Real-Time Transmission and Software Decompression of Digital Video in a Workstation

K. Umemura[1] and A. Okazaki

TR-91-004

January, 1991

Abstract

This paper describes an experiment in which compressed video data is transferred via Ethernet to a workstation, and uncompressed and displayed on the workstation. The workstation has no special hardware. The video data is 192x114 pixel gray scale, 30 frames per second. The data consists of a human speaker with a static background. It is displayed on monochrome display, with dithering, in a 768x576 rectangle. This decompression and display uses about 10 MIPS. The quality of output is suitable for applications such as conferencing, telephony, and presentations.

---

[1]Permanent address: Nippon Telegraph and Telephone Corporation, NTT Software Laboratories, 3-9-11 Midori, Musashino Tokyo 180, Japan.

1. INTRODUCTION

Digital audio and video have come to play an important role for computer applications such as computer supported cooperative work. Digital video boards are being developed for various workstations. Increasing the power and capacity of computers enables them to handle video information. The age of digital video is about to start.

Not all workstations, however, will have the digital video boards. A method that enables us to receive digital video on regular workstation may enable digital video applications to gain popularity and increase their market. In this paper, we describe an experiment to receive video information from a computer network using a workstation without extra hardware. The overall structure of this experiment is described in Fig. 1.

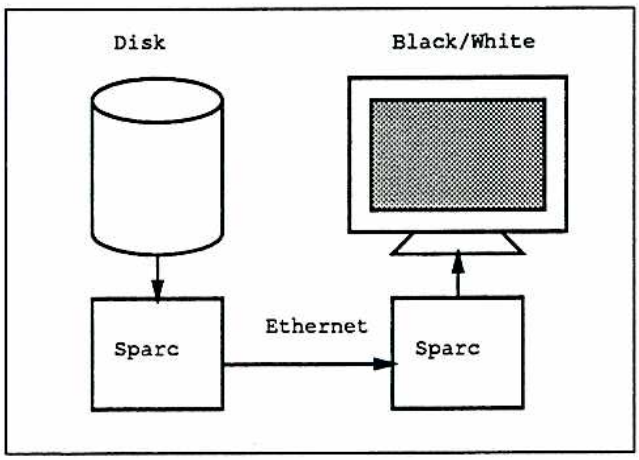

Fig. 1: The overall structure of this experiment.

Compression schemes have a tradeoff between data reduction and algorithmic complexity. We sought a compression scheme such that 1) the data rate is low enough to transmit on a TCP connection over an Ethernet, and 2) decompression and display can be done in software in real time on a workstation.

This paper describes such a compression scheme, and discusses the performance of its implementation. We have attained a quality that is suitable to communication of video speech data, and we are able to transfer and display such data in real time without special hardware using a Sun SparcStation.

2. COMPRESSION METHOD

2.1. Intra-Frame Compression

For the first stage, we have chosen a well known technique called *block truncation compression* [1]. In this technique, a tolerance X is chosen. We regard a given region (or "tile") as uniformly colored if the variance of the image color in the area is smaller than X. If the variance is greater than X, we divide the tile into smaller "subtiles" and do the same operation. Our implementation of this method uses square tiles; a square is subdivided into four smaller squares. The effect of this compression is shown in Fig. 2.

For example, the frame in Fig. 2 is expressed in 3285 bytes; the amount of data is 13% of original image and 0.95 bit/pixel. The regularized variance threshold value is 0.03125 (1/32) in this example. The original gray scale data is in the range 0 to 255, so if the variance in a square is smaller than 8.0 (256*0.03125), the square is regarded as uniformly colored.

Fig. 2: Result of block truncation.

2.2. Inter-Frame Compression

For the second stage, we have chosen another well known technique called *difference compression* [2]. In this technique, each frame is compared with the previous frame, and areas that have less difference in value than the threshold value are regarded as transparent. This avoids sending parts of the video image that are identical between two successive frames.

If a frame is very similar to the previous frame, it can be expressed using a few large transparent tiles. If the average size of the tile becomes bigger than in stage 1, the amount of data decreases. If this condition does not hold (in other words, there are many changes between frames) the result of this stage may be bigger than the previous stage. In this case, difference compression is suppressed and the data from stage 1 is used as it is.

