III-A a more systematic choice.

C. Searching for Unknown Repeating Objects

If we sought only a single *known* object it would be an easy matter to spot recurrences: we would just constantly compare the known object with every incoming block. Seeking K *known* objects (much as is done in [17] and [16]) 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 at time t_p to break the stream into blocks $\mathbf{b}_i(t) = w(t) \cdot \mathbf{r}(t - iL/2), i = 0, 1, 2\cdots, \lfloor 2t_p/L \rfloor$; that is we have blocks of length L with overlap of $L/2$ stretching into the past. Clearly, any RO of length $L/2$ or less will be completely contained in one of the blocks. For example, if an RO begins at t_x, then it is completely contained in $\mathbf{b}_k(t)$, where $k = \lfloor 2t_x/L \rfloor$ and $b_k(t) = w(t) \cdot \mathbf{r}(t - t_k)$.

Allowing the number of blocks we compare to grow without bound (which happens as $j \to \infty$) is impractical, since this would require both infinite computation and that an infinite stream be stored. Instead, we search a finite distance of length NL into the past. This gives the following simple algorithm which searches blocks in the range $t_p - L/2, \cdots, t_p - NL$.

```
1. searchBuffer(b_j)
2. For i = j − 1 to j − 2N in steps of −1. If
   (approxEqual(b_i, b_j) == true) {found := true;
   goto 3}.
3. End.
```

The complexity of the algorithm is determined by the $2N$ calls to ApproxEqual(). So clearly, a buffer of duration NL can be searched in realtime only if $2N$ calls to ApproxEqual() can be computed in time $L/2$. To consider a concrete example, suppose we have a finite amount of computing power available which is capable of performing five calls to ApproxEqual() per second, and our blocks are $L/2 = 90$ s apart. The longest buffer we could search would then be approximately 11 hours in duration. This might be a problem in that ROs that repeat with period greater than 11 hours might never be found. We next show that so long as the stream is constructed by choosing objects at random, as modeled in Section II-A, even ROs that repeat with average period larger than NL will be found eventually.

D. Analysis of Probability of Objects Repeating in the Buffer

For an object to be found by searchBuffer(), it must appear twice in the length NL buffer. Since objects have average duration E, the average time between repeat copies of object i will be $E/(rp_i)$. So all objects for which $E/(rp_i) < NL$ 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 $M = \lfloor NL \cdot r/E \rfloor$ ROs 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 $t = NL$, when the buffer first fills,

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

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

$$\begin{aligned} &Pr\left(i, t + \frac{E}{r}, \text{found}\right) - Pr(i, t, \text{found}) \\ &\quad = [1 - Pr(i, t, \text{found})] \\ &\qquad \cdot Prob\{\text{1 copy of i in buffer} | \text{Not 2 copies of i}\} \\ &\qquad \cdot Prob\{\text{Incoming object is i}\} \\ &\qquad \cdot Prob\{\text{Outgoing object is not i}\} \\ &\quad = [1 - Pr(i, t, \text{found})] \\ &\qquad \cdot \frac{\binom{M}{1} p_i (1-p_i)^{M-1}}{\binom{M}{1} p_i (1-p_i)^{M-1} + (1-p_i)^M} \cdot p_i \cdot \frac{M-1}{M}. \end{aligned}$$

In words, in the interval E/r it takes a new object to enter the buffer, the probability that object i has appeared at least twice increases by the probability that: the buffer previously contained only one copy, and the incoming object is the ith, and the outgoing object is not the ith. In Fig. 1, we graph how $Pr(i, t, \text{found})$ evolves for various scenarios. Using $NL = 1440$ min (i.e., one day) and $r = 0.9$ and $E = 3.5$ min 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 h, one day, and three days, respectively. As can be seen from Fig. 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.

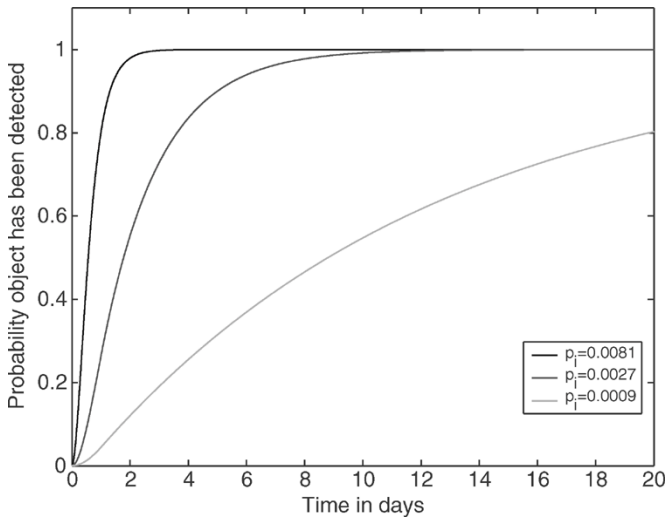

Fig. 1. Probability that two or more copies simultaneously appear in a one day buffer for objects with various probabilities p_i. Observe that even an object which repeats on average every $E/rp_i = 3$ days has a 50% probability of being detected after 10 days.

E. Summary

We saw in Section II-B that determining that two segments of a media stream are the same is simple. Applying this approach by brute force to every segment allows us to find all the objects that appear more than once in a buffer. For very common objects it is clear that they are likely to appear twice or more in a sufficiently long buffer. In Section II-D, we saw that for less common objects time is on our side also: they have increasing probability of appearing at least twice as the buffer evolves.

However, this approach is impractical for a number of reasons.

- Calculating time correlations of audio or video segments is expensive.
- The longest buffer it is feasible to search in realtime may not be long enough to contain many ROs.
- Long buffers of media streams consume much memory (one day of uncompressed 44.1-kHz stereo audio consuming 15.2 GB and one day of NTSC quality video consuming about 70 GB).

The main approaches to improve the efficiency of this algorithm are as follows.

1) Reduce the complexity of each call to ApproxEqual()
2) Reduce the number of calls to ApproxEqual().

The sorting and searching literature [3] will be of some assistance with 2), and we will examine this in Section III-D. Some techniques that are particular to media can help with 1), and we will examine them in Section III-A. We will also show that we can eliminate the need to buffer long stretches of the full-rate stream.

III. Reducing the Complexity of Determining Repeats

Clearly, as we have just seen, determining that two segments of a stream are the same can be done by computing time cor-

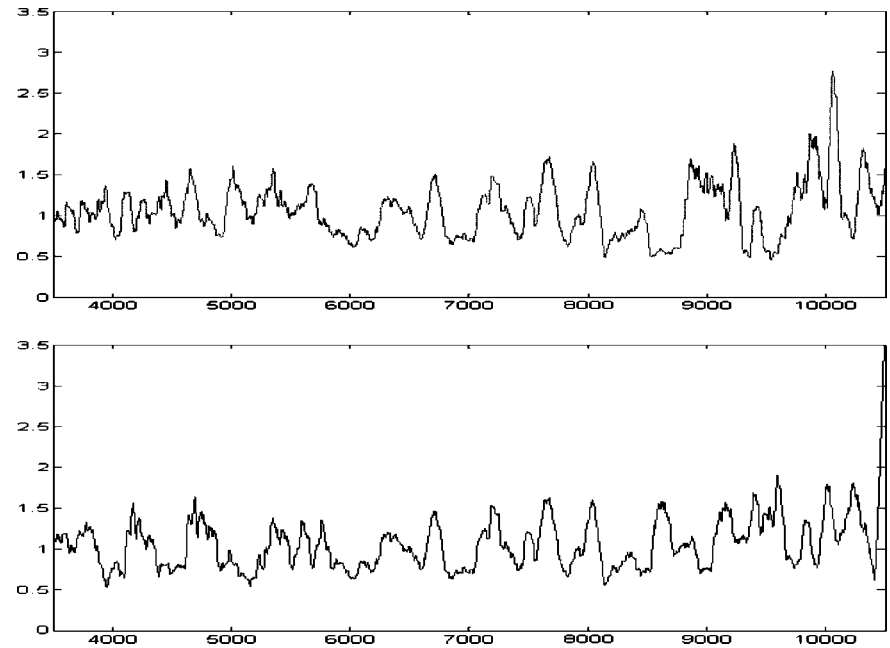

Fig. 2. Two different copies of the same object captured at different times from an FM radio broadcast. The graph is $\Phi_6(\mathbf{b}_i(t))$ versus time sampled at 11 samples/s. The center portions of the two copies approximately coincide, while the beginning and end portions do not. The similarity of the segments is evident, even though we are using a very low-dimension representation.

relations. However, calculating time correlations on video data, or even audio at 44.1 k samples per second is computationally expensive. Both video and audio contain much redundancy that does not help decide similarity. However, results from various other fields indicate that objects can be deemed the same by examining only low-dimension versions rather than the objects themselves. The Pattern Recognition literature [5] has numerous approaches to dimension reduction of large data sets. The database community has long observed that it suffices to compare the hashes of records to determine equality [4], [7].

