Gated Compression:

An Application-Specific Technique for Achieving High Degrees of Video Compression

J. Duane Northcutt and James G. Hanko
Sun Microsystems Laboratories
2600 Casey Avenue
Mountain View, CA 94303

Abstract

There exists a class of interesting video applications that generate large volumes of data, and for which existing video compression techniques alone do not reduce the data to a manageable amount. Examples of this type of application include a wide range of video monitoring applications which use stationary video cameras (e.g., facility security cameras, video monitoring of manufacturing processes, etc.). This paper presents a new compression technique for just such a class of video applications, that can provide compression ratios of on the order of 5000:1 and are realizable with extremely low implementation costs.

This new technique, known as "gated compression," works in conjunction with standard video compression techniques and provides an additional level of compression by suppressing (or "gating") the compressed video stream. This technique involves the detection, in real-time, of when a newly compressed field of video does not differ significantly from the previously transmitted video field, and simply discarding the field.

This compression scheme also lends itself readily to the inclusion of a rate-limiting function. Rate limiting can be used to ensure that the resulting data stream does not exceed a specified storage limit over a given interval of time. Effective rate-limiting is accomplished by applying a modulation function to the on/off gating of the compressed video stream.

This paper defines the key principles of the gated compression technique, describes an Internet-based hardware implementation of this compression scheme, and presents some empirical results to illustrate the effectiveness of this approach.

Keywords
Audio and Video Compression, Special Hardware Devices

I. Introduction

There exist many video applications that continually generate enormous quantities of video data which contains very low information content and is sparsely distributed. This class of applications is characterized by (typically) having a set of fixed-mount video cameras, "staring" at an occasionally varying scene. Examples of this type of application include video monitoring applications such as traffic observation, building security surveillance, manufacturing process monitoring, etc. These "staring" video applications emit huge amounts of data that presents a storage problem, as well as a problem in locating data of interest. For example, a single video camera generates nearly 2TB of raw digital (YUV422 format[8]) video in the course of a day. Even if compression schemes are used that offer compression ratios of 100:1, this volume of data is difficult and costly to transmit, store, and retrieve[4] — and this is all made that much worse by having to deal with multiple video sources in a given application.

The technology described in this paper provides a practical means of compressing this type of video stream with compression ratios that are orders of magnitude greater than is possible through traditional video compression approaches alone. In addition to reducing the required storage space for the acquired video data, this compression method also assists in the automatic identification of significant events within the video data set.

This is made possible by the fact that the video data generated by these applications contains tremendous amounts of temporal redundancy — the cameras are focussed on the same scene for long periods of time, and punctuated with only brief intervals of something interesting happening. Traditional digital video compression techniques (such as MPEG[2]) attempt to minimize the amount of data that is generated from these largely unchanging scenes. However, they still generate unmanageably large amounts of data with this class of application. For example, an MPEG1 video stream of a constant scene representative of a security video camera's view generates a constant data rate of over 16GB of data per day.

A. Overview of Gated Compression

Gated compression is a technique which takes advantage of the characteristics of some of this class of applications in order to achieve a multiplicative increase in compression ratio over that offered by other video compression techniques. In the example implementation described in this paper, the data rate for a single surveillance video camera is reduced to a maximum of 1GB per day, and some selected minimum value (e.g., 500KB per day). The actual amount of data generated by a given camera is, of course, a function of the activity which occurs in the camera's field of view. It is shown in this paper that an average camera in the experiments described below generated approximately 300MB of data per day, providing an effective compression ratio of over 5000:1, without loss of significant data.

The gated video compressor only emits compressed video fields when a particular field is significantly different from the last field it has emitted. This means that the gated compressor goes silent whenever a series of fields that do not differ significantly are detected, and runs at full video capture rate whenever the observed scene is changing.

The key concept behind this form of compression is the detection of "significant" changes between each captured video field and a selected reference field. This field difference information is used to enable (or "gate") on and off a stream of compressed versions of the input video fields.