An example of this compression is shown in Fig. 3 The regularized difference threshold value is 0.03125 (1/32). This frame is expressed in 2493 bytes; the amount of data is 9.8% of the original image and 0.72 bit/pixel. It contains 876 non-transparent blocks. Though the amount of data is not reduced much (77% of the previous stage) the number of drawing operation to display is 27% of the previous stage; the area to be updated is 4% of the entire image. This reduces the

cost to draw this frame.

Fig. 3: Result of difference compression

3. IMPLEMENTATION

3.1. The Compressed Video Format

The image data is a sequence of frames. Each frame consists of a header and the sequence of units. The header has (1) format identifier, (2) length of frame in bytes, (3) relative time from start in milliseconds and (4) size of frame.

Each "unit" is a frame is describe by a single byte. The upper 2 bits encode the size of the square and the lower 6 bits encode the gray value. The position of the tile is implicit from the byte's position within the frame. The size value 0 denotes a 1x1 square, 1 denotes a 2x2 square, 2 denotes a 4x4 square, 3 denotes an 8x8 square. The color value 0 is transparent, the color value 1 is black and the color value 63 is white. The color values between 2 and 62 are gray. The color value 0 is generated by the difference compression. In other words, the description of each frame is a collection of tiles of 4 sizes and 64 colors.

4. The Decompression Algorithm

The task of the decompression routine is to draw gray tiles in their proper places. The algorithm is defined as a recursive procedure that takes a size parameter. If the current tile is smaller than size, the routine calls itself with the smaller size. If the current tile matches the size, a

graphics routine is called to display the tile. The algorithm is defined in Fig. 4. In the actual program, the recursive calls are expanded for efficiency.

```
int level;
int value;
unsigned char * valp;

decompress(r, c, cl)
        register int r, c, cl;
{

        if (level < cl) {
                decompress(r, c, cl/2);
                decompress(r+cl/2, c, cl/2);
                decompress(r, c+cl/2, cl/2);
                decompress(r+cl/2, c+cl/2, cl/2);
        } else {

              if(value>0) /* if tile is not transparent */
                  WriteRectangle(value, /* color */
                                 c,     /* x coordinate */
                                 r,     /* y coordinate */
                                 cl,    /* x width */
                                 cl     /* y width */
                                 );
                value = *valp++;
                level = 2 << (value >> 6);
              value &= 0x3f;
        }
}

decompress_frame()
{ int n;
  register int r, c;
  valp = compress_value_array;
  value = *valp++;
  level = 2 << (value >> 6);
  value &= 0x3f;
  for (r = 0; r < IMR; r += BLK) for (c = 0; c < IMC; c += BLK) {
      decompress(r, c, BLK);
  };
}
```

Fig. 4: Decompression Algorithm.

5. PEFORMANCE MEASUREMENTS

5.1. Effects of Video Data Characteristics

We have tried two sets of data, both of which show a person talking. One, called Miss America, has simple background (See Fig. 2). The other, called Salesman, has many objects in the background, see Fig. 5. These data were also compressed with the variance threshold value 0.03125 (1/32).