A. Low-Dimension Representations of Audio

Since an object in a media streams repeats only if both audio and video repeat, 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. Once a match has been found in the audio, it is simple to verify whether or not the video also repeats. Recall that verifying a repeat is much less expensive than searching for repeats. Hence, there is no need to involve the video data in the search for ROs, though of course it will be involved in the verification.

We saw in (11) that reliable determination of whether two blocks $\mathbf{b}_0(t)$ and $\mathbf{b}_1(t)$ contained a matching object depended on the SNR, the lengths of the block and the ROs (L and E respectively), and the worst-case correlation between non-matching objects ρ. Instead of calculating the correlation between full-rate audio blocks, let us examine using a low-dimension representation. Suppose we split $\mathbf{b}_0(t)$ and $\mathbf{b}_1(t)$ into their projections onto lower dimensional subspaces (for example, particular frequency bands). Thus

$$\mathbf{b}_0(t) = \sum_k \Phi_k\left(\mathbf{b}_0(t)\right). \tag{12}$$

Here, $\Phi_k(\mathbf{b}_0(t))$ is the projection of $\mathbf{b}_0(t)$ onto the kth subspace (we'll examine shortly choosing good subspace decompositions). Of course $\Phi_k(\mathbf{b}_0(t))$ is itself a representation of the stream in the vicinity of t_0, but it can often be sampled at a much lower rate since the subspace has lower dimension than the original space. For example, if the subspaces are frequency bands, a signal that lives in a band a few hundred Hertz wide obviously need not be sampled at the same rate as the 44.1-kHz original.

Now repeat the analysis on the correlations in Section II-B, but now use these projections $\Phi_k(\mathbf{b}_0)$, $\Phi_k(\mathbf{b}_1)$. The idea is that we may be able to determine a match by examining correlations of only a single low-dimension projection of the audio, rather than the full-rate audio. Substituting $\Phi_k(\mathbf{b}_0)$ for $\mathbf{b}_0$ and $\Phi_k(\mathbf{n}_0)$ for $\mathbf{n}_0$ in the calculations of Section II-B, we get that the correlations of $R_{\Phi_k(\mathbf{b}_0)\Phi_k(\mathbf{b}_1)}(\tau)$ can be used so long as

$$\rho_k \cdot \frac{L}{E} < \frac{|\Phi_k\left(\mathbf{O}_i(t)\right)|}{|\Phi_k\left(\mathbf{O}_i(t)\right)| + |\Phi_k(\mathbf{n}_0)|} \tag{13}$$

where $\rho_k = \max\tau,\ i \neq jR_{\Phi_k(\mathbf{O}_i)\Phi_k(\mathbf{O}_j)}(\tau)$. Compare with (11). If the noise is uniformly spaced across all projections (as will be the case if it is white), we have $|\Phi_k(\mathbf{O}_i)|/|\Phi_k(\mathbf{n}_0)| \approx |\mathbf{O}_i|/|\mathbf{n}_0|$, and the righthand side of (13) is almost unchanged from that of (11). Thus, if we can find a projection $\Phi_k()$ such that ρ_k is small, reliable determination of equality is possible using correlations of this projection instead of correlation of the full-rate stream. We replace the examination of $R_{\mathbf{b}_i\mathbf{b}_j}(\tau)$ in routine ApproxEqual() in Section II-B with examination of $R_{\Phi_k(\mathbf{b}_i)\Phi_k(\mathbf{b}_j)}$. We use as threshold a point halfway between the worst-case auto and cross time correlations

$$= \frac{1}{2} \cdot \left(\rho_k \cdot \frac{L}{E} + \frac{|\Phi_k\left(\mathbf{O}_i(t)\right)|}{|\Phi_k\left(\mathbf{O}_i(t)\right)| + |\Phi_k(\mathbf{n}_0)|}\right). \tag{14}$$

TABLE II
WORST CASE CORRELATIONS BETWEEN BLOCKS WITH NO MATCHING OBJECTS. CORRELATIONS FOR THE WHOLE STREAM AND THE BARK BANDS $\Phi_4, \cdots \Phi_8$ ARE SHOWN

ρ	0.10565
ρ_4	0.2786
ρ_5	0.25943
ρ_6	0.23565
ρ_7	0.2646
ρ_8	0.3002

A commonly used tool for audio analysis is to split the audio into critical bands [13] (sometimes known as Bark bands). These split 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, 22 200} 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 each bark band, lowpass filter it, and then down sample. We call these waveforms $\Phi_k()$ for $k = 0, 1, 2, \cdots 24$. The sampling rate appropriate for each of these bands differs with frequency width, but we experimentally examined bands $\Phi_4, \Phi_5, \cdots, \Phi_8$ sampled at a rate of 11 samples per second. By way of example in Fig. 2 we show $\Phi_6(\mathbf{b}_0(t))$ and $\Phi_6(\mathbf{b}_1(t))$ for two sections of audio. Each segment represents slightly greater than 10 min 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.

B. Results: Evaluating $\Phi_6()$ as an Alternative to Full Rate

1) Comparing the Detection Rate of $\Phi_6()$ With the Full-Rate Stream: From (13), we know that the ability of $\Phi_k()$ to determine whether two blocks $\mathbf{b}_0(t)$ and $\mathbf{b}_1(t)$ contain the same object depends critically on ρ_k. To evaluate this, we performed the following experiment. We choose 100 blocks $\mathbf{b}_i(t)$ for $i = 0, 1, \cdots 99$ each 360 s long at random positions in an audio stream. The blocks form a representative collection of segments, meaning that there are songs, commercials, and speech in roughly the proportions that they contribute to the makeup of the stream. None of the blocks contain the same object. Calculating $\Phi_k(\mathbf{b}_i(t))$ for each of k =4, 5, 6, 7, 8, we then evaluated

$$\rho_k = \max_{\tau, i \neq j} R_{\Phi_k(\mathbf{b}_i)\Phi_k(\mathbf{b}_j)}(\tau). \tag{15}$$

We also estimated ρ, as defined in (9), using the same collection of segments. The results are shown in Table II. As can be seen $\rho_k > \rho$ in all cases. Thus, time correlations of any of the $\Phi_k()$ is slightly less good than using the full signal. However, from (13), we see that any of them still allows for reliable discrimination. For example, if $|\Phi_k(\mathbf{O}_i(t))|/(|\Phi_k(\mathbf{O}_i(t))| + |\Phi_k(\mathbf{n}_0)|) > 0.9$, i.e., the SNR in subspace k is an order or magnitude worse than that of the signal, the worst of the $\Phi_k()$ examined still allow reliable detection so long as $L \leq 3.2E$. On this basis, we selected $\Phi_6()$ as the projection to use in our work, having the lowest measured value of ρ_k. We should emphasize that we make no claims of optimality for $\Phi_6()$. We point out many other low-dimension representations that have been explored [16], [17], [20], [24] and several of these might serve as well or better. For determining equality of multimedia objects the Color Coherence Vector used (for video) in [17], [19] the fingerprint algorithms used in [2], [11], the audio measures of [20], [24] and the similarity measures of [16] are among functions that could be used as alternatives to the Bark spectra that we will use. In fact, [17] explicitly points out that any representation which is tolerant of channel deformations, has low dimension, and discriminates well between different objects will suffice. We choose $\Phi_6()$ as an example that has all of these properties. Since the broadcast channel gives a faithful reproduction of the signal it follow that $\Phi_6()$ is tolerant of channel deformations (since it is a subspace). That $\Phi_6()$ has low dimension follows from the fact that it is the low pass filtered version of the signal projection on a narrow frequency band; we verified that it discriminates between different objects in Table II.