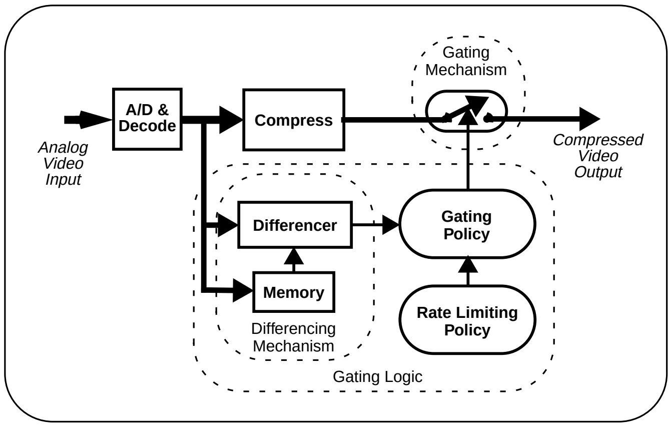

Figure 1 Diagram of a Gated Video Compression Scheme

As shown in the figure above, the main flow of data through a gated compressor is from the digitizer/decoder unit, to a standard video compression unit, and then on through the gating mechanism which determines whether each individual field of compressed video is emitted by the compressor, or discarded. The gating logic operates in parallel with the digitization and compression of the video signal, and is the function which determines which fields pass the gate, and which are suppressed by the gating mechanism.

The gating logic contains a mechanism that calculates the (weighted) difference between the current incoming video field and a stored reference field. At the end of each incoming field, the gating policy module evaluates the accumulated difference information and decides whether the most recently compressed field should be transmitted or dropped. Should the gating policy determine that the incoming field is sufficiently different from the reference, the gate is opened and the compressed field is passed on to the next phase in the video pipeline (e.g., to networking code where it is encapsulated and sent over the network, or to a file where it is stored). Also at the end of each video field time, a new reference field may be chosen according to a defined policy.

In addition to all of this, a rate limiting module monitors the cumulative volume of data being generated by the compressor and signals the gating policy to inhibit the production of fields that it would ordinarily have allowed to be emitted, in order to enforce a given maximum data rate over a given interval of time.

There are many different policy decisions that are applied to the mechanisms of a gated compressor in a specific implementation. These may include a number of parameters which define exactly what constitutes a "significant" change in the video signal, and exactly how the decision to transmit a field is made. The following sections describe some of the policies which have been shown to be effective in an example implementation of a network appliance which incorporates an implementation of a gated video compressor.

II. Details of the Technique

The heart of this compression technique lies within the gating logic function shown in Figure 1. This functional unit which determines when an incoming field is worth passing on to the next stage in the system. In effect, the gating logic is performing a simplified form of motion detection on a live video stream, and generating a control signal which is active whenever motion is detected within the incoming video stream.

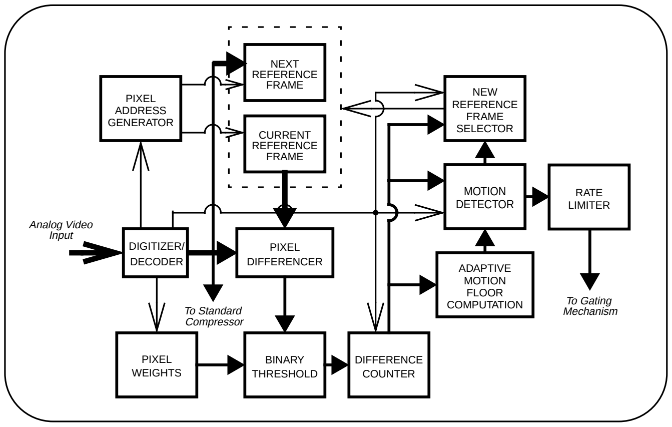

Figure 2 Detailed Block Diagram of the Gating Logic Component

The function of each of the main components illustrated in Figure 2 is described below, along with the technical motivations behind some of the more significant design decisions relating to each of the functions.

