Real-Time Imaging **2**, 35–44 (1996)

Using Content-Based Search to Download Digital Video into a Client Station

In this paper we examine a content-based method to download/record digital video from networks to client stations and home VCRs. The method examined is an alternative to the conventional time-based method used for recording analogue video. Various approaches to probing the video content and to triggering the VCR operations are considered, including frame signature matching, program bar-code matching, preloaded pattern search, and annotation signal search in a hypermedia environment. A concept of go-ahead-downloading is introduced and system capacity of the downloading device is evaluated. Some experimental studies are conducted on preloaded pattern search to provide more insights into this approach. Our results show that with the option of go-ahead-downloading, some content-based search operations can be effectively used for downloading digital video.

Ming-Syan Chen

Electrical Engineering Department, National Taiwan University, Taipei, Taiwan, ROC

Chung-Sheng Li and Philip S. Yu

IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA

Introduction

Due to their significant impact on both information providing services and entertainment, multimedia technologies have led to the creation of several new ventures. Recent advances in compression techniques have greatly reduced the size of digital video, thus overcoming the three major difficuties for handling digital video: the prohibitively large storage required to store digital video data; relatively slow storage devices for real-time video playout, and insufficient network bandwidth for real-time video transmission. As a result, due to its various advantages over analogue video, including better quality, scalability and portability, easy editing, efficient storage/retrieval and ease of being incorporated into a hypermedia environment, digital video is becoming the prevalent standard for video distribution for years to come.

Video-on-demand has been identified as an area with a fast growing market [1–4]. A video server for this purpose is expected not only to serve many clients concurrently (hundreds or more), but also to provide many interactive features for video playout, which home viewers have been enjoying from the current VCR system. However, recent studies indicated that to meet these requirements, the server would need a tremendous amount of computing power, storage, and communication bandwidth [5–6]. Thus, the feasibility of providing interactive video viewing over the network (including backbone and cable networks) needs further cost-justification. Consequently, as suggested in prior work [5], an alternative solution for video browsing would be to download the video data into the storage of the client station (or home VCR) which the end viewer can directly operate.

Currently in a commercially available VCR the functions for downloading are mainly time based. Explicitly, a viewer has to specify the channel, starting time and event

1077-2014/96/010035 +10 $18.00

duration in order to record an event of interest. However, there are many events whose starting times and durations are not available when a viewer intends to record them. For example, a tennis match could range from 1 to 5h. In addition, a regular program is usually pushed away from its scheduled time due to the occurrence of unexpected events, such as encore programs for a concert, over-time plays for sport tournaments and an unscheduled presidential speech, to name a few. As a result, due to the use of time-based recording, a viewer may have to either waste a lot of storage capacity (e.g., up to 400% in the previous example of a tennis match) or miss part or all the event of interest.

Note that the necessity of using time-based downloading basically stems from the difficulty of probing the video content in its analogue form. Such a difficulty is, however, significantly alleviated by the use of digital video. For digital video, a significant amount of research effort has been elaborated upon probing the video content and doing the index search and pattern matching [7–10]. These operations are intended to be done when the digital video is in its compressed form [11]. In view of the increasing popularity of digital video, it is important to explore the approach of content-based downloading and to incorporate it into client stations and home VCRs.

Consequently, we examine in this study a content-based method to download/record digital video from networks to client stations and home VCRs, as opposed to the conventional time-based method used for analogue video. A home VCR or client station with a content-based downloading controller is shown in Figure 1. For ease of description, we shall only use VCR as our platform in the following discussion. This method is particularly useful for downloading events whose occurrence times and event durations are not available *a priori*. Several approaches to probing the video content and to triggering the VCR operations are considered, including: frame signature matching; program barcode matching; preloaded pattern search, and annotation signal search in a hypermedia environment. A frame signature means a reduced set of digital data resulting from certain operations on that frame, and is expected to represent that frame uniquely. Usually, the starting and ending frames of an event of interest can be known in advance, thus allowing the use of a pre-stored frame signature to trigger the desired VCR operation. Barcodes mean some pre-determined numbers, each of which uniquely represents the start/end of a certain event. In addition to recording a single event, the method of content-based downloading can be extended to download a sort/group of events (such as all the news, weather forecasts, or Barney programs). Preloaded pattern search and annotation signal search are used for this purpose.

