Retrieving quality video across heterogeneous networks

Video over Wireless

JOSÉ M. F. MOURA, RADU S. JASINSCHI, HIROHISA SHIOJIRI, AND JYH-CHERNG LIN

The widespread availability of wireless networking will add mobility as a new dimension to computing. However, the wide range of network bandwidths (four orders of magnitudes) and the dynamic abrupt changes of the throughputs that mobile users will experience (two orders of magnitude) pose challenges to the deployment of a truly seamless computing environment. The goal of this article is to address the issues that arise in delivering video across Wireless Andrew. Wireless Andrew is the Carnegie Mellon University (CMU) heterogeneous networking environment [1]. It interconnects wideband and narrowband wireless and wired networks using off-the-shelf technologies.

We design a video system where video servers distributed across the network deliver, upon request, video clips to clients scattered around the campus. The quality of the video is to be automatically adjusted, with graceful degradation, to the current client's quality of service (QoS) parameter requirements. For mobile clients, this readjustment is dynamic with time as the user roams around campus.

The widely different transmission rates force an aggressive compression of the original video sequence, while nomadic computing with low-performance machines calls for simple decoding algorithms. These requirements are beyond what is possible with current video compression technologies like H.261, Joint Photographic Experts Group (JPEG), or Motion Picture Experts Group (MPEG). The main novelty of the article is Generative Video (GV), a content-based meta representation for video that is well suited for the heterogeneous dynamic environment of Wireless Andrew. GV provides a framework for graceful degradation as end-to-end network throughput varies. GV reduces the video sequence to a small set of *still* images and side information needed to reconstruct the original sequence. We have demonstrated for three real live video sequences that GV delivers compression ratios in the range of 1000–10000 with good perceptual quality [2].

We now summarize the remainder of the article. The following section provides further details on the definition of the video service. GV is described in the third section. In the fourth section we develop a scalable implementation for the GV coder/decoder (codec). Scalability coding avoids duplicating the encoding of each video clip when servicing a wide range of throughputs, as in heterogeneous networks. In the fifth section, we discuss the video delivery requirements, the video system architecture, the system implementation, and the trade-offs needed to ensure graceful degradation under real-life network operating conditions. Finally, the last section concludes the article.

The work reported in this article has been partially supported by INI.

Video over Wireless Andrew

The focus of this article is on delivery of video clips over heterogeneous networks. The clips are retrieved from a video database. These clips may come from an International Consultative Committee on Radio (CCIR) 601 video source at 216 Mb/s data rate, from uncompressed maximum frame rate Common Intermediate Format (CIF) quality corresponding to 36.5 Mb/s, or from Quarter CIF (QCIF) at 9.1 Mb/s, [3]. These widely different raw data rates correspond to widely different resolutions. To provide acceptable service over the entire range of throughputs present in the network, it is of major importance to dynamically readapt the QoS parameters of the video service.

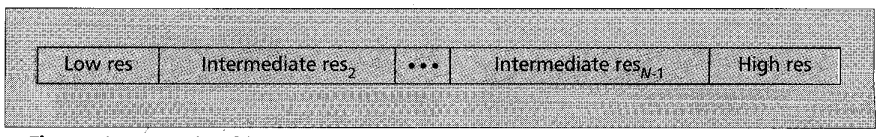
In networks with wireless links, there are abrupt changes in the throughput available to mobile users. If not taken care of appropriately, this may have dire consequences. For example, if a user falls out of the range of service of a wireless local area network (LAN) station, at 2 Mb/s, into the range of a wireless wide area network (WAN) station, at say 20 kb/s, this drop of two orders of magnitude in data rate translates into a two-order-of-magnitude delay in transmission. A video clip taking initially 1 min to transmit will then require about 1.5 hr to retrieve, unless the network quickly readjusts the coding to allow for these dramatic changes in bandwidth.

With applications such as accessing web pages with video, these are increasingly important issues that have to be dealt with for wireless and mobile computing to provide levels of service equivalent to those that wired networks currently deliver.

These considerations entail that we dynamically readjust the quality of the video to the available throughput; in other words, that we strike a balance between the two conflicting requirements of video resolution and transmission delay. However, in a heterogeneous environment, there is a wide range of available throughputs. Rather than having a different encoded stream for each particular value of throughput, we

1070-9916/96/$05.00 © 1996 IEEE

consider hierarchical scalable video encoding mechanisms [4]. A hierarchical bit-stream is illustrated in the diagram displayed in Fig. 1. The leftmost bits are delivered to all clients, providing the lowest video resolution available. The next leftmost bits of the video stream, when combined with the lowest-resolution bits, enhance the video quality. These bits are now delivered to all but the clients with throughput falling below a certain threshold. Each additional section of the video stream, when combined with the previously delivered bits, will enhance the video quality, while requiring a higher throughput.



■ **Figure 1.** *Hierarchical bit-stream.*

Network	Transmission rate	MAC protocol
Ethernet	10 Mb/s	CSMA/CD
WaveLAN	2 Mb/s	CSMA/CA
CDPD	19.2 kb/s	DSMA/CD

■ **Table 1.** *Network specifications.*

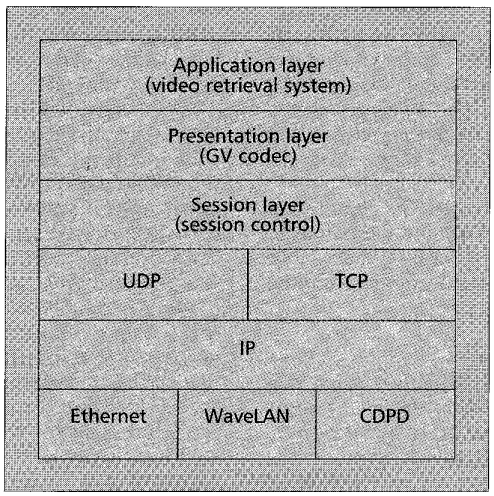
For the purposes of this article, we identify three major networks in Wireless Andrew: Ethernet wired LANs, wireless LAN using WaveLAN technology, and wireless WAN using the CELLULAR DIGITAL PACKET DATA (CDPD) protocol. The CDPD protocol uses analog transmission cellular technology and is available on an experimental basis to CMU researchers, courtesy of Bell Atlantic Mobile Systems. Table 1 provides the specifications for these three networks. The nominal transmission rates are 10 Mb/s for Ethernet, 2 Mb/s for the wireless LAN, and 19.2 kb/s for the CDPD-based WAN. Actual throughputs may be much smaller. Table 1 also lists the low-layer protocols, namely, the medium access control (MAC) protocols: carrier sense multiple access (CSMA) with collision detection (CSMA/CD) for the Ethernet network, CSMA with collision avoidance (CSMA/CA) for the wireless LAN, and digital sense multiple access (DSMA) with CD (DSMA/CD) for CDPD.

Figure 2 shows the protocol stack used: user datagram protocol/Internet protocol (UDP/IP) and transmission control protocol (TCP)/IP suites. Higher layers include the application layer, the presentation layer, and the session layer. These layers are system-specific. The video retrieval lies in the application layer. It includes the user interface and the video display module. The GV representation, GV scalable coding, and video resynchronization modules are in the presentation layer. The session layer manages the dialogue between the client and the video server.

Generative Video

In this section, we introduce Generative Video (GV). GV has the potential to deliver good perceptual quality video after it has been compressed by factors in the range of 1000–10000.

Central to GV is its content-based *meta* video representation. Rather than describing video in terms of pixels and frames, GV represents video by its contents. Consider a sequence displaying a static background with cars moving in and out of view. If represented in terms of pixels and frames, this sequence generates a large amount of video data. On the other hand, if we identify the sequence background, the cars, and their relative motion information, and if we represent each of these components in a compact way, then we reduce tremendously the amount of data needed to describe the sequence. The latter approach provides a content-based representation [5–8] for the video sequence. GV is content-based. Its meta representation is in terms of informational units, structured operators, and side information. We refer to these as the GV *constructs*.