A. Video Digitization and Decoding

The first stage of a gated video compressor is where the analog video signal is digitized and decoded. The decoder emits a serial digital data stream that is comprised of a subchannel of luminance information, along with another subchannel containing the video signal's chrominance information. In addition, the decoder emits a series of control signals which provide pixel, scan-line, field, and frame timing information. This timing information is used to synchronize the various sub-sections of the compressor.

Although the decoder passed on all of its output information to the standard video compressor in the next stage of the gated compressor, it is possible to pass only the luma portion of the video signal to the gating mechanism. By operating only on the luma portion of each video pixel, the cost and complexity of a gated compression unit can be reduced significantly over that required if the full color signal were processed. This optimization can be performed when color changes in the video input signal, in the absence of any changes in the brightness of the signal, are not possible, or deemed uninteresting. This corresponds well with the response of the human visual system which perceives changes in brightness much more readily than differences in color. It should be noted that, while the gating logic may be disregarding the video stream's color information, the standard video compressor would still operate on the full video signal. Therefore, the output of the gated compressor would be in full color. Thus, the use of luma information only in the gating logic section would not necessarily diminish the effectiveness or functionality of this compression scheme. Throughout the remainder of this description it is assumed that the gating logic is operating on luma values only.

B. Serial Pixel Differencing

The first stage in the gating logic function involves the comparison of the incoming video field with the reference field. As each pixel digitized it is compared to the pixel corresponding to it in the current reference field. This comparison determines the numeric difference between these two pixels. In the ideal case where a video scene is not changing, the value of each pixel at the same location within a video field should have the same value across successive fields. However, this is not true in practice as the sensors and transport paths in video cameras are subject to the injection of noise. This means that even when a scene does not change, the value of pixels at any given location in the video field will change from field to field.

The degree of variation in pixel value which occurs from field to field due to noise has been found to be well-defined and consistent (across individual video cameras of moderate to high quality). Therefore, a thresholding mechanism must be applied to the field-to-field pixel differences in order to deal with the effects of pixel noise.

In this compression scheme, this thresholding is accomplished by taking the absolute value of the difference between an incoming pixel and its reference pixel, and comparing it with a given threshold value. If this difference is greater than the threshold, the variance cannot be attributed solely to noise and so the incoming pixel is considered to be significantly different from its reference value.

Taking the absolute value of the pixel differences has the effect of making sure that it does not matter whether the incoming pixel is brighter or darker than its reference pixel, but only the magnitude of the difference between pixels is considered in the comparison operation.

The output of the per-pixel differencing unit is a binary signal that indicates whether the current incoming pixel is significantly different from its reference value. This binary stream is sent on to the motion detection unit which is described below.

An augmentation may be made to this basic pixel thresholding mechanism by using different threshold values for each pixel position. This has the effect of allowing each pixel position to be made more or less sensitive to variations.

In this way, the compressor can be made to be less sensitive to changes in some areas within the field of view, and more sensitive to other areas. This could be useful when an area with a continuously moving object (e.g., a clock or a fan) exists. In such a case, the sensitivity within this area could be attenuated. Similarly, some areas (e.g., a consistently lit door, desk top, or assembly line) could be defined to have a hypersensitivity to change. An image editing interface can be used to specify the regions that should be more or less sensitive to change in pixel values, the output of which can be loaded into an array of pixel threshold weights. These weights define the degree of change required before a given pixel can be deemed to have changed.

C. Significant Video Field Change Detection

The motion detection unit determines whether sufficiently large numbers of pixels differ from their corresponding values in the reference field to result in an indication that the incoming video field has changed. This is done by maintaining a count of the number of different pixels for each field of incoming video, and comparing this count with another threshold value. In an overly simplistic example, a significant field could be identified if more than N of the M pixels in the incoming field are considered different from those in the reference field. While this approach could be used with a gated compressor, such a scheme would suffer from a number of deficiencies which would degrade the effectiveness of the compressor.

The use of a constant pixel threshold would result in a compressor that would have a constant level of sensitivity to change in the video field of view. This would be mean that the unit would behave the same regardless of whether the scene has been static, or varying over a recent past interval of time, resulting in the same number of fields being captured during periods of sustained activity as during isolated events. This is undesirable when the application wishes to record more fields during a sporadic event, and less during sustained periods of activity.