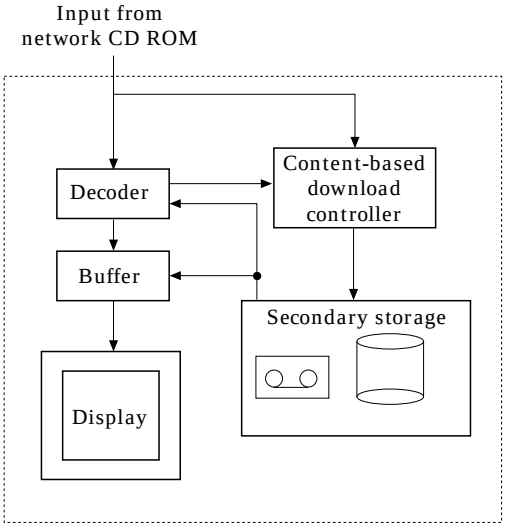

Figure 1. A home VCR or client station with a content-based downloading controller

Note that though the above approaches can be implemented by the current image processing techniques, their applications to real-time downloading may need further performance justification. To avoid the delay of downloading due to the execution of complex media processing, we propose a concept of *go-ahead-downloading*, which means that the downloading can start right after a scene change is detected rather than a scene being validated to be an interesting one. System capacity of the downloading device is evaluated. Moreover, to assess the performance of content-based downloading, we conduct an experimental study for preloaded pattern search, including template matching and texture matching. It is shown by our experimental results that with the option of go-ahead-downloading, some content-based search operations can be effectively used for downloading digital video. In addition, the search in low resolutions and compressed domains, when calibrated properly, can improve the search efficiency significantly. Proper video encoding methods are thus deemed important for the feasibility of content-based downloading.

Methods Considered

In this section, we explain the option of go-ahead-downloading and the methods considered for downloading digital video into client stations. Basically, instead of specifying its starting/ending time, the viewer specifies an event of interest (or a sort of events), which is then converted to an instruction for the content-based downloading controller, as shown in Figure 1, to utilize a predetermined technique to detect the video content and control the start/stop of the download-

ing. By using a proper software component, the VCR can detect the occurrence of such a frame/barcode/pattern and operate the desired functions accordingly.

As pointed out above, although the above approaches on frame/barcode/pattern matching can be implemented by the current image processing techniques, their applications to real-time downloading may need further performance justification. It is noted that in digital video, scene changes can be detected by comparing histograms of consecutive frames [8]. In general, this can be done in a real-time manner. With a proper scene change detection mechanism, it is found that the frame/barcode/pattern detection only has to be performed when a scene change occurs. Note that after a scene change is detected, the frame has to be further investigated to decide whether or not it is an interesting one that is meant to start a downloading process. Such media processing, however, will require a much longer execution time than scene detection, and tens (or hundreds) of frames may have passed before a scene is validated to be an interesting one. To avoid this deficiency, we propose a concept of go-ahead-downloading which, as shown in Figure 2, means that the downloading can start right after a scene change is detected rather than the scene is actually declared, by the corresponding media processing techniques, to be relevant to any programmed event. If that scene is later found to be irrelevant, one can simply reset the storage pointer to ignore the frames downloaded thus far. As such, one can ensure that there is no relevant video content missed due to the time elapsed during the media processing. This go-ahead-downloading option can therefore enable some complex media processing techniques to be useful for content-based downloading.

In this study, methods which can be implemented to probe the video content and control the VCR operations include:

(i) Frame signature matching, where the signatures of certain frames are pre-stored in the content-based downloading controller (CBDC) and used to detect the video content (i.e., occurrence of certain frames) and to control the VCR operations.
(ii) Barcode matching, where barcodes, representing the start/end of programs, can be embedded into compressed video streams, in-band transmitted together with video streams to the home VCR, and used by the CBDC to control VCR operations. There is no work on image processing required for this alternative.
(iii) Preloaded pattern search, where certain image patterns are stored in the CBDC to download a group of events.
(iv) Annotation signal search, where meta data is provided in hypermedia streams and used by the CBDC to interpret the stream content and take proper actions if necessary.

The concept of content-based downloading can be generalized into content-based filtering, which is expected to be important in collecting interesting information from hypermedia streams (video/audio/text) in an information super highway. The use of content-based filters in a hypermedia environment is shown in Figure 3, where an end user specifies the content of interest by a set-top-box (STB). The use of meta data in hypermedia streams and program barcodes in the broadcasting industry could further simplify the implementation of content-based downloading and increase its practicality and market value. As an effort to evaluate these media processing techniques, later in this paper we conduct a performance study for preloaded pattern search.