■ **Figure 2.** *Protocol stack.*

The informational units are called *world images*. World images are augmented still images obtained from the video sequence by integrating across the sequence the nonredundant information present in the sequence. The world images *encode* the video sequence information contents, for example, the image background and the different objects (cars, persons) moving across the image.

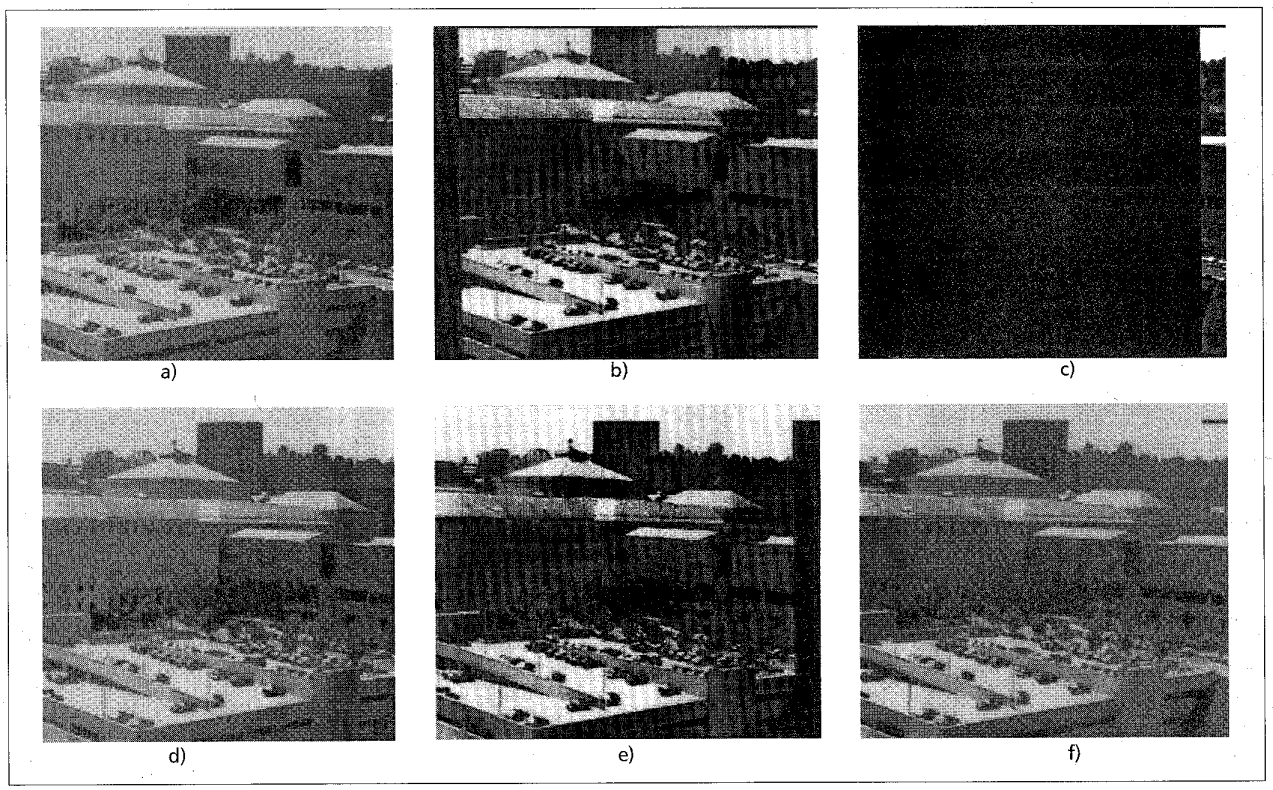
Two important issues related to the world images are the Tesselation Principle and the Stratification Principle. The Tesselation Principle specifies how the shape of the moving objects is described in GV. This shape may be very complex, making it awkward to manipulate the objects. To simplify the implementation of GV, we use simple geometric shapes to outline the moving objects (see below). The Stratification Principle states that the moving objects move parallel to the background. This means that the relative depth of the moving objects with respect to the camera (stratification information) stays the same throughout the sequence. This stratification information is needed to determine image occlusion.

The *generative operators* describe a suite of operators which *access* and *manipulate* the video sequence contents. They include signal processing operators, motion operators, window operators, and cut-and-paste operators. The signal processing operators estimate motion and segmentation information from the video sequence. The motion operators encode the dynamics of the camera or of the moving objects. Window operators frame each of the actual images in the sequence at display time. Cut-and-paste operators incrementally build the world images. These operators are also used to reconstruct the original video sequence or a new video sequence by recombining the informational units.

The side information, referred to in GV as *ancillary data*, specifies the velocity of the camera, the velocity of each moving object, the shape of each moving object, and the relative stratification of the moving objects.

GV is clearly distinct from pixel-based methods like wavelets or dis-

[1] *In fact, these methods can be applied in conjunction with GV for compressing, as still images, the world images; see the following section.*



■ **Figure 3.** *Carnegie sequence: a) Image* I_1*; b) Image* I_1 *registered with respect to (expanded) background world image; c) vertical stripe; d) Image* I_{10}*; e) Image* I_{10} *registered with respect to (expanded) background world image; f) resulting background world image.*

crete cosine transforms.[1] GV does have similarities with mosaicing [9–11], which is used, for example, to stabilize the field of view of a camera going through rough motions; it is concerned with building panoramic views — mosaics — of a background. These mosaics are constructed by registering successive frames and averaging them. Mosaicing is simple and efficient for stabilization purposes; but when moving objects are present, this averaging introduces "ghost" artifacts at the locations where the objects have been. Therefore, GV goes beyond mosaicing; it is a comprehensive representation for video. It handles video with moving objects, recorded with a moving camera; it decomposes video into its basic components, background and moving objects, which may possibly occlude each other. The output of GV is a complete representation of all structural elements composing the video. By determining camera velocity and tracking each moving object as a whole, GV deals with global motions rather than pixel or local motions. GV provides compact models to represent the shape of the moving objects and defines highly structured and simple GV operators to efficiently handle video analysis, manipulation, and synthesis.

In the following subsection, we discuss the GV representation and, after that, the general applicability and limitations of our current implementation of GV. In the fourth section, we develop a scalable implementation and analyze the potential for compression afforded by GV.

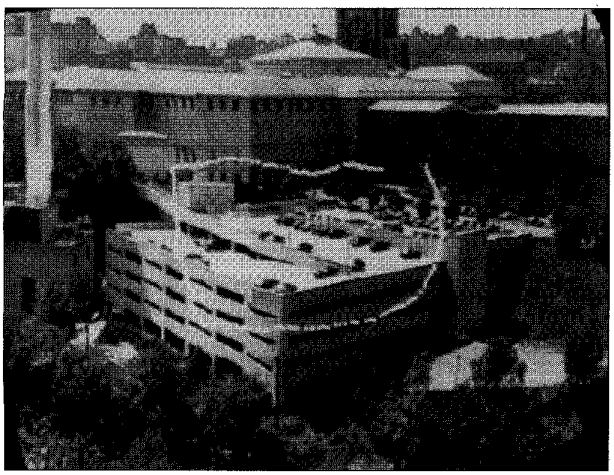
GV Representation

In general, a video sequence comprises elementary video subsequences, which we will refer to as *scene shots*, or simply *shots*. For example, a shot may correspond to a static background with a person walking across the scene. Another shot might show a different background panned by the camera with cars moving in and out of the scene. Camera panning induces background motion. We assume that the video sequence is segmented into different shots. GV reduces each shot to a set of constructs — world images, generative operators, and ancillary data — and represents video in a highly structured form. GV accomplishes this by spatial and temporal integration of the video sequence information.