A better approach is one where the unit is able to detect when the current "ambient" level of change is high and desensitize the motion detector in proportion to the amount of change in video frames that is currently being observed. This can be done in a particularly effective manner by computing the average motion over a recently passed interval of time and defining a field to be significant when it's number of changed pixels differs from the average by a given amount (as opposed to when it exceeds a fixed value). It is possible to maintain an effective average of recent per-field cumulative pixel difference counts, without having to maintain a long history and perform a sliding window average over these values.

A trailing exponential average function such as that given in Equation 1 can be used to generate the average number of pixels which are changing within a field over time. The result is an average where previous field's values contribute successively less to the current average. This provides a good approximation to a sliding window average, at a much lower cost, and much simpler implementation.

$$a_i=(\alpha*a_{i-1})+((1-\alpha)*x_i) \qquad \text{Equation 1}$$

One problem with any sort of averaging calculation like this is that it is effectively a low-pass filter, which suffer from a problem known as phase lag — i.e., the value of the computed average lags behind that of the actual value. The effect of this phase lag is a reaction time “shadow,” where the sudden drop off in actual activity leaves the compressor excessively insensitive to smaller changes in motion. This could allow the unit to fail to detect significant motion following a sustained period of higher motion.

This problem can be dealt with by building into the adaptive activity detection function an asymmetric response to changes in the amount of motion being observed. In particular, when the difference count for a field is significantly less than the current average, the current value is weighted more in the average. This has the effect of bringing the average value down quicker, thereby reducing the sensitivity “shadow” area, and making the compressor more adaptive to different situations. In effect, this detects the video signal’s noise floor — i.e., the average of the recent least changing frames.

D. Reference Field Selection

In many applications of this type of compressor, it is important that changes in lighting not be confused with motion or some other significant change in the scene being monitored. The ability of the compressor to distinguish significant motion from other artifacts depends greatly on the policy used to choose the reference fields.

At the end of each video field time — and in parallel with the determination of whether the compressed field is to be emitted — a decision is made as to whether the current reference field is to be updated with the newly arrived field. A simplistic reference frame selection policy would have each incoming field serve as the reference field for the subsequent input field. However, the use of this policy has the effect of making the gating logic unable to detect very slow moving objects. Another policy might choose a new reference field periodically (e.g., every Nth frame). Unfortunately, this too has the effect of not being able to detect small changes which occur over time. While this might be a desirable property for some applications, another policy might suit other applications better. In particular, a policy which chooses the reference frame to be the last field of video which was captured prior to the video being gated off is particularly useful. With this policy, the motion effects of slow moving objects will accumulate as the incoming pixels gradually diverge from their reference pixels, eventually causing a significant field event to be triggered.

Regardless of the specific reference field selection policy that is chosen, it is important to take into account the specific fields being compared in making the gating determination. If care is not taken in the choice of fields being compared, it is possible for a source of apparent, but false, change can be introduced in the field comparison process. This is due to the phase alternation of chrominance information in standard video coding formats. This phase alternation is useful in the analog domain, but introduces an artificial shift in the value of a given pixel location from field to field. An effort must be made to ensure that comparisons are only done between incoming video fields which belong to the same phase group as the reference field.

E. Cumulative Data Rate Limiting

Another mechanism which is used to great advantage with this compression scheme is the data volume limiter function. This mechanism is used to ensure that the effective data rate of the device does not exceed a given amount over some period of time. This function is particularly useful for ensuring that devices which use gated compression do not overload a network, or to ensure that storage space is not exceeded in the event that the output of a device is being recorded.

This mechanism keeps track of the number of bytes which have been sent by the device over some past period of time. In addition, the mechanism has user-definable parameters which indicate the maximum amount of data which can be sent over a given interval of time. The rate controlling unit monitors the device’s history of data transmission and projects forward in time to determine whether the current data rate needs to be reduced in order to meet the overall data production restrictions.