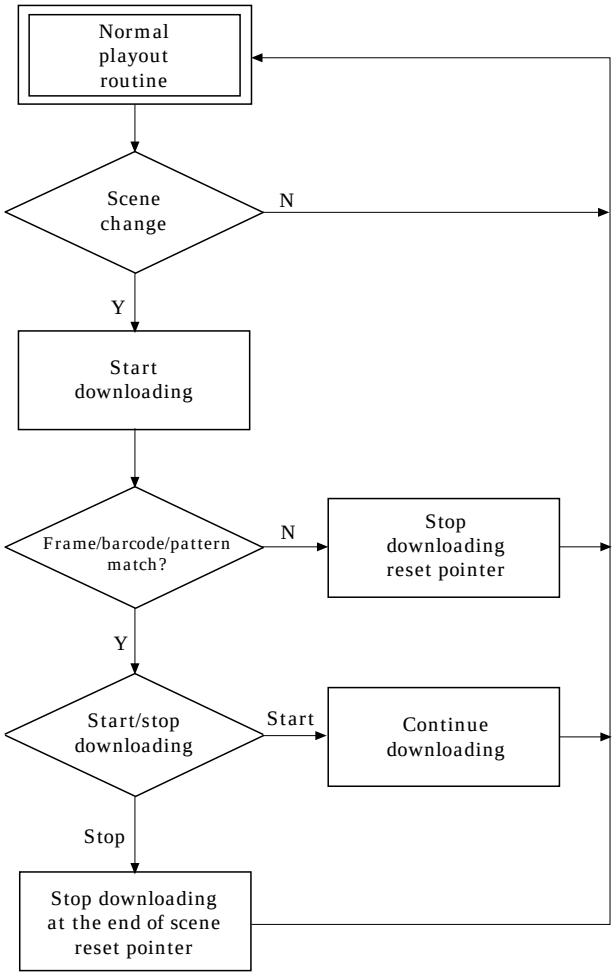

Figure 2. A flow chart for the VCR operations with content-based downloading controller

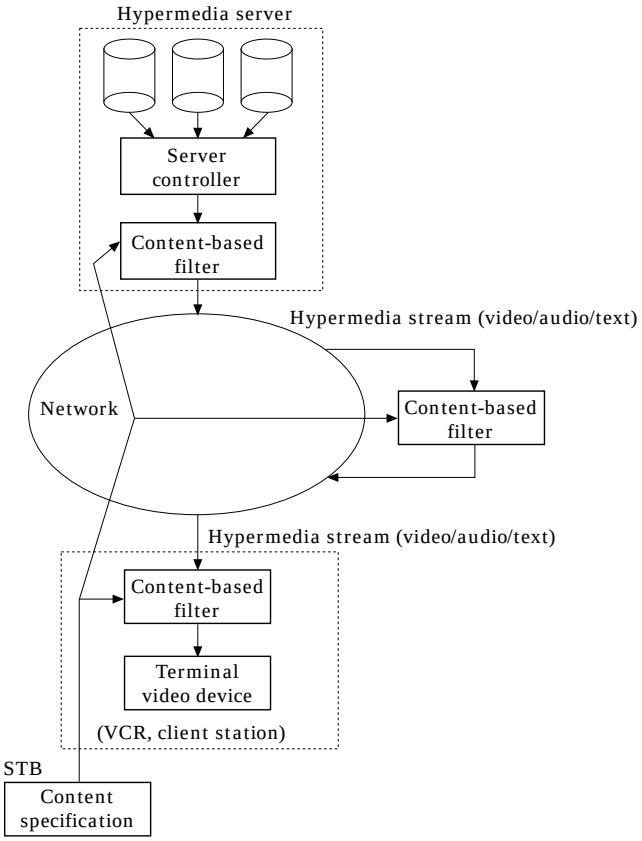

Figure 3. The use of content-based filters in a hypermedia environment

It can be seen that the other three methods described above can in fact be implemented in a straightforward manner, and their relative performance will be dependent upon the individual computing environment. Henceforth, we shall use the preloaded pattern search as the media processing technique in our discussion.

System Model

The proposed methods are devised for being incorporated into a home VCR or client station to provide content-based downloading capability. In addition to some technical issues on image processing, the primary challenge of using preloaded pattern search for content-based downloading is to search single or multiple video streams for single or multiple events of interest to be downloaded in real time. Generally speaking, the multiplication of the number of simultaneous video streams and the number of events which can be programmed for each stream can be viewed as the computation *capacity* of a content-based downloading device. Such a computation capacity of a content-based downloading device is in fact dependent on several factors, including scene change rates (say, 200–400 scenes/h), the computing power of the downloading device, and the buffer available to store programmed events.

Due to the random nature of the scene change process in each video stream, it is desirable to design a content-based downloading device to accommodate a scene change rate less than its peak value in order to reduce the hardware complexity. As stated before, the video streams from the network are stored directly into the secondary storage after being processed for scene change detections, as shown in Figure 2. Indexes to those unprocessed scenes are stored in the scene index table located at the content-based download controller. Note that overflow could occur in the controller memory during excessive scene changes. In that case, those unprocessed scenes will have to be retrieved from the secondary storage through the scene index table.

The scene length in video streams can be modeled by a geometric distribution [12]. Using this model, the probability p that a video frame contains a scene change is dependent upon the average length of a scene (in seconds), L, in such a way that

$$p = \frac{1}{1 + fL}, \tag{1}$$

where f is the frame rate (frames/s).