The information units in GV are defined by attributes. In the implementation of GV that we discuss in this article, the defining attribute is motion. Other attributes that can be used are shape, depth, color, or texture, for example. Objects moving with respect to the background are identified as *figures*. Anything else that exists in the sequence, but does not move with respect to the background throughout the shot, is considered part of the background. Key steps in GV are motion estimation, segmentation, and shape tesselation. Once all figures have been segmented and tesselated, we construct the world images by *cut-and-paste* operators.

GV reduces the video shot to a set of still images. The still images are the world images, one distinct world image for each independently moving figure and one world image for the image background. These world images are ordered in layers according to how the figures occlude each other and how they occlude the image background. This is described by the Stratification Principle [5–8], which encodes image occlusion information at the world image level.

We explain GV in steps. We first consider the simple case of a sequence where there are no moving objects, only camera motion. The sequence is the *Carnegie sequence*. We assume that the camera motion is known and explain how to construct the world image. Then we consider a more complex sequence, the *Samir sequence*, with a figure — a walking person. We discuss how to segment the figure and how to tesselate it using simple shapes. Finally, we comment briefly on how to estimate the motions and other related signal processing issues. We



■ **Figure 4.** *Background world image for the Carnegie sequence.*

refer the reader to [5–8, 12] for full details. The references explain how to generate the GV representation and the associated signal processing algorithms in the general case of moving camera and multiple moving objects. They describe the experimental results with more elaborate sequences with up to 12 objects moving in and out of view.

World Image Generation: Carnegie Sequence – The Carnegie sequence is a 10 s video clip recorded with a handheld camera viewing at a distance a stationary background (the back of the Carnegie Museum in Pittsburgh). The only GV components needed in the representation of the Carnegie sequence are the background world image and the ancillary data. The background world image Φ is the enlarged view of the world as observed by the camera throughout the entire sequence, after we eliminate all redundancy between frames. The ancillary data collects all side information needed to regenerate the original sequence like the dimensions, in pixels, N_x^Φ x N_y^Φ, of the background world image Φ, the dimensions, N_x^I x N_y^I, of each frame I_k in the sequence, the number N_r of frames in the video clip, the starting pixel position (x_0, y_0) of the camera in the background world image, and the camera motion in pixels per frame, $\bar{v} = (v_{x,k}, v_{y,k})$, $k = 1, \ldots, N_r$. More complex sequences will include additional data such as figure world images, motions, and occlusion information for each moving object.

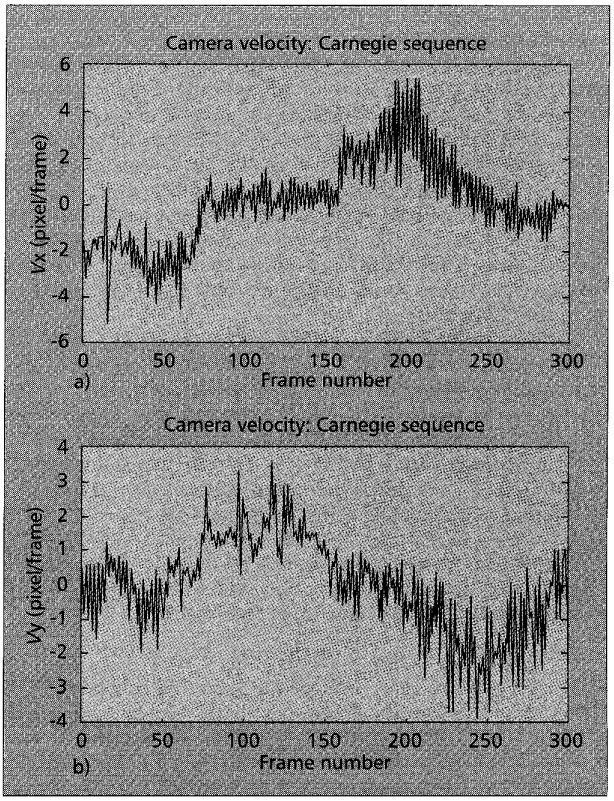
We describe briefly the construction of the world image by cut and paste operators using images I_1 and I_{10} of the Carnegie sequence.[2] These images are shown as Figs. 3a and 3d.

The world image is generated incrementally. The world image is initialized as $\Phi_1 = I_1$, that is, the first instance of the world image is the first frame in the sequence. When the next frame is available, in this case frame I_{10}, we first expand the dimensions of the background world image from the original dimension in pixels of 240 x 256 to 243 x 278. This corresponds to an overall vertical motion of 3 pixels and to a horizontal motion of 22 pixels. Then we register the current (old) instance of the background world image and register the current frame with respect to the expanded background world image (see Figs. 3b and 3e). As mentioned, the expansion and registration operations use the camera velocity data. We then cut from this expanded background world image instance its overlap with the frame being processed (blackened portion in Fig. 3c) and paste the remainder of the old background world image (right portion of Fig. 3c) to the registered current frame, Fig. 3d; this generates the new instance of the background world image, which is illustrated as Fig. 3f. It has dimensions 243 x 278 pixels. Figure 4 shows the background world image of the Carnegie sequence after processing, in a similar way, all of its 300 frames. The white line superimposed on the figure shows the trajectory of the center of the recording camera.

The horizontal and vertical components of the camera motion, in pixels per frame, are shown in Fig. 5.

Motion Estimation and Segmentation of Figures – We discuss briefly the issues of motion determination and figure segmentation. In general, video sequences contain camera motion, which induces background velocity on the image plane, and figure motion. Our motion and segmentation algorithm is detailed in [5, 6, 12] and briefly discussed in [7, 8]. The algorithm is described by the following steps:

1. Estimation of the (local) image velocity vectors. We work with an affine motion model, which is appropriate for describing translation, rotation, and scaling motions. We use a pyramid structure to expedite the computation of the motion field.
2. Determination of the (global) image background velocity by a velocity histogram method. We assume that the peak of the histogram corresponds to the motion of the image background.
3. Image registration, which aligns consecutive images of the sequence such that the resulting image background velocity



■ **Figure 5.** *Camera velocity (pixels/frame) for Carnegie sequence: a) horizontal velocity; b) vertical velocity.*

2 In actual processing, we process consecutive pairs of images, but to illustrate the visual effect of cutting and pasting, we consider these two well-separated frames.

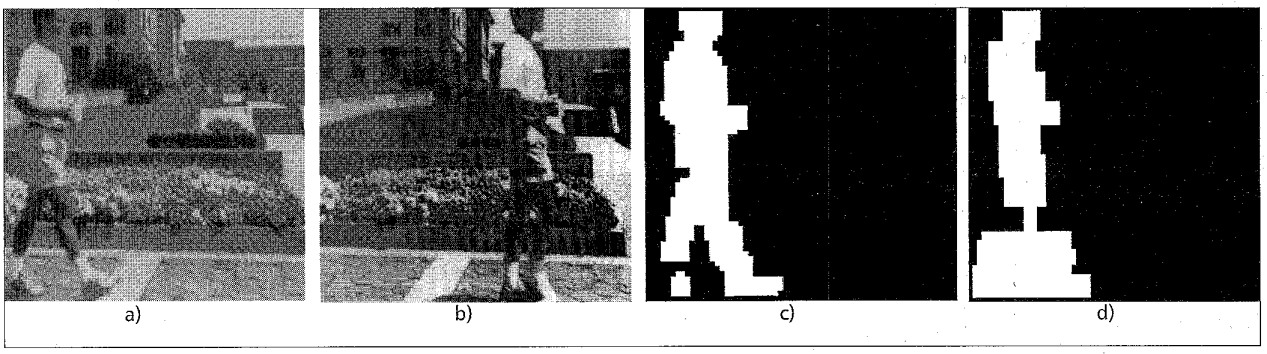


Figure 6. *Samir sequence: a) first frame I_1; b) last frame I_{30}; c) segmented template; d) tesselated template.*

is zero. As a result of this, the remaining velocities correspond to figure velocities.