2) False Positive Rate Using $\Phi_6()$: To test the hypothesis that $\Phi_6()$ 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 $\Phi_6()$ for the entire stream. We randomly selected 500 360 s segments; some of the segments were voice, some music, some a mixture of voice and music. We then formed the time correlation of each segment with every 6-min segment from the entire stream. When the time 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 time correlations of $\Phi_6()$ is very low.

3) False Negative Rate Using $\Phi_6()$: Determining the false negative rate is somewhat harder, since labeled multimedia streams are not readily available. Most broadcast radio stations, for example, do not publish playlists. To accomplish this we hand parsed a 48-h section of the stream; i.e., hand labeled every RO greater than 2 min in length. There were 282 such objects. We randomly selected 50 objects from the first 24 h of labeled stream, and for each of them: time correlated a 6-min segment of $\Phi_6()$ centered on the object with the entire second 24 h of labeled stream. No false negatives were found in the labeled stream. That is, in every case where our labeled stream showed that an object repeated, our algorithm found it; none were missed. We describe a further examination of the efficacy of the system in Section V.

Thus, we find that $\Phi_6()$ is a suitable low-dimension representation of the signal. In going from 44.1 k samples/s to 11 samples/s, we have achieved a 4000-fold reduction in the data rate, with no meaningful loss in recognition ability.

C. Identifying the Boundaries of Found Objects

We have seen that ROs can be identified in a number of ways. Time correlations, Audio fingerprints [2], [11], or Color Coherence Vectors [17], [19] are among the approaches that work.

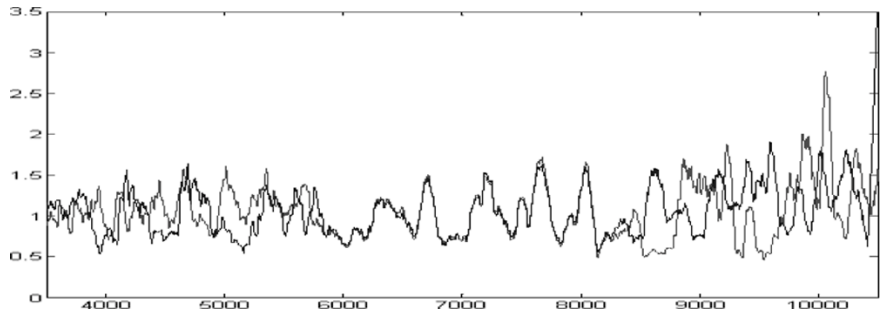

Fig. 3. Example indicating how the boundaries of an object can be calculated once two or more copies have been found. The graph is $\Phi_6(\mathbf{b}_i(t))$ versus time sampled at 11 samples/s. Overlay of the first and second instance of the object once they have been aligned. The points where the two streams diverge at the beginning and end are the endpoints of the object.

Any of these methods will determine that the data in the stream at locations t_0 and t_1 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 it is 10 s or 3 min long before any action can be carried out. Here we reach an important distinction between systems where the ROs are drawn from a library of known objects and those where the objects are unknown, which is the case we are interested in.

- ROs from finite known library: endpoints determined from metadata (e.g., the library contains length information for all ROs) or inferred (e.g., commercials are known to be 30 s or 15 s long).
- ROs not known: endpoints must be determined from the stream.

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 fact, in Fig. 2, we saw that $\Phi_6()$ showed considerable visual similarity between two different occurrences of the same object. In Fig. 3, we align, normalize and overlay these two segments. 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 8300 and thus (since $\Phi_6()$ is sampled at 11 samples/s) the object is approximately 227 s in duration.

Assuming that we have determined that the streams at t_0 and t_1 are approximately the same, we examine a windowed section of $\Phi_6()$ centered at each of these locations

$$\Phi_6\left(\mathbf{b}_0(t)\right) = \left[w(t)\cdot\Phi_6\left(\mathbf{r}(t-t_0)\right)\right]$$
$$\Phi_6\left(\mathbf{b}_1(t)\right) = \left[w(t)\cdot\Phi_6\left(\mathbf{r}(t-t_1)\right)\right].$$

We can align the two sections of stream by calculating $\tau_x = \tau_0 - \tau_1$ where τ_0 and τ_1 are the locations at which $R_{\Phi_6(\mathbf{b}_0')\Phi_6(\mathbf{b}_0')}(\tau)$ and $R_{\Phi_6(\mathbf{b}_0')\Phi_6(\mathbf{b}_1')}(\tau)$ are maximum, i.e., the peaks of the auto and time correlations, respectively. We set

$$\Phi_6\left(\mathbf{b}_0'(t)\right) = \left[w(t)\cdot\Phi_6\left(\mathbf{r}\left(t-t_0+\frac{\tau_x}{2}\right)\right)\right]$$

and

$$\Phi_6\left(\mathbf{b}_1'(t)\right) = \left[w(t)\cdot\Phi_6\left(\mathbf{r}\left(t-t_1-\frac{\tau_x}{2}\right)\right)\right].$$

Since there is a match, both $\Phi_6(\mathbf{b}_0'(t))$ and $\Phi_6(\mathbf{b}_1'(t))$ should now have the form given by (5)

$$\Phi_6\left(\mathbf{b}_0'(t)\right) = \left[\Phi_6\left(\mathbf{a}(t)\right)\left|\Phi_6\left(\mathbf{O}_i(t)\right)\right|\Phi_6\left(\mathbf{b}(t)\right)\right]$$
$$\Phi\left(\mathbf{b}_1'(t)\right) = \left[\Phi_6\left(\mathbf{c}(t)\right)\left|\Phi_6\left(\mathbf{O}_i(t)\right)\right|\Phi_6\left(\mathbf{d}(t)\right)\right]$$

where $\mathbf{a}(t)$, $\mathbf{c}(t)$ and $\mathbf{b}(t)$, $\mathbf{d}(t)$ are the parts of the stream that precede and follow $\mathbf{O}_i(t)$ at the two locations in the stream where we have determined it repeats.

These two stream segments can now be compared directly. This has been done in Fig. 3. The boundaries of the object $\mathbf{O}_i(t)$ can be estimated by, for example, thresholding the accumulated difference between the waveforms as one works out from the center. Once the normalization has been properly performed many schemes work well. A simple approach is shown next.

1) $\mathrm{getBoundaries}(\Phi_6(\mathbf{b}_0'), \Phi_6(\mathbf{b}_1'))$.
2) Get average absolute sample difference $a := (1/L)\sum_{t=0}^{L-1}|\Phi_6(\mathbf{b}_0'(t)) - \Phi_6(\mathbf{b}_1'(L-t))|$.
3) $t := L/2$.
4) if $(|\Phi_6(\mathbf{b}_0'(t)) - \Phi_6(\mathbf{b}_1'(t))| > a * 0.2)$ Goto 6.
5) $t := t-1$; Goto 4.
6) $t_{start} := t$; $t := L/2$.
7) if $(|\Phi_6(\mathbf{b}_0'(t)) - \Phi_6(\mathbf{b}_1'(t))| > a * 0.2)$ Goto 9.
8) $t := t+1$; Goto 7.
9) $t_{end} := t$.

We give this simplistic algorithm to illustrate the method. Greater robustness can be achieved by allowing the absolute difference to exceed a threshold for some time before terminating. Also, this algorithm works from the center $L/2$ out toward the endpoints. It is worthwhile to also work from the outside (i.e., $t = 0$ toward $L/2$ for t_{start} and from L toward $L/2$ for t_{end}) in, and combine the answers.

1) Results: To test the accuracy of our segmentation we randomly selected 100 of the ROs from the hand-labeled 48-h stream referred to in Section III-B. Objects were longer than 180 s on average. We exhaustively searched for all matches in the following 48 h. 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[1] 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 six of the 88 cases was the difference greater than 2 s.

Thus, the endpoint determination procedure based on examining $\Phi_6()$ data alone allows reasonably accurate segmentation of objects. *This implies that archiving large stretches of the original media stream is not necessary to accurately find and extract objects*. We saw, in Section III-A, that whether two segments of a stream matched could be determined by examining a low-dimension version such as $\Phi_6()$. Now we see that segmenting objects can also be accomplished using a low-dimension version.