Clearly, it is important to minimize the probability of missing a triggering event for either starting or terminating the downloading operation. Errors in determining a triggering event could arise due to:

(i) misses or false alarms in performing the prestored pattern matching, or
(ii) misses of scenes due to buffer overflow.

Explicitly, the probability of missing a triggering event, P_M, given the existence of such an event can be expressed by:

$$P_M = P_m(1 - P_s) + P_s, \tag{2}$$

where P_m is the probability of missing a triggering event under the approach of preloaded pattern and P_s is the probability of missing a scene change due to buffer overflow. Let P_{fa} be the probability that a frame causes a false alarm. Then, the probability of introducing a false alarm, P_{FA}, given that the event does not exist is

$$P_{FA} = P_{fa}(1 - P_s). \quad (3)$$

Hence, the total error probability of both misses and false alarm equals

$$P_E = P_M P_{event} + P_{FA}(1 - P_{event}), \quad (4)$$

where P_{event} is the probability that a triggering event will occur. The contour plot of P_E as a function of P_s and P_{event} for $P_m = 10^{-5}$ and $P_{fa} = 10^{-5}$ is shown in Figure 4. From Figure 4, it can be seen that given an event probability, P_{event}, the total error probability is minimized when the probability of missing a scene change due to buffer overflow, P_s, is minimized.

Note that the missing of a starting event could cause the missing of the entire program, while the missing of a termination event could cause the secondary storage overflow. On the other hand, a false alarm of the starting event is most likely to cause the overflow of the secondary storage unless it is paired with the false alarm of an ending event. False alarm of an ending event will cause early termination of downloading of a program.

To provide more insights into the buffer size required, a time-driven simulator is developed to evaluate the relationship between the probability of missing a scene due to buffer overflow. Two design alternatives for buffer management are evaluated here. In the first case, it is considered that the content-based downloading device has a limited memory buffer available and the size of the secondary storage for storing the video frames is assumed to be adequate. As such, memory buffer overflow occurs only when the detection of a new scene coincides with the buffer full condition. Under this circumstance, the downloading device deletes the latest scene stored to make room for the newly arrived scene. The scene loss probability, as a function of the buffer size, is shown in Figure 5 for various values of the scene processing time *T*. Note that the scene processing time, *T*, is dependent upon the media processing technique employed. An average scene length of 12 s/scene and a frame rate of 30 frames/s are employed in the simulations. As shown in Figure 5, when the processing time is less than 8 s/scene, to achieve a scene loss probability less than 10^{-5}, the buffer size required is less than 10 (i.e., the size required to accommodate 10 frames which are associated with scene changes). However, when the processing time approaches the scene length, the required buffer size increases drastically.

In the second case, the incoming video frames are assumed to be stored into the secondary storage directly. Those frames with scene changes are then retrieved for prestored pattern matching. When buffer overflow occurs, those frames associated with the currently processed scenes are purged from the secondary storage to make room for the incoming frames. The scene loss probability as a function of the buffer size for this scenario is shown in Figure 6. Similarly to the previous case, an average scene length of 12 s/scene is used in the analysis. It can be inferred from Figure 6 that when the processing time is 4 s/scene (corresponding to 120 frames), the buffer size required is approximately for 750 frames in order to achieve a scene loss rate less than 10^{-5}. The buffer size required increases to 3000 when the processing time increases to 8 s/scene.