4. Estimation of figure velocity. The estimation of figure velocities is done by recursively applying the algorithm in steps 1 to 3 (i.e., after compensating for the image background velocity).
5. Segmentation of independently moving figures. We apply, to two consecutive registered images (see step 3 above), a motion detection operator followed by a binary thresholding operation. The resulting (binary) image is then filled in to determine the area in the image corresponding to the moving figure.
6. Tesselation of figures. The filled-in image determined in step 5 is tesselated to determine a compact shape model.

The result of the above preprocessing operations is the ancillary data: image background velocity, figure velocities, figure (tesselated) shapes, image size, and image occlusion information.

Figure 6 illustrates the result of this algorithm when applied to the Samir sequence. This sequence is a short shot of 30 frames, of 1 s duration, showing an outdoor scene with a person moving across a static outdoor scene. Figures 6a and 6b show the first and last images of the original sequence. Figure 6c shows the segmented person which was obtained by applying the algorithm described above: first, registering two successive images with respect to the (small) image background velocity; second, applying a motion detection operator [8]; and, third, making the result of the detected image binary. Figure 6d shows the result of tesselating the person's figure. This tesselated template is a compact shape model representation for the figure.

Compact shape models as those illustrated by the template in Fig. 6d are obtained in GV by tesselations using rectangular shapes. The complex shape of a moving figure is reduced to a set of rectangles. Tesselating the shape of a figure by rectangles describes this shape with a parsimonious model. The fidelity with which the shape of the figure is described versus the model complexity (i.e., the number of rectangles) is determined by minimizing a cost function. This cost is the sum of two terms, one term given by the square difference between the original figure area and the area spanned by the tesselating rectangles, and the other corresponding to a penalty term proportional to the total number of rectangles.

GV Constraints

We discuss several issues relevant to GV in this subsection.

GV, as described above, has been applied to sequences that illustrate a variety of real live video conditions. It has been applied to sequences with only camera motion (Carnegie sequence), to sequences with a single moving object (Samir sequence), and to sequences with simultaneous camera motion and multiple moving objects (*Forbes sequence*, not illustrated here [12]). The Forbes sequence lasts about 20 s. It has significant camera motion as well as 12 moving cars, moving in and out of the sequence, with, in certain portions of the video, up to four stratified layers.

The compression ratios provided by GV increase with the length of the sequence. With longer shots like the Carnegie sequence (10 s) and the Forbes sequence (20 s), the compression ratios with GV are well in the range of 1000–10000 (see the "Compression" subsection). In comparison, MPEG's compression ratio, regardless of the length of the sequence, is typically below 100.

A relevant question is how often we should reset the GV representation (i.e., restart the world images generation). If there are significant changes in the background between successive frames, GV should be restarted. In a video shot, as we defined it at the beginning of the previous subsection, the background does not change radically from one frame to the next, so this is not a major concern here. This issue relates to the actual length of the video shots, the desired level of fidelity, the compression ratio required, and the level of delay that the application can support.

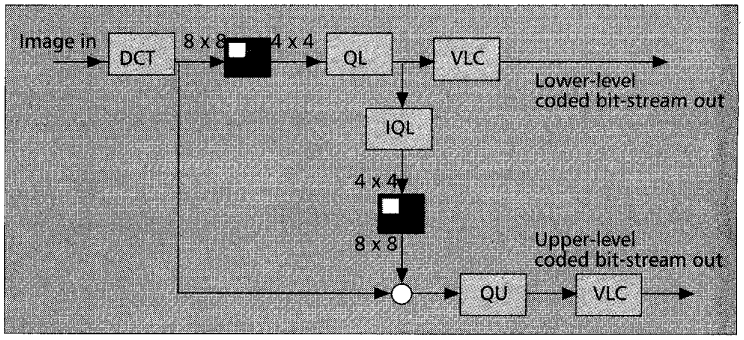
Although in this article we focus on video retrieval, there are many real-time applications where GV is quite useful. In video conferencing, remote classroom, or surveillance, the background and moving objects remain the same through long periods of time. In these applications, we distinguish two modes of operation. The first is the initial mode, a transient mode where the frames are essentially coded in small groups as in MPEG. However, after a few minutes either the source or the receiver can build world images and resort to the GV mode of tracking motions and variations of the moving objects.

The motion algorithm described in the previous subsection assumes that the figures are rigid objects. This assumption leads to good perceptual results in sequences where, for example, the figures are cars. In [12], we show the results of processing the Forbes sequence, which, as mentioned above, is a 20 s video shot in 600 frames where there are 12 cars moving in and out of the sequence coupled with camera panning motion. For this sequence, GV leads to a compression ratio of over 7000 with good perceptual quality. As shown in Fig. 6, we applied GV to the Samir sequence, where the figure, being a person, is clearly not rigid. In this case, if we treat the person as a rigid object, as we did in Fig. 6, GV introduces certain artifacts at the joints of the articulated parts (e.g., the knees). We are currently addressing this issue by incorporating into GV signal processing operators that can handle the motion of articulated rigid objects.

GV assumes that all motions are parallel to the image plane and that the outside world is two-dimensional. This is reasonable if the camera is relatively far from the scene being

recorded, or if the objects do move parallel to the camera. If not, this may constitute a limitation of GV as described in this article. In [13], we address this issue by considering three-dimensional visual information. This enables general three-dimensional motions. It requires that depth information be abstracted from the video data, either from stereo or motion, or directly by the availability of range data.

A final comment regards the assumptions on illumination and noise. Our current implementation assumes that the illumination conditions remain the same throughout the sequence and that the video is noiseless. With real live video, illumination conditions may change significantly from one frame to the next. What GV does is to equalize the intensity levels through histogram techniques. The played-back video may have an illumination level different from the original video, but the perceptual quality is not impaired. With the examples we have tested GV with, this issue and noise have not been significant.



■ **Figure 7.** *Frequency-scalable encoder.*

Scalable Spatial Compression: Block-Based DCT

Video compression schemes try to reduce the redundancy that exists across frames (temporal or interframe correlation) and within each frame (spatial or intraframe correlation). Since GV reduces the video sequence to a few still images — the world images — and ancillary data, there is no need for further temporal compression, only spatial compression. There are a number of techniques available. Due to its simplicity, we implemented a discrete cosine transform (DCT)-based spatial encoder. We provide details below.

In a heterogeneous network, the duplication effort involved in coding as many bit-streams as there are end-to-end throughput values is an important issue. We avoid this problem by developing a scalable implementation of the spatial encoder that produces hierarchical bit-streams like the one illustrated in Fig. 1. Decoding increasingly larger portions of the hierarchical bit-stream delivers increasingly higher resolution at an increasing bandwidth cost. Scalable coding enables access to the image or video at different resolution or quality levels through the same bit-stream.

We now detail our implementation of the spatial encoder.

Block-Based DCT – Block-based DCT is widely used in the JPEG and MPEG standards. A block-based DCT coder partitions the image into blocks. We choose 8 x 8 blocks in our experiments. In our implementation, we pad the world image to the right and bottom by pixel replication in order to have an integer number of 8 x 8 blocks. DCT is applied to each block independently. The DCT coefficients are normalized by an 8 x 8 Q matrix and rounded to integers (quantization). Usually, images are low-pass, so the higher-order DCT coefficients are much smaller than the lower-order coefficients. After the quantization step, many of the high-frequency DCT coefficients are rounded to zero. To take advantage of these zeros, zigzag scanning of the DCT coefficients is followed by variable-length coding (VLC) lossless compression (e.g., Huffman coding).

Scalability – We consider briefly three types of scalability, well discussed in the literature [14], that are directly applicable to block-based DCT coding: frequency scalability; signal-to-noise ratio (SNR) scalability; and spatial scalability.