Simple though it is, the endpoint determination is key to our ability to determine the underlying structure of a media stream. We already saw, in Section III-A, that finding ROs was possible in real time. Determining the endpoints makes it possible to reliably determine the duration without needing a pre-existing database, or needing to make assumptions of the nature of the objects, or the semantics of the stream. Thus, for the first time, we can decompose a repetitive stream into its component objects. Further, the fact that the endpoint detection can be done reliably on a low-dimension version of the signal means that we do not have to buffer large amounts of full-rate data. For example, an object that recurs only twice in a week-long video stream can be identified and accurately segmented based on a buffer of $\Phi_6()$, which consumes only 26 MB. This hugely simplifies the requirements of the system we will use to extract the underlying structure in our media streams; the architecture of such a system is shown in Fig. 5.

D. Increasing the Efficiency of the Search for Repeating Objects

The search strategy we introduced in Section II-C was essentially brute force: break a buffer of the stream into blocks and compare the current block with all past blocks in the buffer. We've seen (in Section III-A) how to improve the efficiency of the comparisons, and (in Section III-C) how to extract the objects once found. Recall, from Section II-C, that for fixed computational resources our algorithm could search a finite distance into the past. The complexity clearly grows linearly with the length of the buffer we wish to search. We now show that by exploiting the repetitive nature of the stream we can improve the search.

The key observation is that once an object is found, and its endpoints identified, that segment of the buffer does not have to be searched again, and the object can be added to a list of known objects. Every time we find a RO, we can shorten the length of the buffer that remains to be searched or, extend the distance into the past that we can search. In fact, once we find a RO, we add it to a library of 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 RO will be in the library and will be identified from there without having to search the buffer. As 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 ROs can be ordered by frequency of repetition, so that most common object are checked first; this further improves the efficiency.

[1]Note: the endpoint is regarded as the point where the two copies of the object diverge. For example, a 10-s news clip which is itself part of a larger 20-s clip will count as a RO. The RO will be considered to be 10 s long, however.

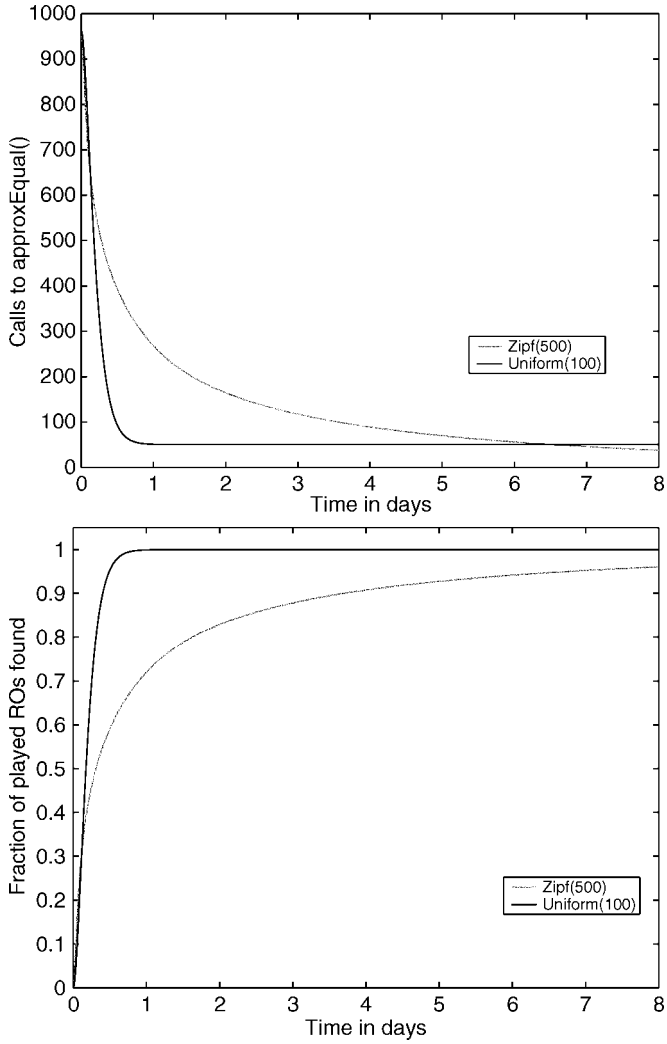

Fig. 4. Improving the efficiency of searching for unknown ROs. A buffer of length one days is assumed, for the two distributions shown. (a) Number of calls to ApproxEqual(). Observe that the number of calls begins as the cost of searching the buffer, but rapidly converges to the cost of searching the library. (b) Fraction of the ROs played in a stream found. Observe that after two days almost every RO being played has already been found.

This improves the efficiency of the search by an amount related to how repetitive the stream is. The most common objects are found first, and also reduce the remaining length most. This gives rise to the following simple variation on our first algorithm searchBuffer().

1. $\text{searchLibraryAndBuffer}(\Phi_6(\mathbf{b}_j))$.
2. $K := 0$.
3. For $k = 0$ to $K - 1$: if $\text{approxEqual}(\Phi_6(\mathbf{b}_j), \Phi_6(\mathbf{O}_k))$ {$\text{getBoundaries}(\Phi_6(\mathbf{b}_j), \Phi_6(\mathbf{O}_k))$; found := true; goto 5.}
4. For $i = j-1$ to $j-2N$ steps of -1: If $(\text{approxEqual}(\Phi_6(\mathbf{b}_i), \Phi_6(\mathbf{b}_j)) == \text{true})${

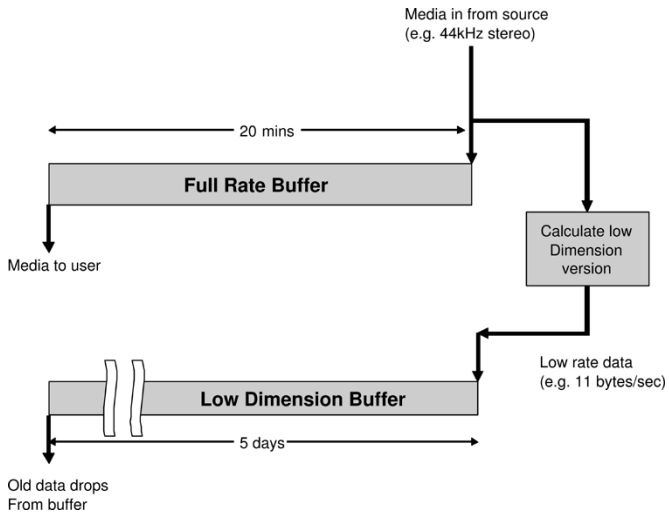

Fig. 5. Block diagram of a system to automatically identify and extract ROs from a media stream. The media stream enters a full-rate buffer and is delayed before playing. A low-dimension version of the stream is sent to a low-dimension buffer. This buffer is constantly searched for ROs using routine searchLibraryAndBuffer(). Found objects are stored in an RO library. Boundaries calculated by comparing two copies in the low-dimension buffer are used to generate the boundaries in the full-rate buffer. Deletion and copying then become simple. The full-rate buffer needs only to be longer than the longest expected RO. The low-dimension buffer needs to span an interval long enough to allow objects to repeat. This allows objects that occur days apart to be found without having to buffer the full-rate stream.

getBoundaries($\Phi_6(\mathbf{b}_j), \Phi_6(\mathbf{b}_i)$); found := true; $K := K + 1$; goto 5.}
5. End.

The first loop (step 3) searches the library of found objects, while step 4 searches the buffer. The major difference is that once an object is found we add it to the library, and the library is always searched first.