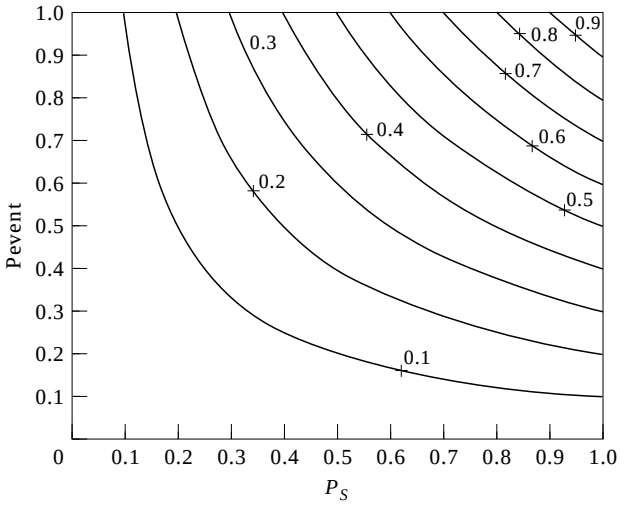

Figure 4. The error probability P_E as a function of P_s and P_{event}

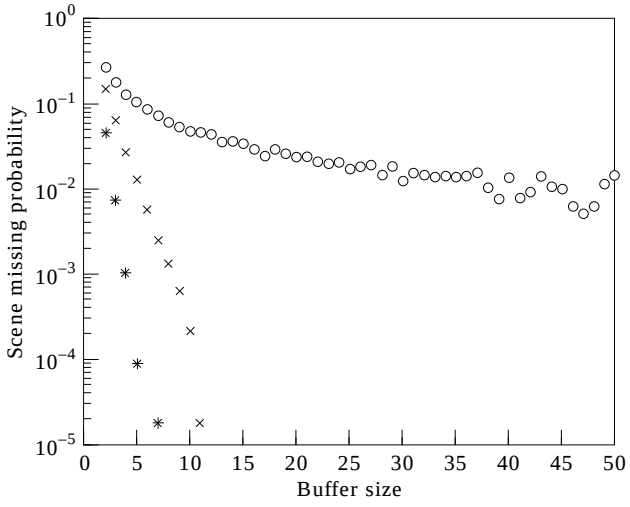

Figure 5. Case 1: scene loss probaility vs. memory buffer size. *, T = 4s; X, T = 8s; O, T = 12s

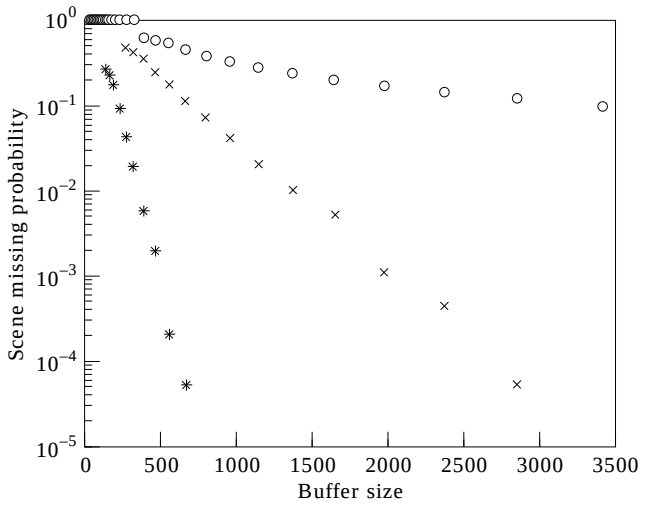

Figure 6. Case 2: scene loss probability vs. storage buffer size. *, T = 4s; X, T = 8s; O, T = 12s

When the processing time of a scene approaches the length of a scene, the required buffer size increases drastically. In practice, a combination of memory buffer and secondary storage with finite size is used to store those unprocessed scenes and video frames. Using an average scene length of 12 seconds and a scene processing time of 2 s, it can be obtained that a total of 10 memory frame buffers and 750 secondary storage buffers are needed to achieve a scene loss probability of 10^{-5} or lower. There is a clear design trade-off between the cost and the system reliability.

Search by Pattern Matching

In this section, two methods on preloaded pattern search, template matching and texture matching, are investigated and their potential to be performed in real-time is evaluated.

Template Matching

Template matching can be used for searching video program events which contain certain images, such as the CNN or IBM logo, photos of presidents, or some objects of interest (e.g., a space shuttle). A template here refers to the image of interest.

Template matching can be accomplished by computing the statistical correlation function, $R_s(m, n)$, between the image, $I(i, j)$ and the template, $T(i, j)$, as defined below [10]:

$$R_s(m,n) = \sum_j \sum_k I_N(i,j) T_N(j-m, k-n) \qquad (5)$$

where $I_N(i,j) = \dfrac{I_M(i,j)}{\sqrt{\Sigma_i \Sigma_j I_M(i,j)}}$ and $I_M(i,j) = I(i,j) - \bar{I}.\ \bar{I}$

denotes the average of the image [13]

Similarly, $T_N(i,j) = \dfrac{T_M(i,j)}{\sqrt{\Sigma_i \Sigma_j T_M(i,j)}}$ where $T_M(i,j) = T(i,j)$

A legitimate match between the image and the template is declared when the computed value of the corresponding correlation function is larger than a certain threshold. Note, however, that template matching is very computation intensive and speed-up methodologies are thus required to make it useful as a real-time video operator. In our experiments, we employ a progressive method to achieve the necessary speed-up [14]. Specifically, each scene is restructured into a pyramid such that the correlation can be performed hierarchically. The restructuring method needs to preserve both the spatial and spectral locality of an image at each level of the pyramid so as to allow the search process to perform progressively. This restructuring process can then be accomplished by either:

(i) decomposing the image spatially into blocks and applying block-based transformation (e.g., DCT) to decompose the image spectrally, or
(ii) decomposing the image spectrally by multi-resolution type of transformations (e.g., subband and wavelet transformation), and then decomposing the image spatially into blocks.

It is noted that digital video data is usually compressed by DCT-based compression algorithm, such as motion-JPEG or MPEG, whereas wavelet or subband-based products are also becoming available. After each image frame in a video stream is restructured either spectrally or spatially, the template matching process can be progressively performed accordingly.