Frequency Scalability – Figure 7 shows a two-level frequency-scalable encoder. The top path encodes the lower-class bits, while the bottom path encodes the additional high-resolution bits. Following the 8 x 8 block DCT, only the lower-frequency 4 x 4 upper left corners of the DCT coefficients are kept, quantized, and VLC-encoded. The output of the top quantizer is then input to an inverse quantizer, reimmersed into an 8 x 8 block, and compared to the original 8 x 8 DCT block. The difference is quantized and VLC-encoded. The outputs of the two VLC blocks are the low-resolution and high-resolution portions of the hierarchical bit-stream.

SNR Scalability – This is similar to the frequency-scalable encoder of Fig. 7, except that the top path uses a coarse quantizer Q_L to quantize the full 8 x 8 block, not the 4 x 4 block as in Fig. 7, while the bottom path applies a finer quantizer, Q_U.

Spatial Scalability – Again, the block diagram of this encoder is similar to Fig. 7, except that its front-end is a subsampler. The encoder combines spatial subsampling of the original world image with a scalable DCT coder. The world image is initially subsampled; the subsampled image is then DCT-encoded and quantized. These are the low-resolution bits. The quantized low-resolution bits are inverse-quantized, inverse-DCT, upsampled, and subtracted from the original image. The residual is then transformed with the DCT, quantized, and VLC-encoded. These are the high-resolution bits that form, with the low-resolution bits, the data in the hierarchical packet.

We have implemented the three scalable quantizers. We borrowed, after some adaptation, the Q matrices from the MPEG-2 video standard, [15]. We also used, with a slight modification, MPEG-2's Huffman coding tables B-12, B-15, and B-14. Scalable decoding is accomplished by reversing the encoder, in particular, the block diagram in Fig. 7.

Compression

We discuss the compression potential of scalable GV in the context of the Carnegie sequence. The GV representation for this sequence is reduced to the background world image in Fig. 4, and the world image and the camera motion information in Fig. 5. In other words, GV reduces the 300 frames of the video shot to a single still image plus ancillary data. GV resynthesizes the sequence by sliding a rectangular window — the *image window operator* — over the world image. The size of the image window operator is N_x^I x N_y^I (i.e., the size of each original frame); its motion is given by the camera velocity. The background world image and the ancillary data are what we need to transmit to the client for the decoder to be able to reconstruct a version of the original sequence.

Video compression usually has temporal or interframe compression and spatial or intraframe compression. In GV, temporal compression corresponds to reducing the original

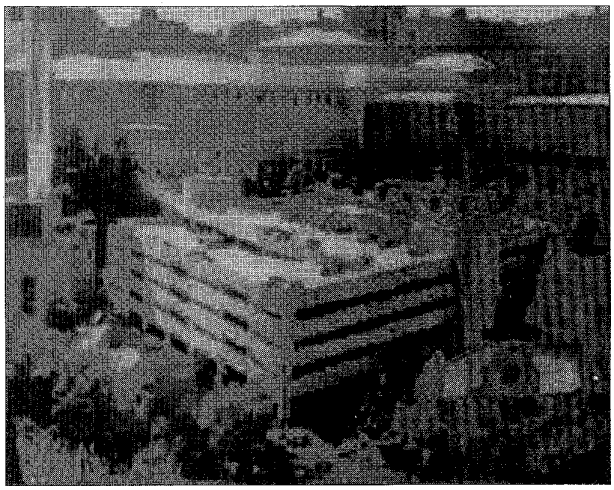


Figure 8. *Low-resolution background world image by frequency-scalable encoder with 22.15 spatial compression ratio.*

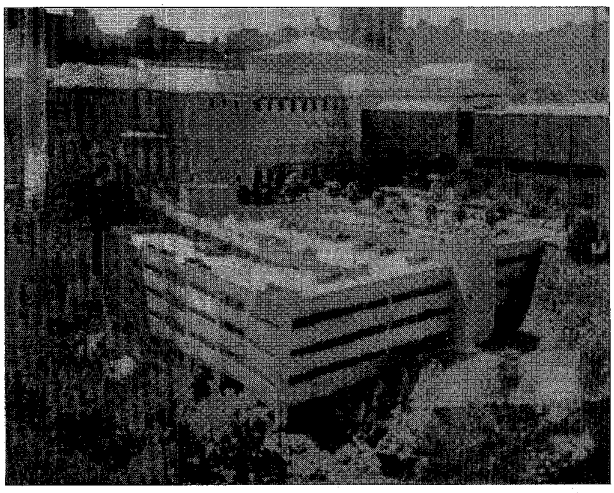


Figure 9. *High-resolution world image by frequency-scalable encoder with 1.45 spatial compresion ratio.*

sequence to the world images. The world images eliminate the temporal redundancy across the frames. This property, by itself, achieves a significant temporal "lossless" compression ratio $\mathcal{C}_t$. For the Carnegie sequence, this is

$$\mathcal{C}_t = \frac{N_x^{\Phi} \cdot N_y^{\Phi}}{N_y^I \cdot N_y^I \cdot N_r} = 106,$$

given that $N_x^{\Phi} = 376$ pixels (p), $N_y^{\Phi} = 462$ p, $N_x^I = 240$ p, $N_y^I = 256$ p, and $N_r = 300$ frames.

In addition to the temporal compression due to world image generation, the world images, being still images, can be further compressed before transmission by exploiting their spatial redundancy. We can use any intraframe coding technique to spatially compress the world image, for example, a wavelet based compression algorithm or a DCT method. In [7], we applied a noncausal predictor with vector quantization (NCP-VQ) scheme [16, 17]. We compressed the world image with the technique described in [17], achieving a compression ratio of 42.5 with acceptable quality. This led to an overall compression ratio of about 4500.

NCP-VQ decoding is more complex than block based DCT decoding. Since we need a simple decoding algorithm for low-performance mobile computing platforms, we choose to implement the scalable block DCT coder. Block-based DCT coding does not provide spatial compression as large as NCP-VQ at the same visual quality. At compression ratios above 8 or 10, block-based DCT introduces noticeable blocking artifacts. We discuss next our results with scalable block-based DCT compression of GV. In Fig. 8 we display the low-resolution decoded background world image, and in Fig. 9 the high-resolution decoded background world image, after compression by the frequency-scalable encoder of Fig. 7. The spatial compression ratios are 22.15 and 1.45, respectively. The total compression ratios for the Carnegie sequence after multiplying the time and spatial compression factors are then 2350 and 150, respectively.

The original Carnegie sequence has 300 frames, each 240 x 256 pixels, 8 b/pixel. This represents 147.5 Mb of data. To transmit this video clip over CDPD at 10 kb/s requires about 4 hr. With a compression ratio of 2350, the compressed world image is 63 kb. At the same throughput of 10 kb/s, the transmission of the GV scalable coded 10 s Carnegie sequence, ignoring associated overheads, takes about 6 s to transmit. The higher-resolution sequence, with total compression ratio of 150, takes about 100 s.

Figure 10 shows the low and high resolution of the tenth image in the Carnegie sequence after compression by a factor of 2350 and 150, respectively. Blocking effects and loss of detail (e.g., in the walls of the Carnegie Museum building at the back of the world image) are apparent in the low-resolution world image in Fig. 8 and in the low-resolution image I_{10} in Fig. 10. However, when displaying at the frame rate of 30 frames/s, the regenerated sequence still provides a quality that can be judged acceptable.

The other two scalable coders deliver similar images. Due to lack of space we do not display the images. We note that with spatial scalability the low-resolution image is one-fourth its original size.

The peak-to-SNR (PSNR) was computed for the background world image and for the individual frames after compression with each of the three scalable coders. The PSNR values are quite consistent across the three coders, set at about 44 dB for the high-resolution images and 21 dB for the low-resolution images. The spatial scalability coder had a slight degradation of 0.3 dB in the high-resolution images with respect to the other two.