We now analyze the improvement of this algorithm over the brute force approach in Section II-C. If the current block contains the ith 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 $K+(2N-1)$ otherwise. Actually, if we sort the library of objects by frequency of occurrence and search it in that order, the number of calls will on average be less. The average number of objects in the library is $\sum_{i=0}^{K-1} Pr(i,t,\text{found})$, so it should take

$$\frac{\sum_{i=0}^{K-1} Pr(i,t,\text{found})}{K} \cdot i$$

calls to approxEqual() on average to find the ith object in the library (assuming that the p_i are sorted in descending order). Thus, the average number of calls to approxEqual() is less than

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

Thus, the average number of comparisons will be upper bounded by

$$\sum_{i=0}^{K-1} p_i \cdot \left(Pr(i,t,\text{found}) \cdot \left(\frac{\sum_{i=0}^{K-1} Pr(i,t,\text{found})}{K} \cdot i \right) + [1 - Pr(i,t,\text{found})] \cdot \left(\sum_{i=0}^{K-1} Pr(i,t,\text{found}) + 2N - 1 \right) \right). \quad (16)$$

As shown in Fig. 1, $Pr(i,t,\text{found})$ tends to one for increasing time; hence, the average number of comparisons tends to decrease to the number required to search the library. 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. 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 NL. Now we have shown that, using searchLibraryAndBuffer(), the complexity is initially proportional to N, but quickly converges to the cost of searching the library. Thus, for repetitive streams (i.e., those for which r is close to 1), the complexity of identifying *unknown* ROs converges to the complexity of identifying a collection of *known* ROs.

We take the example of a stream with composed of $K = 100$ objects drawn from a uniform distribution and $K = 500$ from a Zipf distribution. This means that the nth most common object occurs with a frequency inversely proportional to n (a distribution that occurs naturally in a wide variety of contexts) [3]. We choose $r = 0.9$ and the average object is of length $E = 210\,\text{s}$, and a buffer of length NL of four days. In Fig. 4(a), we show how the average number of calls to approxEqual() as predicted by (16) evolves over time. As can be seen, the number is high as the entire buffer has to be searched. It drops as the RO library fills, however, and approaches the size of the library when almost all objects are found from the library and the buffer seldom needs to be searched. In the case of the uniform distribution, the number of calls approaches $K/2$ since this is the average required to search the library once all objects have been found. In Fig. 4(b), we show the fraction of the K objects that comprise the stream that have been found as a function of time. In this example, using a buffer length of one day, most objects have been found within two days.

E. Summary

In Section II, we saw that searching for unknown ROs in a stream was possible but demanding because of the following.

- Complexity of each call to approxEqual().
- Number of calls to approxEqual() required to search a buffer for unknown objects.
- Memory required to buffer long stretches of a media stream.

We have now solved each of these problems, making the identification of ROs not merely possible, but within reach with modest compute and storage resources.

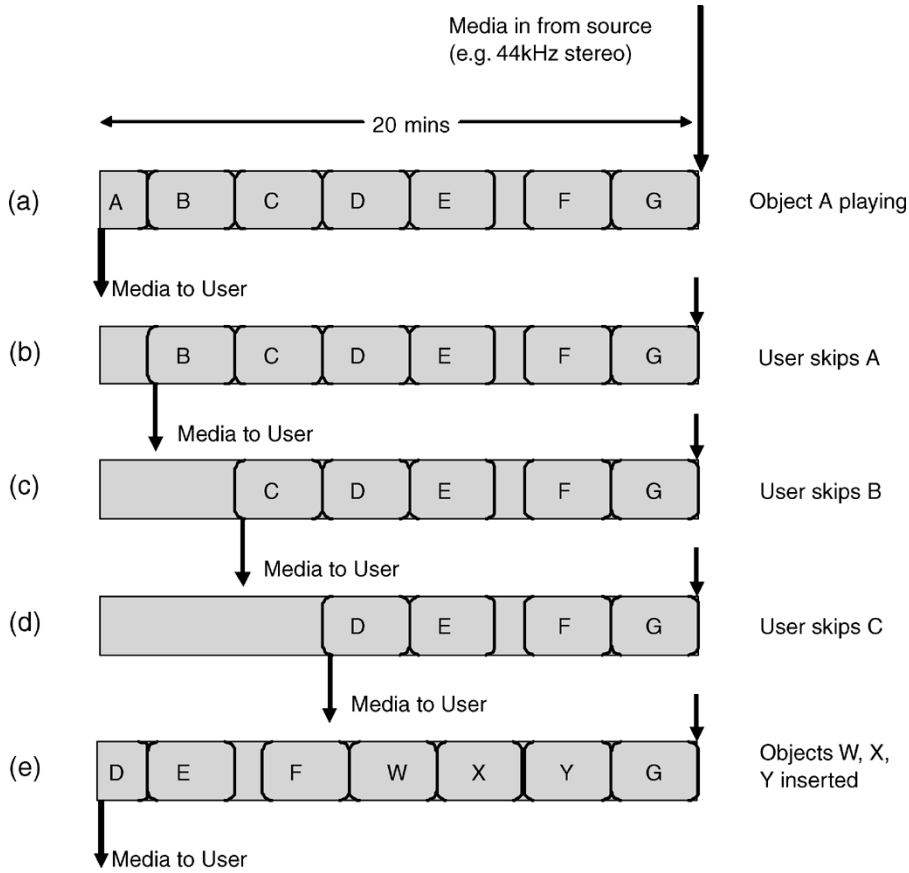

Fig. 6. Evolution of the full-rate buffer. The ROs are found by searching the low-dimension buffer (not shown). (a) Objects A-G have been found and their boundaries marked. The user is currently viewing object A. (b) User skips object A. The read pointer advances to the next object boundary. (c) User skips object B. The read pointer advances to the next object boundary. (d) User skips object C. The read pointer advances to the next object boundary. Observe that the read and write edges of the buffer are getting closer. (e) System inserts objects W, X, and Y between F and G.

IV. Architecture of a System to Allow Automatic Identification and Segmentation of Repeating Objects

Having abstracted the blocks that:

- determine whether two segments are the same object (Section III-A);
- identify the endpoints of the object (Section III-C);

we are now in a position to illustrate the architecture of a scheme for identifying and extracting repeats.

A block diagram of the architecture is shown in Fig. 5. An incoming media stream $\mathbf{r}(t)$ is fed into a full-rate buffer before playback, so that the incoming stream is delayed by 20 min before viewing. At the same time, a low-dimension version of the stream $\Phi_6(\mathbf{r}(t))$ is calculated and fed into a second (low dimension) buffer, which is five days long. The first buffer is full rate, meaning that it contains the full fidelity stream, e.g., 44.1-kHz stereo for an audio stream, or 50 frames per second for video. This buffer can be quite short, as we will use it only for performing operations on objects when they have been identified and before they are played. The second buffer, containing the low-dimension data, can represent a far longer stretch of the stream. This buffer will be used to determine when a segment in the stream is recurring (as covered in Sections III-A and III-D) and to identify the endpoints of the objects once found (as covered in Section III-C). That is, once two copies of an RO have been identified in the low-dimension buffer using searchLibraryAndBuffer() the boundaries are identified using getBoundaries(). One of the copies will have just entered the buffer, the other may be some distance in the past. Thus, the most recent copy will also be sitting in the full-rate buffer. We use the boundaries to delete, copy, or perform other action on it.

Again, to use the concrete example of 16-bit stereo audio, a full-rate 20-min buffer would consume $20 \cdot 60 \cdot 44.1e3 \cdot 2 \cdot 2 \approx 211$ MB if the audio were uncompressed. Using $\Phi_6()$ for the low-dimension representation, a week-long low-rate buffer would consume only 26 MB. Thus, neither of these presents any real burden for a modestly capable desktop computer, and can generally be accomodated in RAM. To use the concrete example of an video signal at NTSC quality, a 20-min buffer would consume approximately 2.4 GB uncompressed, while the low-dimension representation would take 26 MB, as for the audio.