For block DCT-based transformation, each image is divided into two hierarchies in the pyramid in such a way that: (a) the DC coefficients of each block are collected for a low-resolution representation of the image; (b) all the other coefficients within a block are used for the full representation of the image. A progressive template matching is hence to correlate the low-resolution version of the template with that obtained from the image. The correlation of the neighborhood of those possible candidates in full resolution is then computed if the exact location of the match needs to be determined.

Texture Matching

Texture matching can be used to search for video program events which contain objects of certain textures (such as grasslands, forests, mountains and lakes). Allowing region-based image indexing, texture is usually used to describe two-dimensional variations of an image, in which the elements and rules of spacing or arrangement may be arbitrarily manipulated while a characteristic repetitiveness remains. Texture usually depends on some local order repeated over a region, non-random arrangement of elementary parts, or uniform entities with approximately the same dimensions within the textured region.

Similarly to template matching, the speed of the texture extraction and comparison process can be significantly improved by computing the texture features progressively according to the following steps for a given scene:

(i) apply subband coding or wavelet transformation to generate scenes with multiple resolutions; or collect DC coefficients from DC T-transformed scenes for a low-resolution representation of the image;
(ii) divide the transformed scenes into regular or irregular regions at each resolution according to the boundaries extracted by edge detection or image segmentation;
(iii) compute the texture feature vector of each region at the lowest resolution;
(iv) select regions whose feature vectors are closest to those extracted from the target template for the comparison at the next higher resolution;
(v) compute the feature vectors of those regions at the next higher resolution and compare them with the corresponding results from those extracted from the target template;
(vi) continue this process until the target resolution is reached;
(vii) output region(s) identified.

In our experiments, we consider the following texture features: fractal dimension; coarseness, and entropy. Based on the foregoing descriptions, we extract the texture features at various resolutions and perform the progressive feature extraction accordingly.

Fractal dimension of a random function $I(x)$ is defined as $T + 1 - H$ where T is the topological dimension of $I(x)$, while H is a parameter of the fractal Brownian function $I(x)$ if for all x and $\triangle x$:

$$Pr\left(\frac{I(x+\Delta x)-I(x)}{\|\Delta x\|^{H}} < y\right) = F(y) \tag{6}$$

is satisfied. Fractal dimension is one of the measures that can be used to measure disordered texture. In particular, it is useful for measuring the surface roughness. It is chosen in this paper because of its invariance to linear transformation of the data and the transformation of the scale. In this paper, the reticular cell counting method [15], in which the number of cubes at different scales that are passed by the image surface are counted, is used to compute the fractal dimension.

Coarseness is defined as

$$C = 1 - \frac{1}{1+S_D} \tag{7}$$

It is related to the dispersion of the image,S_D, defined as

$$S_D = \sum_{i=0}^{L-1} (i - S_M)^2 h[i] \tag{8}$$

where D_M is the mean of the histogram, and L is the number of levels used in accumulating the histogram, $h[i]$.

Entropy is defined as

$$S_E = -\sum_{i=0}^{L-1} h[i] log_2 h[i] \tag{9}$$

where the entropy is a stochastic measure on the randomness of an image. Note that both coarseness and entropy are computed from the first-order histogram of the image.

Experimental Results

In this subsection, we examine the feasibility of performing template matching and texture matching for content-based downloading by investigating the computation speed and accuracy of these two operations. The experiments were performed on an IBM RS/6000 Model 560 system with approximately 50 MIPS.

Two images, referred to as images A and B, are used in our experiments. Image A, as shown in Figure 7, is a 1024 × 1024 image of the San Francisco downtown, and is employed for template matching. The event to be searched in this case is the landmark tower in the photograph, as shown in Figure 8. The size of the template is 256 × 256. Image B, as shown in Figure 9, is a 1024 × 1024 satellite image taken from Landsat Multispectral Scanner (MSS), and is employed for both template and texture matching. Image B is selected

Figure 7. Image A used for the template experiment

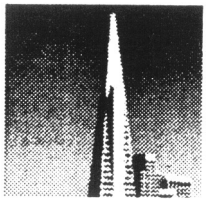

Figure 8. Template A used for the template matching experiment

because of its rich texture features so that template and texture matching on the same image can be performed. The template used, as shown in Figure 10, is a 64 $\times$ 64 image extracted from the original image at location (128,128). Different resolution levels of the images are obtained from subband coding with quadrature mirror filtering [16]. Thus, level 0 corresponds to the full resolution, and a larger resolution level corresponds to a coarser resolution of an image. It should be noted that even though subband coding is used in this study, other transformation, such as DCT, can also be used to extract the lower resolution version of an image.