The ancillary data is lossless-encoded. For the Carnegie sequence, the 300 frames require 300 motion vectors. Coding the two components of each vector with 1 byte leads to a total of 2400 b, which is less than 4 percent of the total required by the coded background world image. Lossless compression reduces this further. In the next section, we explain that, in fact, our video communication system treats these two types of data differently: the world images as off-line and the ancillary data as real-time data.

Video System Architecture

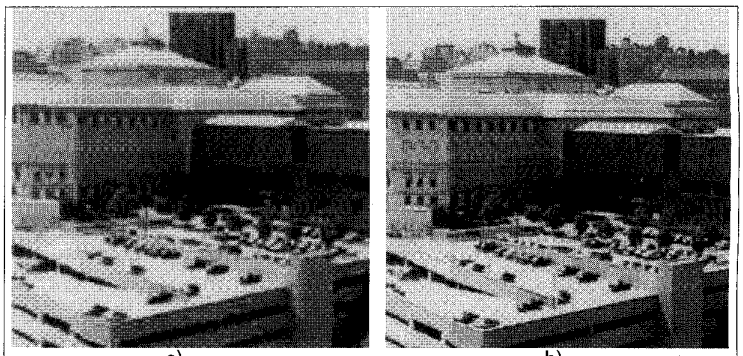
This section addresses design issues and the system architecture implementation, and analyzes optimization questions.

Design Issues

We consider the following design issues.

Real Time – Ideally, a video communication system should be real-time. This, however, has stringent implications in terms of technology requirements. Real-time performance requires the ability to reserve computing as well as network resources. With dedicated T1 line, integrated services digital network (ISDN), or fiber distributed data interface, version 2 (FDDI-II) with isochronous transmission mode, we can guarantee the

required bandwidth throughout the connection. These networks guarantee the throughput and provide stable delay. Also, dedicated hardware or real-time operating systems can be employed to reserve the central processing unit (CPU) resources for video processing. Our goal, however, is different: to design and implement a system that handles video and uses widely deployed technology — the heterogeneous environment described earlier with Ethernet, CDPD, and wireless LANs, and PC- and UNIX-based platforms. All of these networks use contention-based multiple access schemes in the MAC layer (Table 1). These are not real-time protocols; DOS or Windows and UNIX are not real-time operating systems, either. Therefore, we loosen the real-time timing constraints by relaxing the error requirements and allowing for frame-skipping with optimized buffering strategies, as explained below.



■ **Figure 10.** *a) Low-resolution (compresion 2350); b) high-resolution (compression 150) tenth image I_{10} by GV with frequency-scalable DCT.*

Video Representation – We use the GV representation presented previously. This representation reduces video to world images (background and moving objects) and ancillary data (motions and occlusion). The world images are treated as still images. They are transmitted after a step of hierarchical lossy scalable coding, followed by lossless compression. In general, there are N hierarchical classes. A destination receiving class i will receive and process all classes $j, j = 1, ..., i$. The higher the class, the higher the resolution of the displayed sequence, the lower the compression, the higher the data rate required for transmission. The ancillary data is transmitted subject only to lossless compression.

Regarding transmission real-time constraints, we note that video retrieval is not strictly real-time. Data transmission can take as much time as is acceptable to the user. However, once started, redisplaying is real-time and, as such, does have hard timing constraints. This leads us to deal differently with the encoded world images and the ancillary data. The bulk of the data is the encoded world images. We transmit the world images in the first phase (see the following subsection) off-line, with no real-time constraints. On the other hand, the ancillary data, which represents the motion observed between two successive frames, is sent within real-time constraints, every 33.4 ms, which corresponds to the video real-time rate. We could send the ancillary data off-line. We choose to transmit it in real time, avoiding, in longer sequences, the extra delay of off-line transmission of the ancillary data. This extra delay could be significant if the number of figures is large and the sequence long. For the Forbes sequence, with camera motion and 12 figures, the 600 frames would lead to an overhead of 62.4 kb. This is a significant fraction of the number of bits taken by the world image. The world image for this sequence, after GV scalable compression by a factor of 4550, is 79 kb.

Graceful Degradation – The video system should degrade gracefully in the presence of errors, unexpected changes in operating conditions such as overload and network congestion, or abrupt variations in the end-to-end throughput due to client mobility. This requirement has different impacts on the two types of data in the GV coded representation. World image packets represent the critical data in terms of overall quality: errors in the world images persist throughout the whole sequence; and, because world images are highly spatially compressed, errors may cause unacceptable degradation of the video quality. On the other hand, the ancillary data is lossless-compressed, and errors do not propagate across the sequence. Therefore, the world image compressed data should be transmitted without errors, while the transmission of the ancillary data can sustain a certain number of errors.

Transport Protocol: TCP/IP and UDP/IP – From the above considerations on real time and graceful degradation, we choose to transfer the world image data with TCP/IP and the ancillary packets with UDP/IP. TCP is a reliable transport protocol using packet retransmission to guarantee error-free communication. This, however, is prone to delays which do not affect the world image transmission given its soft time constraints. With the ancillary data, we use the UDP protocol. UDP does not support retransmission, so it avoids the additional delays introduced by packet retransmission.

QoS Requirements – We introduce two QoS parameters: the maximum time period the user is willing to wait for the beginning of the video playback, T_w, and the frame rate, F_r. It is conceivable that a client interrogating a video server with a low-speed link, say CDPD, is willing to wait longer, even if to receive a lower-quality video clip, than a client on a wideband connection, where the expectations are usually much higher in terms of both delivered quality and speed of service. Parameter T_w takes care of this consideration. It is particularly relevant with video applications in heterogeneous networks due to the wide range of end-to-end throughput rates available to different clients and the dramatic swings in available bandwidth experienced by mobile users. The parameter T_w and the end-to-end throughput R_n determine the total amount of data to be delivered to the client, and so determine the hierarchical video class to be delivered to the client (see the previous section). Parameters T_w and R_n are updated dynamically, since, over time, mobile users will shift from narrowband to wideband wireless and back, with over two orders of magnitude variation in throughput, implying two orders of magnitude change in transmission waiting time.

The parameter F_r indicates the CPU resources needed to process the video request since its inverse gives the maximum time available to compute and display a frame. A slow machine requires more time to perform these tasks, and so may need to skip every other frame.

Architecture Implementation

Our implementation of the video retrieval system contains three components: the video servers, the video directory service, and the clients. These three components are interconnected via a heterogeneous network.

The video servers store the video repository to be retrieved by the clients. The world images are hierarchically coded in multiple classes (see the previous section). The hierarchical

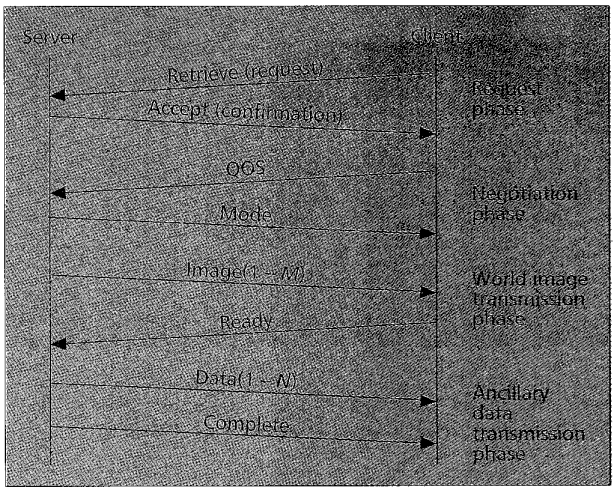


Figure 11. *Video server/client interaction procedure.*

classes to be retrieved and transmitted are determined by the current QoS parameters of the client and the network throughput between the server and the client. These classes are transmitted through TCP/IP channels, one for each hierarchical class. The world images are transmitted first, followed by the transmission of the ancillary data through a UDP/IP channel. As an alternative to having multiple TCP/IP channels to transmit the world images, we could multiplex the data from the same DCT blocks from different hierarchical classes into the same TCP/IP channel. This multiplexing requires strict synchronization when recovering the data so that the multiplexed data corresponds to the same block. We have not implemented this variation.