A. Binding Actions to Objects

The architecture of Fig. 5 allows identification of objects as they repeat, and determination of their endpoints. Thus, before an object is played in the high rate buffer, we will know whether it is a RO, and will have identified its endpoints if it is. By providing a suitable User Interface (UI), we can allow users to specify actions that they wish to associate with objects as they recur. For example:

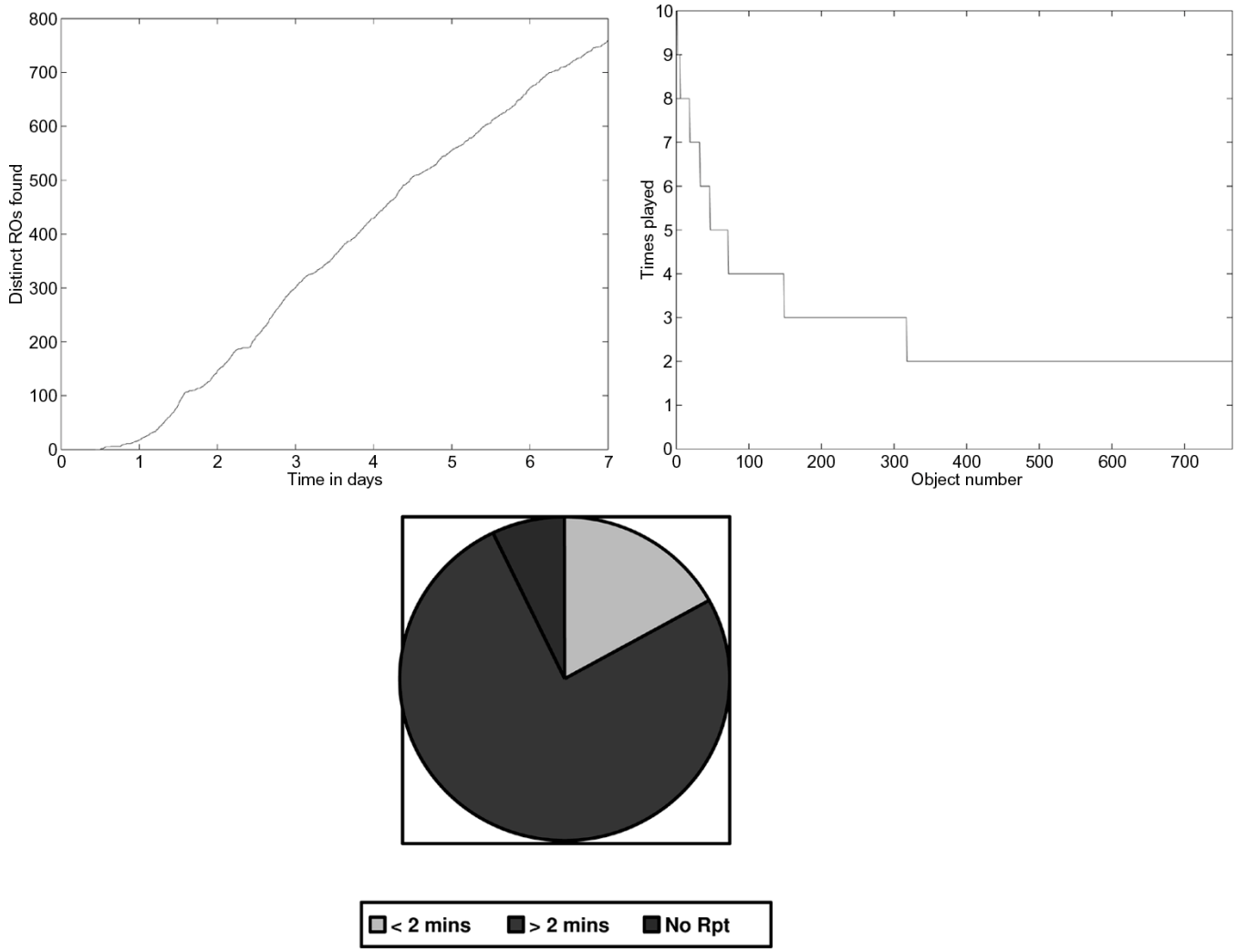

Fig. 7. ROs detected on a real broadcast stream. The Seattle pop radio station KYPT FM 96.5 analyzed for a period of seven days. (a) Number of unique ROs found as a function of time. For the first day few objects are found, since repeats do not occur. After the first day many objects are found initially, but the rate slows as fewer objects remain to be found. A total of 766 unique ROs greater than 2 min in length were 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. (c) Pie chart of the ROs greater and less than 2 min in length and nonrepeating parts of the stream.

- keep a copy of this object;
- skip this instance of this object;
- delete all future instances of this object.

Keeping a copy is easily accomplished: using the endpoint information derived from the low-dimension buffer, we determine the boundary points of the object in the full-rate buffer. Copying merely involves storing the appropriate section of the full-rate buffer to a file. Deleting involves advancing the read pointer of the buffer to the rightmost boundary of the object playing. An example of deletion and insertion on the full-rate buffer is shown in Fig. 6. Of course, every time an object is deleted, the distance between the read and write pointers of the buffer decreases. Deleting many objects will bring the read and write edges of the buffer close together. The edges can be pushed further apart by inserting objects in the buffer. These can be taken from a library or from objects identified and copied earlier in the stream. For example, a stream is made up of objects some of which the user elects to skip; as he does so, the buffer starts to get smaller, and hence the ability to continue deleting objects will be reduced if no action is taken. To prevent this, objects that the user has seen and not deleted are inserted to prevent the read and write edges of the buffer from getting too close. This makes it possible to listen to a stream indefinitely, yet while deleting undesired objects.

V. Results of Identifying Repeating Objects on Broadcast Streams

We implemented the algorithm above and ran on several broadcast streams. Using a Pentium III 750-MHz PC with 512 MB of RAM, we were able to search a buffer of five days in real time using only approximately 24% of CPU.

A. Examining Performance Against Ground Truth

In Section III-B we examined the efficacy of $\Phi_6()$, and had promising results. To do so, we had to hand label a stream to estimate the false negative rate of our detection algorithm. The reason was that finding multimedia streams with accurate labeling information (i.e., an accurate log of the contents of the