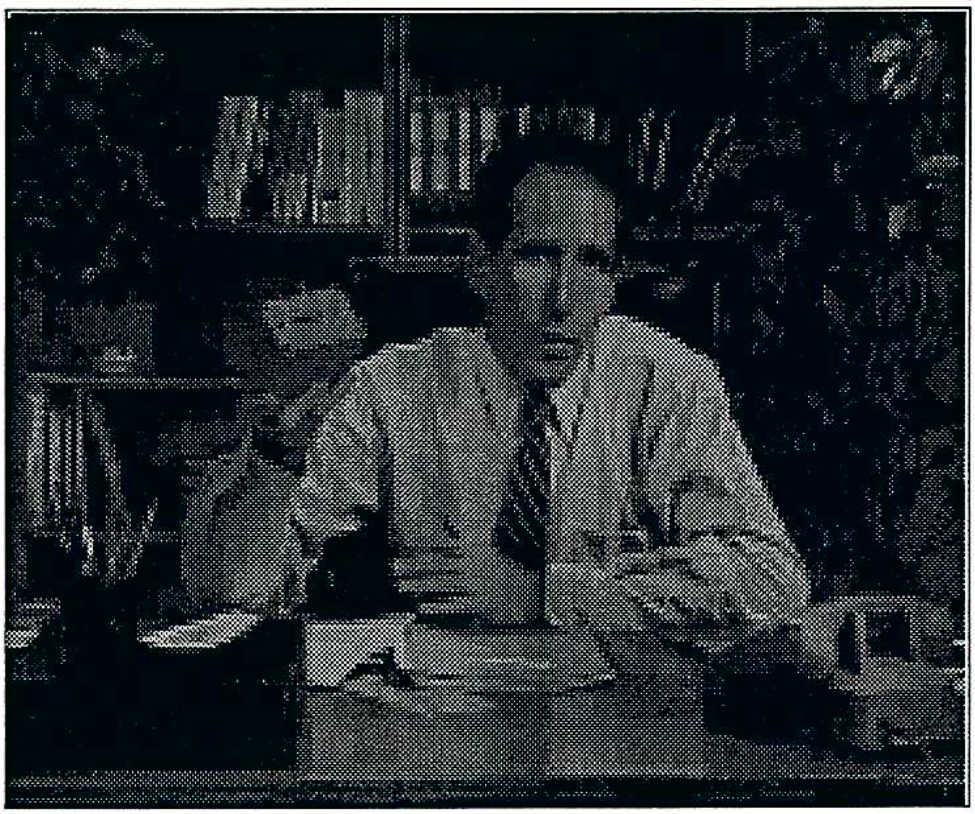

Fig. 5: Video data with complex background.

This compressed data is efficient to display. Drawing filled rectangles is the only graphic routine used to display frames. It is a fundamental operation in many window systems, and it is carefully implemented to make window system fast; some systems, which is not intended to display video, have hardware to make tile filling faster. Since filling a rectangle is simpler to implement than drawing lines or writing character strings, it is reasonable to rebuild graphic routines for fast drawing of rectangles. In addition, transparent tiles do not cause graphic operations. The region that is not changed much will have the transparent color as the result of the difference compression. In other words, only the changed region is rewritten by the graphic operations. This also makes drawing frames efficient.

We measured the maximum rate at which frames can be transferred and displayed. The data was measured using a Sun Sparcstation 1+, with Monochrome display and 8 MB of memory. The processor is roughly 16 MIPS. No additional boards are used. We built a dedicated square-filling function to display video. This graphic routine bypasses the window system and writes the

data on the frame buffer.

For the measurements, the video data is stored on a remote machine, transferred by TCP/IP, and then decompressed, halftoned and displayed. The structure is shown in Fig. 1. During this experiment, the load average on the remote machine is less than 0.60, while the local machine is fully loaded. The variance and difference thresholds are both 0.03125 (1/32). The value is the middle value of three trials.

We measured the data without difference compression for the pessimistic case, in which data changes in every frame. The bandwidth is the average amount of data for 30 frames; that is 1 second. Speed is measured in frames per second. Data ratio is measure in kilobytes per second.

Condition	Speed	Data rate	Bandwidth
Miss America	72.0(frame/sec)	225.(KB/sec)	93.(KB/sec)
Salesman	50.4(frame/sec)	221.(KB/sec)	131.(KB/sec)
Miss America / Difference off	37.5(frame/sec)	155.(KB/sec)	123.(KB/sec)
Salesman / Difference off	12.4(frame/sec)	189.(KB/sec)	456.(KB/sec)

Table 1: Transfer and display speed in our experiment

Since the data rate is close to the bare TCP/IP transfer rate, the bottleneck is network bandwidth if difference compression is effective. The worst case for the display routine is simple background frames without difference compression, drawing the entire image for every frame. In this case, data rate is about 2/3 of the base TCP/IP transfer rate. This means the display routine consumes 1/3 of the CPU power.

In this experiment, realtime performance (30 frames/sec) is attained in both the simple background and complex background cases. If the display has a complex background, difference compression is required for real-time display. The system load is less than 50% for the simple background case and less than 60% for the complex background case. Since this computer has 16 MIPS, the power necessary to receive and display the speech video is about 10 MIPS.

5.2. Effect of Threshold Adjustment

In our method, data rate and quality highly depend on the variance threshold. If the threshold value is large, the data is compressed further and the quality of video is degraded. Even in this low quality case, however, the important part for communication such as eyes and mouth tend to remain. In this sense, the quality of video degrades gracefully. In the data that are presented in Fig. 6, the amount of data is about 1 K bytes per frame; it is half the size of the data shown in Fig. 2. The regularized variance threshold value for its variance is 0.078.

Fig. 6: Example of low quality image.

We measured the maximum speed in frames per second for various values of the variance. The required bandwidth is the average amount of data for 30 frames. The video data is "Miss America," which has a simple background. The other condition is identical to the previous measurement.