The video directory service cross-references the video clip's information with the network addresses of the video servers. The video directory service address and port are known to all clients. A client retrieves the address of the video server that stores the specific video clip by querying the video directory service.

The clients retrieve video clips stored in the video servers, reconstruct the video sequences from the world images and ancillary data, and display them. A client sends its QoS requirement parameters to the video server prior to requesting the world images. The video server replies with the number of hierarchical classes it is going to transmit. The client connects that number of TCP/IP channels to the video server for transmission of the world images. After receiving the world images, the receiver establishes a UDP/IP channel for the ancillary data. The client reconstructs and displays the video clip while receiving the ancillary data.

Video Server/Client Interaction Procedure – The time line of the video server/client interaction is shown in Fig. 11. There are four phases, starting with the request phase and ending with the completion of the retrieval.

- Request phase — A client requests the server to transmit a video sequence. Assuming no errors, the server responds with a confirmation accepting the request.
- Negotiation Phase — In the next step, the client informs the server of its QoS parameters: T_w, the time that the client is willing to wait for the beginning of replaying, and F_r, the frame rate. The server determines the video quality class from the QoS parameters specified by the user and the network throughput information maintained by the server. The server then sends the Mode information, which specifies the video quality class, and the number and size of the world images.
- World image transmission phase — The server transmits the world images. As explained in the previous subsection, we use TCP.
- Ancillary data transmission phase — After receiving and decoding all world images, the client sends a Ready message requesting of the server the ancillary data. The server sends this data every $(33.4 \times 30)/F_r$ ms. In this phase, we use UDP. When the server transmits the ancillary data for the last frame, it sends a Completion message to the client, and the session is closed.

Optimization

In this subsection, we discuss several trade-offs that need to be balanced in order for the video communication system to perform well. We discuss buffering policy, video resynchronization, and concurrency.

Buffering Policy – Redisplaying the regenerated video and transmitting the ancillary data impose real-time timing constraints. However, the underlying networks do not necessarily have mechanisms guaranteeing minimum throughput, maximum delay, or maximum delay jitter; nor do the machines's operating systems allocate the CPU resources with precedence to guarantee these timing constraints. Buffers allocated in the data path can absorb timing mismatches and help the system to degrade gracefully. The size of the buffers associated with the UDP/IP channel affects the quality of the displayed video clip. Larger buffer sizes may increase the delay when the client experiences an overload condition or the network is congested. The accumulation of these delays may become unacceptable to users. Smaller buffers, on the other hand, may cause the UDP/IP connection to discard ancillary packets more frequently. There is a clear trade-off between the delay and frame drops. With video systems, it is preferable to drop frames rather than experience longer delays. We show next that the buffers can be optimized to achieve a reasonable compromise between these two conflicting requirements.

Figure 12a illustrates how the delay varies as a function of the load of the client. The inverse of the load specifies the fraction of the client's computing resources dedicated to the decoding and redisplaying of the video sequence. Load is normalized by $30/F_r$. The parameter F_r is the frame rate at which the client can display the video clip under a given load level. The upper curve in Fig. 12a shows the delay experienced by the 200th frame of the Carnegie sequence, while the middle line shows the delay for the 100th frame, when we use the default sizes of the buffers set by the UNIX operating system. These curves indicate that the delay increases monotonically with the load and may become very large, beyond what is acceptable for video. The lower curve, which is actually two curves that are practically indistinguishable, shows the delay experienced by the same 100th and 200th frames when the buffer sizes are reduced to the optimized values chosen by us. The plot shows that the delay is kept under control below a maximum acceptable value.

We now address the issue of frame skips for the optimized values of the frame buffers. To achieve graceful degradation for a constant load level, the number of frame skips should stay on average constant through the video sequence.

Figure 12b shows the distribution of frame skips in the Carnegie sequence under different load levels. The upper solid line, the middle dashed line, and the lower solid line show the occurrences of frame drops under load levels of 8, 4, and 2, respectively. These curves show that, as desired, with the user-set buffer sizes the number of frames skipped stays relatively constant throughout the sequence.

Video Resynchronization – Video resynchronization retains the original video timing of (33.4 x $30/F_r$) ms/frame displayed at the client. We discuss two alternatives:

- Independent clocks — In this scheme, the server and client have independent clocks. They process tasks based on their individual clocks. This method provides relatively accurate timing since the display timing does not depend on the uncertainties associated with the packets delivery timing. On the other hand, this requires a clock synchronization mechanism between the server and the client. In addition, handling the delay jitters and frame skips is complicated.
- Dependency on the server timing — The client follows the server's timing, provided as the input packet rate. The major advantage of this scheme is simplicity. Clients easily handle frame skips due to overload and network congestion. The reconstructed video timing fluctuates with the delay jitters of the network.

We adopted the server timing slavery approach. In the absence of network congestion, this mechanism works well and degrades gracefully under overload conditions.

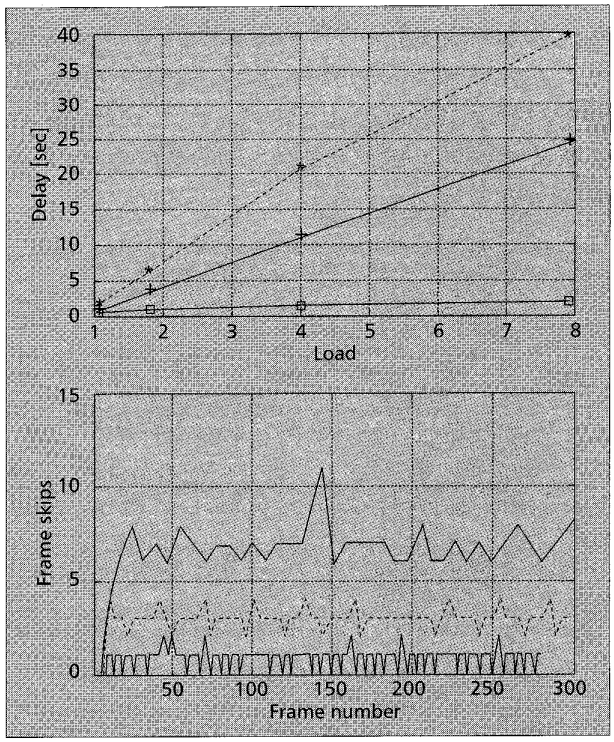
Concurrency and Shared Memory – To improve performance, we implemented the following:

- Concurrency — At the client, the communications manager module, which receives packets and puts them into an input buffer, and the video decoder, which gets data from the input buffer and decodes the world images or reconstructs the frames, are implemented separately into two processes that work concurrently. At the server, the video service retrieval, which retrieves the data files, and the communication manager module, which transmits the data, are implemented as two separate processes that work concurrently.
- Interprocess communication — We utilize the shared memory mechanism provided by UNIX for interprocess communication (IPC) in the same machine. Shared memory supports high-performance IPC by avoiding additional copies, which are required when using other IPC mechanisms such as pipelining.

Conclusion

This article has addressed the issues arising when transmitting video across wireless and heterogeneous networks. Key factors to consider in this application are the large volumes of raw data created by the video sources, in contrast with the wide range of network throughputs, the dramatic and abrupt changes in these throughputs experienced by nomadic users, and the relatively low performance of portable machines. These factors combine to conflicting requirements: aggressive compression ratios of 1000–10000 with simple decoding algorithms.