We first performed template matching on image A. The computation time of template matching is shown as a function of the resolution level in Figure 11. Note that only the existence of a certain event needs to be determined in content-based downloading applications. This is in contrast to applications such as image registration where the location of an image that matches to a template is also important. As the resolution decreases (i.e., the resolution level increases), the computation time decreases significantly while

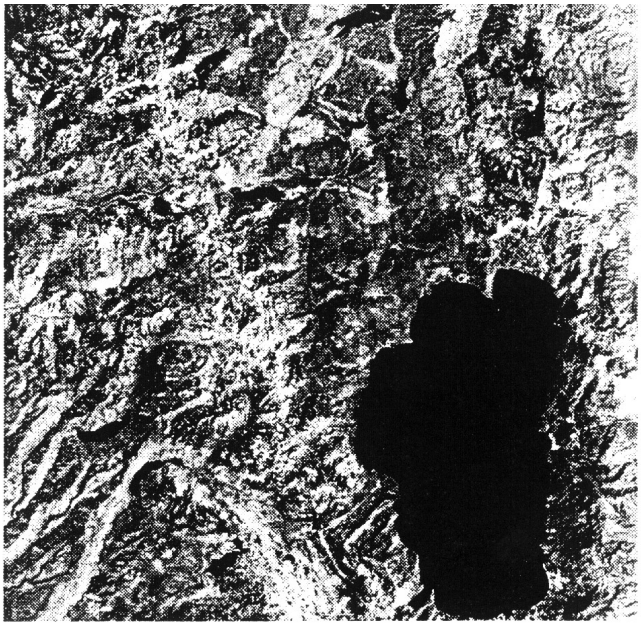

Figure 9. Image B used for the template and texture matching experiment

Figure 10. Template B used for the template and texture matching experiment

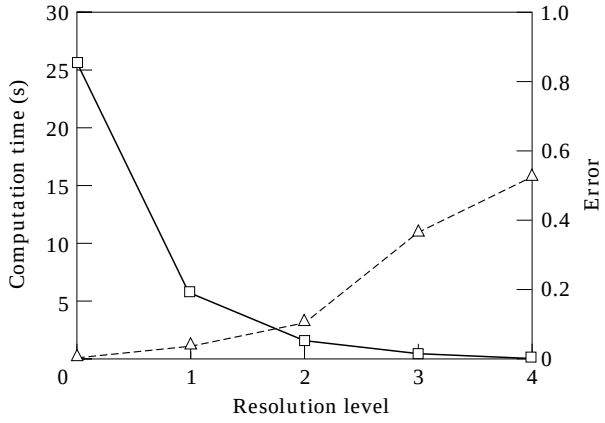

Figure 11. Speed and accuracy of template matching on image A under the transformed domain

a match may still be identified. However, the accuracy of the correlation decreases. From our experiments, it is shown that the event can still be located even at the fourth level of the resolution due mainly to the distinctive feature possessed by the landmark used for the matching.

The template matching experiment is also conducted on image B with the template in Figure 10. The computation time of this template matching is shown in Figure 12 as a function of the resolution level. Similarly to the previous image, the computation time decreases significantly as the resolution decreases. However, a correlation peak can no longer be found beyond the third resolution level in this case because of the lack of distinct feature in the template.

The computation speed of texture matching, as a function of the resolution level, is shown in Figure 13. It is observed that a similar drop in the computation speed occurs when the resolution level of the image becomes larger. However,

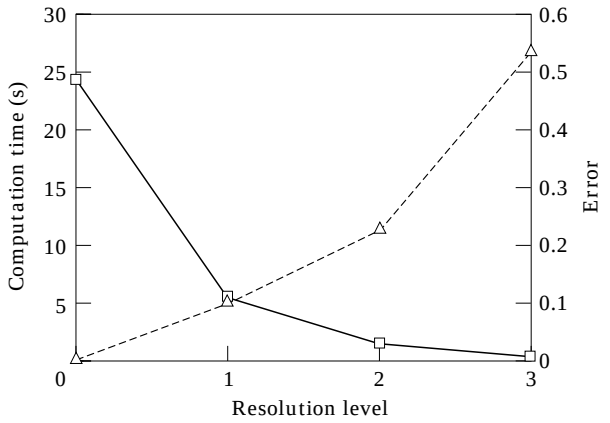

Figure 12. Speed and accuracy of template matching on image B under the transformed domain