Threshold	Maximum Speed	Bandwidth
0.000	52.6 (frame/sec)	127.(KB/sec)
0.008	54.6 (frame/sec)	124.(KB/sec)
0.016	57.2 (frame/sec)	119.(KB/sec)
0.032	72.4 (frame/sec)	93.(KB/sec)
0.047	97.0 (frame/sec)	69.(KB/sec)
0.063	120.0 (frame/sec)	54.(KB/sec)
0.078	146.3 (frame/sec)	44.(KB/sec)
0.094	171.9 (frame/sec)	36.(KB/sec)

Table 2: Maximum speed and regulated data rate.

If the data has rapid movement and complex objects, it may not fit the bandwidth of TCP/IP after the compression. In that case, we can adjust the compression rate by modifying the threshold value. We can realize graceful degradation by controlling the variance threshold value.

6. RELATED WORK

Standard video compression methods, such as MPEG or JPEG, are being defined. Several digital video boards are developed and under development. The Gigabit network hardware is under development. This work will coexists with these technology in the future because not all workstations have both gigabit and digital video board even after they becomes popular. There is still need to display video data on a lower priced workstation that has no digital video board.

7. CONCLUSION

A workstation that has more than 10 MIPS power can receive and display video, using the method described in this paper. The hardware that is used for this experiment is a SparcStation 1+ with monochrome display. Video data is compressed and stored on the remote machine. The original video format is 256-level gray scale, 192x144 pixels; about 25K byte / frame. They are compressed to less than 5K byte / frame (1.45 bit / Pixel) and stored on disk. On playing, the data is sent through an Ethernet using TCP/IP. The SparcStation does decompressing, magnifying and dithering at the same time. The displayed result is dithered 768x576 monochrome, 30 frames per seconds. On the tested data, the average processor load is about 50%; the SparcStation can do the other job at the same time. The displayed quality is good enough for video communication, such as video conference, video telephone, and video presentation.

Commonly used workstations can receive video information with this method; this means we can get more places for digital video to be shown. It provides more chance for the video application system to be popular. Though it is only one bit of information that the workstation can receive and display video, it affects the design of digital video system. The quality of this method allows the viewer to recognize what is going on, but not good enough to play movies; the quality is something like a telephone and not a stereo. Considering the quality, the application should use the merit that the information is controlled by computers.

A video database is one reasonable application; for example, it might contains a manual for a complicated instrument. Video mail is another good application. Since video data is huge even with compression, it is reasonable to send the data when it is needed, instead of spooling on a local disk. It is reasonable for readers to open a channel to the sender's computer and to get the video information related to the message when he/she reads the mail. The combination with networks reduces the usage of disk space and makes video mail system feasible.

Experiments with real-time compression are the next step to make video communication possible and to make data retrieval easier. We have already developed an experimental real-time compression program, but it has still much room for improvement; measurements have not yet been done.

The design and implementation of a video transfer protocol for our compressed video data is future work. Though TCP/IP is usable in this experiment, it is not an ideal protocol for video. [3] By changing protocol, we can get more bandwidth from the Ethernet and may be able to prevent timing errors. Many work has been done for this purpose. Since the implementation of protocol does not require additional hardware, it satisfies the constrains of this work.

Consideration of color video is also future work. The idea of sending various size of tiles will be applicable to color video. Since we have remaining processing power for display, and may get extra bandwidth by changing the protocol, it is reasonable to think of color display in the same manner.

8. ACKNOWLEDGEMENT

We express deepest thanks to Michael Gilge at International Computer Science Institute at Berkeley, (ICSI), and Riccardo Gusella (ICSI) for their video data, and discussion of various compression algorithms. We thank George Homsy (ICSI) for the discussion on the difference compression algorithm. We also thank Domenico Ferrari (UC Berkeley) and David Anderson(UC Berkeley) for encouragement and guiding the direction of this work.

REFERENCES

[1] D.J. Healy and O.R. Michell, "Digital Video Bandwidth Compression Using Block Truncation Coding", IEEE Trans. on Communications, Vol. COM-29, No.12, Dec. 1981.

[2] B.G. Haskell, F.W. Mounts, and J.C. Candy, "Interframe coding of video telephone pictures," Proc. IEEE, vol.60, pp.792-800, July 1972.

[3] L.F. Cabrera, E. Hunter, M. Karels and D. Mosher, "User-Process Communication Performance in Networks of Computers", IEEE Trans. on Software Eng. Vol. 14, No. 1, pp. 38-53. Jan. 1988.