The main contribution of the article is the presentation of Generative Video, a content-based video meta representation that reduces video sequences to a small number of still images (world images) and side information (ancillary data). The GV representation is particularly suited to the wireless environment: it has the potential of large compression ratios at acceptable perceptual quality, it is easily scalable, and it is an asymmetric encoding/ decoding algorithm. Decoding is trivial. We have described a hierarchical encoding by simply scalable-DCT-encoding the world images. Scalable-coded GV is easily decoded in real time. Our experimental results demonstrate that GV delivers overall compression ratios in the desired range of 1000–10000 while preserving the visual quality at acceptable subjective levels. With VHS digitized video at a frame rate of 30 frames/s and frames with over 100 kpixels, the raw video data rate is in excess of 72 Mb/s. Under GV, with a compression ratio of 2500, this is reduced to 28.8 kb/s, which is in almost real time with CDPD. We have presented the overall system architecture and the pertinent trade-offs, showing how hard real-time timing and error constraints can be handled with networks and operating systems that do not provide for resource reservation control mechanisms.



■ **Figure 12.** *a) Load versus delay for default buffer size; b) number of frames skipped versus frame number for optimized frame buffers.*

In the very near future, we will replace the IP protocol with the Mobile IP protocol described in [18], affording clients automatic handoff capability and true roaming ability. Header information will provide the video application process with the user's current QoS parameters. This will enable the video server to dynamically adjust the user hierarchical class level. We are currently applying GV to other video applications, in particular to nonlinear video editing, where video is manipulated and addressed by its contents, [19].

References

[1] A. Hills and D. B. Johnson, "Wireless Data Network Infrastructure at Carnegie Mellon," *IEEE Pers. Commun.*, this issue.

[2] R. S. Jasinschi, GV video compression tape demos; three sequences: Carnegie sequence, May 1994; Samir sequence, July 1994; and Forbes sequence, Sept. 1995.

[3] A. N. Netravali and B. G. Haskell, *Digital Pictures, Representation, Compression and Standards*, [Plenum Press, New York, 2nd ed., 1995].

[4] T. Chiang and D. Anastassiou, "Hierarchical Coding of Digital Television," *IEEE Commun. Mag.*, vol. 32, no. 5, May 1994, pp. 38–45.

[5] R. S. Jasinschi and J. M. F. Moura, "Representing Global Motion on Lattices: Model-Based Image Sequence Analysis," Tech. rep., Dept. of Electrical and Comp. Eng., Carnegie Mellon Univ., Pittsburgh, PA, Oct. 1993.

[6] R. S. Jasinschi and J. M. F. Moura, "Generative Video," Tech. rep., Dept. of Electrical Eng., Carnegie Mellon Univ., Pittsburgh, PA, July 1994.

[7] R. S. Jasinschi *et al.*, "Video Compression via Constructs," *ICASSP '95, Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing*, Detroit, MI, vol. 4, May, 8–12 1995, pp. 2165–68.

[8] R. S. Jasinschi and J. M. F. Moura, "Content-Based Video Sequence Representation," *Proc. IEEE Int'l. Conf. on Image Processing, ICIP '95*, Cristal City, VA, vol. 2, Oct., 23–26, 1995, pp. 229–32.

[9] P. Anandan *et al.*, "Video as an Image Data Source: Efficient Represen-

The GV representation is particularly suited to the wireless environment: it has the potential of large compression ratios at acceptable perceptual quality, it is easily scalable, and it is an asymmetric encoding/decoding algorithm.

tations and Applications," *ICIP '95, Proc. IEEE Int'l. Conf. on Image Processing*, Cristal City, VA, vol. 1, Oct. 23–26 1995, pp. 318–21.

[10] P. J. Burt and P. Anandan, "Image Stabilization by Registration to a Reference Mosaic," Technical reports of the ARPA Image Understanding Workshop, SISTO, ARPA, Monterey, CA, 1994.

[11] J. Y. A. Wang and E. H. Adelson, "Representing Moving Images with Layers," *IEEE Trans. on Image Processing*, vol. 3, no. 5, Sept. 1994, pp. 625–38.

[12] R. S. Jasinschi, "Generative Video: A Meta Video Representation," Ph.D. thesis, Electrical and Comp. Eng. Dept., Carnegie Mellon Univ., Pittsburgh, PA, Sept. 1995.

[13] F. Martins and J. M. F. Moura, "3-D Video Compositing: Towards a Compact Representation for Video Sequences," *Proc. IEEE Int'l. Conf. on Image Processing, ICIP '95*, Cristal City, VA, vol. 1, Oct. 23–26 1995, pp. 550–53.

[14] N. D. Wells and P. N. Tudor, "Standardization of Scalable Coding Schemes," Chris Toumazou, ed., "Circuits and Systems Tutorials," Ch. 3.12, *Proc. IEEE ISCAS '94*, 1994, pp. 121–30.

[15] MPEG software simulation group, an independent research group headed by S. Eckert and C. Fogg, web page, html://ftp.netcom.com/pub/cf/cfogg.

[16] N. Balram and J. M. F. Moura, "Noncausal Predictive Image Codec," *IEEE Trans. on Image Processing*, to be published, 1996.

[17] A. Asif and J. M. F. Moura, "Image Codec with Noncausal Prediction, Residual Mean Removal, and Cascaded VQ," *IEEE Trans. on Video Tech.*, vol. 6, no. 1, Feb. 1996.

[18] D. B. Johnson and D. A. Maltz, "Protocols for Adaptive Wireless and Mobile Networking," *IEEE Pers. Commun.*, this issue.

[19] R. S. Jasinschi and J. M. F. Moura, "Nonlinear Editing by Generative Video," *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing, ICASSP '96*, Atlanta, GA, May 1996, accepted for publication.

Biographies

JOSÉ M. F. MOURA [S '71, M '75, SM '90, F '94] received the Engenheiro Electrotécnico degree in 1969 from Instituto Superior Técnico (IST), Lisbon, Portugal, and the M.Sc. in electrical engineering and D.Sc. in electrical engineering and computer science from the Masachusetts Institute of Technology (MIT), Cambridge, in 1973 and 1975, respectively. He is presently a professor of electrical and computer engineering at Carnegie Mellon University, Pittsburgh, Pennsylvania, which he joined in 1986. Prior to this, he was on the faculty of IST, where he was an assistant professor (1975), professor agregado (1978), and professor catedrático (1979). He has had visiting appointments at several institutions, including MIT and the University of Southern California.

RADU S. JASINSCHI received the B.S., M.S., and Ph.D. degrees in physics from the University of São Paulo, Brazil, in 1975, 1978, and 1984, respectively. He was awarded the degree of Ph.D. in electrical and computer engineering from Carnegie Mellon University in 1995. He was a post-doctoral fellow in physics at the Lyman Laboratory, Harvard University, from 1994 to 1996. He was a visiting scientist at the Robotics Institute from 1986 to 1988, and an assistant research scientist at the Center for Automation Research at the University of Maryland from 1988 to 1992. He is currently a lecturer in the Department of Electrical and Computer Engineering at Carnegie Mellon University.

HIROHISA SHIOJIRI was born in Tokyo, Japan, in 1962. He received the B.S. degree in electrical engineering from Tohoku University in 1984. In the same year he joined the Second Transmission Division of NEC Corporation, Kanagawa, Japan. At NEC he was involved in the development of image processing and video compression equipment. Since 1994 he has been studying at Carnegie Melon University, supported by NEC Corporation, where his main interest is video communication technology. In December 1995 he received the M.S. degree in information networking from Carnegie Mellon.

JYH-CHERNG LIN received the B.S. degree in electrical engineering and the M.B.A. degree from National Taiwan University, Taipei, Taiwan, in 1990 and 1993, respectively. He was previously with the Service Orders Processing Systems Data Center at the Northern Taiwan Telecommunications Administration. He is currently pursuing the M.S. degree in information networking at Carnegie Mellon University.