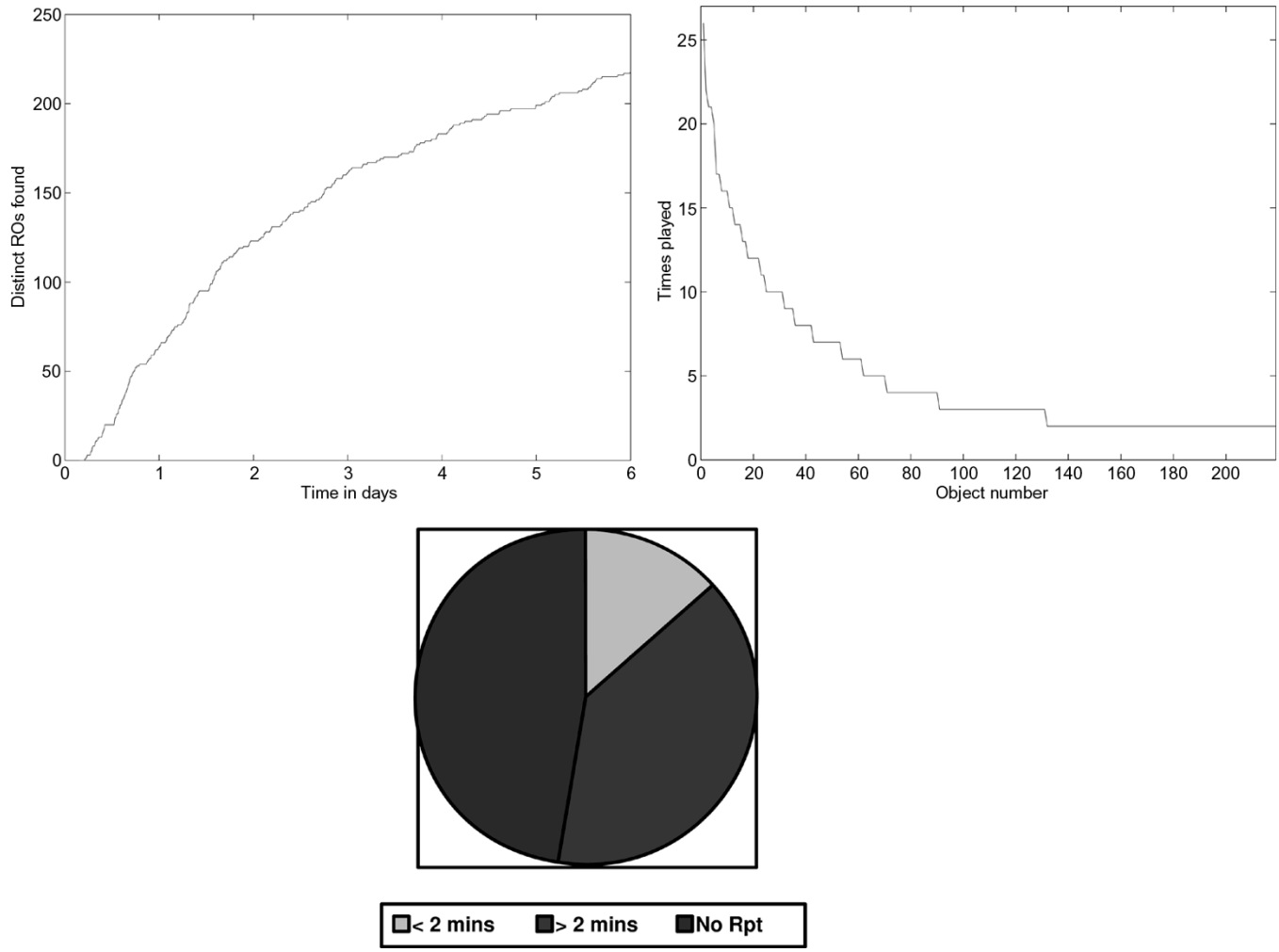

Fig. 8. ROs detected on a real broadcast stream. The Seattle pop radio station KNDD FM 107.7 analyzed for a period of six days. (a) Number of unique ROs found as a function of time. Many objects are found initially, but the rate slows as fewer objects remain to be found. A total of 219 unique ROs greater than 2 min in length were 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. (c) Pie chart of the ROs greater and less than 2 min in length and nonrepeating parts of the stream.

stream) is difficult. Most broadcast streams do not publish such labeling. While some internet radio stations publish playlist information, we wish to test how our system performs on an over the air broadcast stream (i.e., complete with the noise that that implies). One radio station that does publish labeling data is Seattle FM 90.3 KEXP. An FM receiver was connected to the line-in of the PC. We listened to this station for a period of two weeks using ARGOS. As listed in the logs, the station played a total of 2762 songs during that time, of which 2211 were distinct. That is the stream during this period could be considered to comprise objects $\mathbf{O}_0, \mathbf{O}_1 \cdots, \mathbf{O}_{2210}$. According to the log files, a total of 350 objects repeated during the two-week listening period. The ARGOS algorithm found and segmented a total of 344 objects. The maximum that the system could have found would have been 350, since objects that do not repeat would not be detected. This implies a false negative rate of less than 2%. There were no false positives. This is not surprising, since, even if the detection algorithm of Section III-A falsely registered a match, the boundary finding algorithm of Section III-C would find a zero-length object. Our algorithm had a sanity check to reject objects shorter than $L = 15$, since such objects cannot be reliably detected with this approach.

B. Results on Audio Streams

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.

Examples of the results listening to two different FM Radio stations are given in Figs. 7 and 8. In both cases, we used $L = 360$ s. In Fig. 7, we show the results of listening to a Seattle pop music radio station (FM 96.5 KYPT) for seven days. We plot the rate at which unique new objects were found, and the distribution of play frequencies. We also give a pie chart showing the portions of the stream that are made up of ROs greater than 2 min, less than 2 min, and nonrepeating content. Roughly speaking these might be taken to represent the portion of the streams that are songs, advertisements and talk respectively. Clearly, in Fig. 7(a), very few objects were found in the first day. This reflects the fact that objects will not be recognized until they repeat, and repeats do not happen

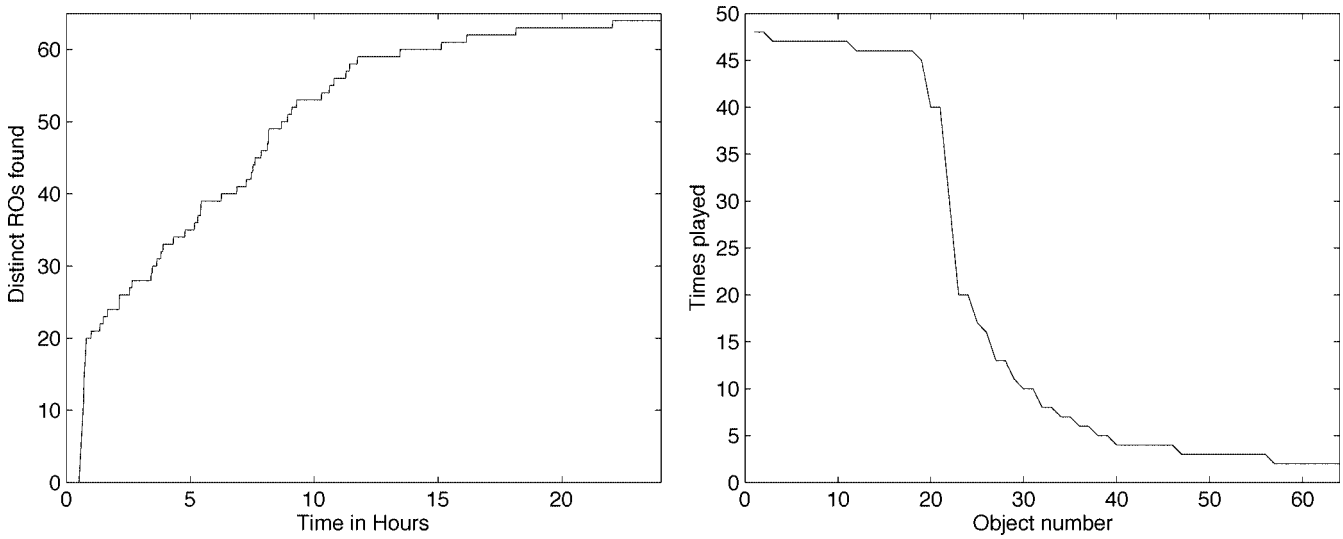

Fig. 9. ROs detected on a real broadcast stream. The cable channel CNN Headline News analyzed for a period of 24 h. (a) Number of unique ROs found as a function of time. For part of the first hour, few objects are found, since repeats do not occur. After this many objects are found initially, but the rate slows as fewer objects remain to be found. A total of 766 unique ROs greater than 20 s in length were 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.

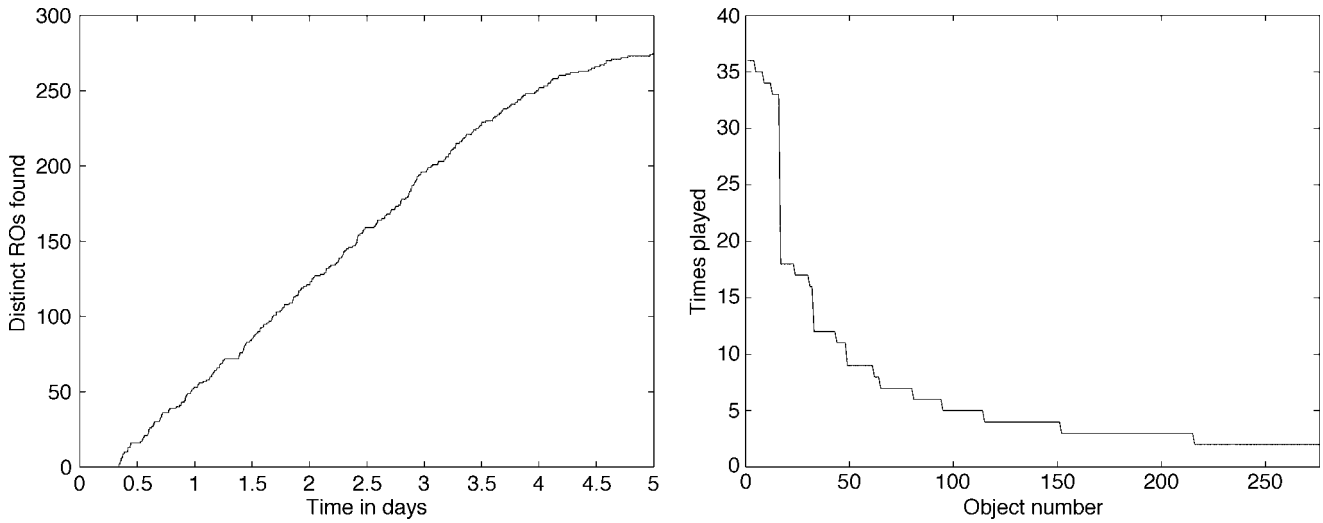

Fig. 10. ROs detected on a real broadcast stream. The cable channel MTV was listened to for a period of five days, played in a stream. A small number of popular objects are played very frequently and the stream is very repetitive. (a) Number of unique ROs found as a function of time. Many objects are found initially, but the rate slows as fewer objects remain to be found. A total of 276 unique ROs greater than 2 min in length were 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.