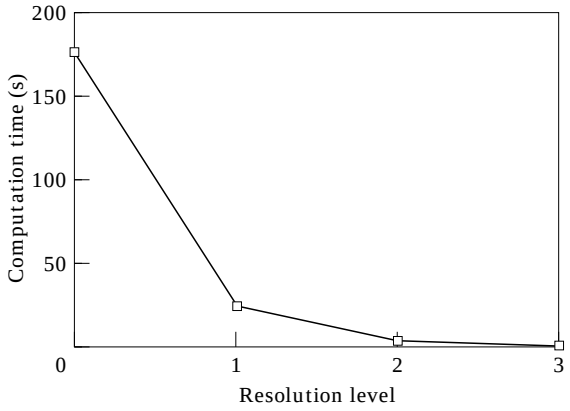

Figure 13. Speed of texture matching on image B under the transformed domain

the best match might be overlooked when one goes beyond the third resolution level, again showing a design trade-off among the computing speed and the accuracy of the results. Certainly, the experimental results will be very dependent upon the nature of the events of interest. From the magnitude of the execution time observed in our experiments, it can be seen that by utilizing the option of go-ahead-downloading, content-based search operations, such as template and texture matching, can be effectively applied to downloading digital video. In addition, it is shown that the search in low resolutions and compressed domains, when calibrated properly, can improve the search efficiency significantly. Thus, proper video encoding methods are deemed important for the feasibility of content-based downloading. With a given computing power and available buffer space of a downloading device, one can hence empirically estimate the computation capacity of a downloading device.

Conclusions

In this paper, we examined a content-based method to download/record digital video from networks to client stations and home VCRs as an alternative to the conventional time-based method used for analogue video. Various approaches to probing the video content and to triggering the VCR operations were considered, including frame signature matching, program barcode matching, preloaded pattern search, and annotation signal search in a hypermedia environment. A concept of go-ahead-downloading was introduced and system capacity of the downloading device was evaluated. Experimental studies have been conducted on preloaded pattern search, including template matching and texture matching, to provide some insights into this approach. Our results suggested that with the option of go-ahead-downloading, some content-based search operations can be effectively used for downloading digital video. In addition, the search in low resolutions and compressed domains can improve the search efficiency significantly.

Acknowledgements

The authors would like to thank J. Turek at IBM Research for his codes on C++ image library, and T. Chen at AT&T Bell Lab. for his codes on subband encoding. This project is in part supported by NASA/CAN Grant No. NCC5-101.

References

1. Deloddere, D., Verbiest, W. & Verhille, H. (1994) Interactive video on demand. *IEEE Comm Mag*, **32**(5):82–88.

2. Ozden, B., Biliris, A., Rastogi, R. & Silberschatz, A. (1994) A low-cost storage server for movie on demand databases. In *Proceedings of the 20th International Conference on Very Large Databases*, pp. 594–605.
3. Rangan, P. V., Vin, H. M. & Ramanathan, S. (1992) Designing a multi-user multimedia on-demand service. *IEEE Comm Mag*, **30**(7):56–65.
4. Vin, H. M. & Rangan, P. V. (1993) Designing a multi-user HDTV storage server. *IEEE J Selec Areas Comm*, **11**(1):15–16.
5. Chen, M.-S. & Kandlur, D. D. (1995) Downloading and stream conversion: Supporting Interactive Playout of Videos in a Client Station. *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pp. 73–80.
6. Chen, M.-S., Kandlur, D. D. & Yu, P. S. (1994) Support for fully interactive playout in a disk-array-based video server. *Proc. ACM Multimedia*, pp. 391–398.
7. Arman, F., Depommier, R., Hsu, A. & Chiu, M.-Y. (1994) Content-based browsing of video sequences. *Proc. ACM Multimedia*, pp. 97–104.
8. Deardorff, E., Little, T. D. C., Marshall, J. D. & Venkatesh, D. (1994) Video scene decomposition with the motion picture parser. *Proc. SPIE*, 2187, *Digital Video Compression on Personal Computers: Algorithms and Technologies*, pp. 44–55.
9. Smolair, S. W. & Zhang, H. (1994) Content-based video indexing and retrieval. *IEEE Multimedia*, **1**(2).
10. Zang, H. & Smolair, S. W. (1994) Developing power tools for video indexing and retrieval. *Proc SPIE*, 2185, *Storage and Retrieval for Image and Video Databases*, pp. 140–149.
11. Arman, F., Hsu, A. & Chiu, M.-Y. (1993) Image processing on compressed data for large video database. *Proc. ACM Multimedia*, pp. 267–272,
12. Krunz, M. & Hughes, H. (1995) A traffic model for MPEG coded VBR streams. *Performance Evaluation Rev*, **23**(1):47–55.
13. Pratt, W. K. (1971) *Digital Image Processing*. New York: Wiley.
14. Li, C.-S., Turek, J. J. & Feig, E. (1995) Progressive template matching for content-based retrieval in Earth observing satellite image database. *Proc. SPIE Photonics East 95*, 2006.
15. Sarkar, N. & Chaudhuri, B. B. (1994) An efficient differential box-counting approach to compute fractal dimension of image. *IEEE Trans Sys, Man Cybernet*, **24**(1):4–24.
16. Smith, J. R. & Chang, S.-F. (1994) Quad-tree segmentation for texture-based image query. *Proc ACM Multimedia*.