# Experiences with real-time software video compression

Ron Frederick
Xerox PARC

July 22, 1994

## 1 Introduction

As workstation speed continues to increase, real-time processing of multimedia streams on the system's general purpose CPU has become more and more feasible. This approach has several attractive properties over using dedicated special purpose hardware. The development environment is much more advanced, and the code which can be written there is often much more portable, allowing it to continue to ride the wave of workstation performance improvements into the future. A software-only solution is especially attractive, as it can be quickly installed on all workstations at a site without the expense involved in replicating the hardware.

These motivations led to the development of **nv**, an Internet video conferencing tool. This paper will describe the basic design goals of **nv**, detail its video compression algorithm, and relate some of the experiences of trying to put this video on the Internet.

## 2 An overview of the nv design goals

The design of **nv** revolved around two major goals. First, it should be as portable as possible. This meant that it should allow one to receive video in a standard X11[1] window without requiring special hardware of any kind. It also meant that transmitting video should require only a very basic frame grabber, capable of returning raw pixels of the captured video image. Second, the program should run over a wide range of network bandwidths, as the Internet is very diverse in the regard. It should support sending low frame rate video over something as slow as a modem line, while also allowing higher quality video to be sent over a high speed LAN.

To achieve the first goal, a video compression scheme was designed that could be performed completely in software at a reasonable speed. It takes advantage both of the similarity between consecutive frames in a typical video stream and the similarity between neighboring pixels in any given region of a frame. It typically provides compression ratios of 20:1 or more, while taking only a few tens of milliseconds on a typical workstation to compress or decompress a frame.

To achieve the second goal, two approaches were used. First, the sender is presented with multiple resolutions they can send at, as well as the choice about whether or not to send color. This allows very crude control over the bandwidth required, by trading off some quality. In addition, they are given a slider which allows them to pick precisely a maximum bandwidth to send. The program enforces this bandwidth by carefully spacing packets being delivered to the network. If the video image has too much information to fit in the bandwidth selected, frames are sent at a slower rate. On a high bandwidth LAN, frame rates of 10-15 fps are quite possible with a typical workstation. On a much lower speed link like a modem, it might only achieve a frame every few seconds.

## 3 The nv video compression algorithm

The **nv** video compression algorithm is a two-step process. First, the current frame is compared to the previous frame, looking for which portions of the image have changed significantly. Only those areas which have changed will be retransmitted. This differencing step can be done very quickly, reducing not only the bandwidth required to send the video but also the total amount of work to compress the frame since only the areas which change need to be processed by the full compression step. In a typical teleconference, this step will achieve a compression ratio of 3:1 or more.

Each block marked as having changed is then compressed further using transform coding. The transform is a two dimensional Haar wavelet. The output of the transform is thresholded to eliminate low energy terms and then run-length encoded. This step with typically achieve additional compression of 6:1 or more. Experimentation is also being done with a Discrete Cosine Transform in place of the Haar wavelet. While this transform is more expensive to compute, it almost doubles the compression ratio achieved. The program dynamically selects between the DCT and Haar transform based on whether network bandwidth or local computation is currently the limiting factor on performance.

To improve the image quality, **nv** also periodically transmits portions of the image which have become stationary in higher resolution. The same transform and run-length encoding is used, but the thresholding step is eliminated. This produces a nearly lossless version of the original image, while still achieving approximately 2:1 compression. The result is that all stationary portions of the image are updated to high resolution fairly quickly after a scene change.

# 4 Putting nv video on the Internet

Dealing with transmitting video over a lossy network has presented some interesting challenges. Experiences with **nv** and other Internet audio and video tools were fed into the development of a new transport protocol by the Audio/Video Transport working group of the Internet Engineering Task Force (IETF). This protocol is called the Real-time Transport Protocol (RTP), and it provides support for sequencing, timing, and quality of service reporting of real-time packet streams transmitted either point to point or point to multipoint using something like IP multicast. The video streams sent by **nv** are now using RTP framing. In addition, each packet of an **nv** data stream is designed to be processed largely independent of the order in which it is received, by properly identifying what part of the frame is contained within it.

Most packet losses are coped with by motion in the image causing retransmissions in a subsequent frame. To deal with portions of the image which have become stationary, the sender spends a fraction of the bandwidth periodically retransmitting stationary blocks. This allows receivers which join a conference already in progress to get a complete image and also fills in any damage caused by packet losses after a short time. This mechanism requires no explicit acknowledgement from receivers, which makes it particularly well suited to a multicast transmission where there are many simultaneous receivers which might be seeing very different amounts of loss.

The smooth control over transmitted bandwidth has proven an extremely useful feature in the Internet environment. When receivers notice large amounts of loss due to congestion, the sender can simply lower the transmitted bandwidth to compensate. In version 2 of the RTP protocol, a standard mechanism has been defined for reporting losses, and work is underway to develop an automatic bandwidth adaption system which uses these reports.

## 5 Conclusions

Workstations have reached a level of performance which makes software video compression a very serious possibility. With only a simple frame grabber at the sender and no special hardware at the receiver, reasonable quality video can be sent across the Internet. Using the main workstation processor allows the code to be highly portable and the quality to continue to improve as workstations get faster.

Video has very different timing and retransmission requirements from other types of data transfer. As a result, new transport protocols for it make sense. This is especially important when considering multipoint delivery. Protocols such as RTP are a first step in this direction. Additional work is still required on good congestion control schemes, prioritized delivery, and other mechanisms to provide quality of service guarantees.

The **nv** software is freely available for research and evaluation purposes. It has been ported to most major UNIX workstation platforms. Source code and executables can be found on **parcftp.xerox.com** in the directory **/pub/net-research**. Comments and questions should be sent via email to **frederick@parc.xerox.com**.

## References

[1] Robert W. Scheifler, James Gettys, "The X Window System", third edition, Digital Press, 1992.

[2] H. Schulzrinne, S. Casner, V. Jacobson, R. Frederick, "RTP: A Transport Protocol for Real-Time Applications", Internet Draft ietf-avt-rtp-05. July, 1994.