immediately. Once repeats start to occur (after one day) objects are found rapidly at first and then tail off, with a total of 766 unique objects of length 2 min or greater being found. Fig. 7(b) shows the play frequencies of the objects found; some objects played almost every day while others appeared only twice in the seven day period. In Fig. 8, we show the results from listening to FM 107.7 KNDD for a period of six days. Comparing the results from the two stations is interesting; clearly KYPT plays a somewhat larger collection of objects, while KNDD plays fewer objects (a total of 219 unique objects of length 2 min or greater were found) but plays some objects very often. The two pie charts [Figs. 7(c) and 8(c)] show the difference in makeup of the streams. On KYPT almost 75% of the stream consists of ROs of length 2 min or greater (presumably music), while only 7.2% contained no repeats (presumably talk or objects that occurred only once and hence were not detected as repeats). For KNDD, these fractions were 39% and 47%, respectively.

C. Results on Video Streams

A cable signal was connected to a tuner card on the PC. Our low-dimension representation of the audio portion of the signal was used to find repeats and detect endpoints of objects once found. The video portion was examined only to verify that the repeat also involved the visual portion of the signal. Those where video did not match were rejected as not being ROs following our definition.

Examples of the results listening to two different cable TV stations are given in Figs. 9 and 10. In Fig. 9, we show the results

of viewing "CNN Headline News" for 24 h. We plot the rate at which unique new objects were found, and the distribution of play frequencies. This is a very repetitive channel, consisting mostly of short news briefs and commercials. Typical ROs are 30 s in length. We used $L = 90$ s. The rate at which ROs are found, shown in Fig. 9(a), is very different from the FM radio examples. A large number of objects are found very quickly, since news items repeat twice an hour; after that new ROs consist only of relatively infrequent new stories, and commercials that have not yet been seen. The profile of repeat rates, shown in Fig. 9(b), shows that a small collection of ROs repeat many times (presumably the news stories), while a larger collection appears less frequently (presumably the commercials).

In Fig. 10, we show the results of viewing MTV, a musical video station for five days. Since the objects we were interested in have an average length of 180 s or so, we used $L = 360$ s. Here, the profile of ROs follows that of one of the FM stations more closely. In Fig. 10(a), we plot the rate at which unique ROs greater than 2 min in length arrive. For the first few hours, no ROs are found, but the rate at which they are found is almost linear before tailing off after about four days. In Fig. 10(b), the play frequency of the ROs is shown. Again, a combination of objects that play many times a day (the most common played 32 times in a five-day period) and less common objects (which played only twice) is evident.

VI. Conclusions

We have shown that identifying repeats in media streams that are days or even weeks apart is perfectly feasible for a consumer PC. Key to this process was the selection of a low-dimension representation that reduced the complexity of the search, exploiting the repetitive nature of the stream, and the use of two copies to identify the endpoints of the object. The main contributions are: we show how to efficiently search for unknown objects; we show that for repetitive streams the complexity converges to that of searching for known objects; we show how to determine the endpoints of objects; and we show that buffering large sections of the stream is not necessary to extract even objects that recur infrequently. A number of applications are advanced.

Acknowledgment

The author would like to thank C. Burges, P. Chou, P. England, D. Florencio, J. Platt, E. Renshaw, and J. Stokes for numerous discussions on the subject of this work. He also thanks all three anonymous reviewers for thorough comments that materially improved the paper.

References

[1] A. D. Bimbo, P. Pala, and L. Tanganelli, "Retrieval by content of commercials based on dynamics of color flows," in *Proc. ICME*, New York, 2000, pp. 479–482.

[2] C. J. C. Burges, J. C. Platt, and S. Jana, "Distortion descriminant analysis for audio fingerprinting," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 3, pp. 165–174, May 2003.

[3] D. E. Knuth, *The Art of Computer Programming*. Reading, MA: Addison-Wesley, 1973, vol. 3.

[4] K. Delaney, *Inside SQL Server 2000*. Redmond, WA: Microsoft Press, 2001.

[5] R. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: Wiley, 2000.

[6] C. Faloutsos, *Searching Multimedia Databases by Content*. Norwell, MA: Kluwer, 1996.

[7] H. Garcia-Molina, J. Ullman, and J. Widom, *Database System Implementation*. Englewood Cliffs, NJ: Prentice-Hall, 2000.

[8] A. Hampapur and R. Bolle, "Feature based indexing for media tracking," in *Proc. ICME*, New York, 2000.

[9] C. Herley, "Extracting repeats from streams," in *Proc. ICASSP*, Montreal, QC, Canada, May 2004.

[10] J.-L. Hsu, C.-C. Liu, and A. L. P. Chen, "Discovering nontrivial repeating patterns in music data," *IEEE Trans. Multimedia*, vol. 3, no. 3, pp. 311–325, Sep. 2001.

[11] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *Proc. Int. Conf. Music Information Retrieval*, Paris, France, Oct. 2002.

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

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

[14] H. Jiang, T. Lin, and H.-J. Zhang, "Video segmentation with the assistance of audio content analysis," in *Proc. ICME*, New York, 2000.

[15] J. Kang, J. F. Naughton, and S. D. Viglas, "Evaluating windo joins over unbounded streams," in *Proc. 28th Int. VLDB Conf.*, Hong Kong, China, 2002.

[16] K. Kashino, T. Kurozumi, and H. Murase, "A quick search method for audio and video signals based on histogram pruning," *IEEE Trans. Multimedia*, vol. 5, no. 3, pp. 348–357, Sep.. 2003.

[17] R. Lienhart, C. Kuhmuench, and W. Effelsberg, "On the detection and recognition of television commercials," in *Proc. Int. Conf. Multimedia Computing and Systems*, Jun. 1997, pp. 509–516.

[18] T. Muramoto and M. Sugiyama, "Visual and audio segmentation for video streams," in *Proc. ICME*, 2000, pp. 1547–1550.

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

[20] S. Pfeiffer, S. Fischer, and W. Effelsberg, "Automatic audio content analysis," in *Proc. ACM Multimedia Conf.*, 1996, pp. 21–30.

[21] S. L. Marple Jr., *Digital Spectral Analysis With Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[22] H. Sundaram and S.-F. Chang, "Video scene segmentation using video and audio features," in *Proc. ICME*, New York, 2000.

[23] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*. Englewood Cliffs, NJ: Prentice-Hall, 1990.

[24] E. Wold, T. Blum, D. Keislar, and J. Wheaton, "Content-based classification, search and retrieval of audio," *IEEE Multimedia*, vol. 3, no. 3, pp. 27–36, Fall 1996.

[25] M. Yeung, B.-L. Yeo, and B. Liu, "Extracting story units from long programs for video browsing and navigation," in *Proc. IEEE Conf. Multimedia Computing and Systems*, Jun. 1996, pp. 296–305.

[26] R. Ragno, C. J. C. Burges, and C. Herley, "Inferring Similarity Between Music Objects with Application to Playlist Generation," in *ACM MIR 2005*, Singapore, pp. 73–80.

Cormac Herley (M'93) was born in Cork, Ireland. He received the B.E. (Elect.) degree from the National University of Ireland in 1985, the M.S.E.E. degree from the Georgia Institute of Technology, Atlanta, in 1987, and the Ph.D. degree from Columbia University, New York, NY, in 1993.

He has worked for Kay Elemetrics Corp, Pine Brook, NJ, AT&T Bell Laboratories, Murray Hill, NJ, Hewlett-Packard Labs, Palo Alto, CA, and has been at Microsoft Research, Redmond, WA, since 1999. His research interests include theoretical and applied problems in signal processing, multimedia, machine learning, and statistics.