The rate limiting mechanism calculates a nominal rate for the compressor as a whole, and uses that rate to determine if additional fields should be discarded in order to reduce the unit’s total output of data. For example, if the device’s parameters are set so that it is not to generate more than 50MB per hour, the nominal rate over that hour is roughly 7Mbps. At the beginning of a period of activity there is plenty of budget, so no attempt is made to limit the rate and bursts of traffic greater than the nominal rate are allowed. However, the rate limiting mechanism keeps track of what has been sent and as time goes on, if the amount of data sent exceeds the nominal rate, the rate limiter will instruct the gating mechanism to suppress additional fields until the nominal rate is achieved. After a prolonged burst of traffic above the nominal level, the rate limiter will cause fields to be dropped in a dithering-like fashion (i.e., with an irregular duty-cycle) so that the effective data rate will remain at or below the nominal level, but large groups of consecutive fields are not discarded. In addition, when fields are being dropped for rate control purposes, the reference field is not changed, so that the next field allowed under the rate-control will show a significant difference.

III. An Example Implementation

To validate the viability and explore the effectiveness of the gated video compression described here, a video Internet appliance known as the NetCam was created. A photo of an actual NetCam unit can be seen in Figure 3.

The NetCam is a small, self-contained unit, which acts as a full network citizen, abiding by all the conventions and operating with all the standard protocols which define a host device on an Intranet or on the greater Internet. NetCam connects to a source of power and then directly to the network and has no switches, jumpers, or external displays — it simply plugs into the network as if it were a household appliance.

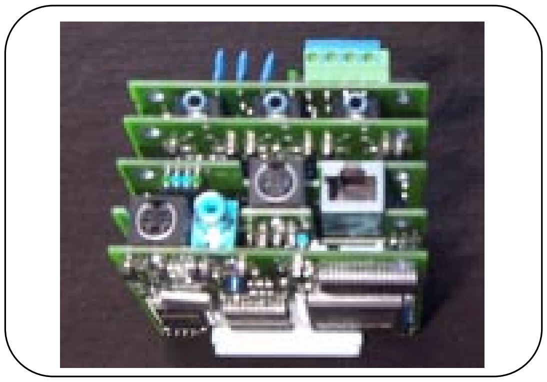
Figure 3 Photograph of the NetCam Internet Video Appliance

Multiple instances of the NetCam can be connected with off-the-shelf computer networking hardware to create digital versions of the traditional analog closed circuit television systems. In addition, these devices can coexist with computer systems on a (TCP/IP-based) local area network (such as Ethernet), or on the larger Internet. This device includes support for configuration via DHCP, ICMP message handling, address resolution with ARP and RARP protocols, sending and receiving of IP, UDP, and TCP packets, as well as transfers between files on a remote host and its local memory by way of the TFTP protocol. Most significantly however, this device can send and receive digital audio and compressed digital video using the Internet standard RTP and RTCP protocols[6], as well as being able to use IP multicast protocols. The use of multicast RTP-encapsulated motion JPEG (MJPEG) and audio was popularized on the MBONE[5]— the experimental Multicast Backbone of the Internet. This device interoperates with a number of applications written for use with these standards — e.g., it can be used to send audio and video data to the popular "vic" and "vat" video teleconferencing software[7], as well as the ShowMeTV product from Sun Microsystems.

While the RTP/RTCP protocols have been specified to work with other forms of video coding, such as MPEG, Motion JPEG is a better choice for most applications. While MPEG can provide higher compression ratios than motion JPEG, it suffers when run on a computer communications network with the possibility of congestion-based packet loss. MPEG is not well suited for use with unreliable or datagram-based protocols. This is because the loss of a single packet of an MPEG stream could result in the loss of an entire Group of Pictures (GOP), which could be tens of frames of video. With MJPEG, it is possible to lose a packet (and perform very simple error concealment) and not lose even one frame of video. Another benefit of the use of MJPEG over MPEG is that MJPEG encoding is significantly simpler and cheaper to encode and decode than MPEG encoding. This allows the NetCam to operate at full video frame capture rate and still be a small, low-cost, and low-power device.

A. Overview of NetCam Hardware

The key functional specifications for the NetCam hardware include:

- multi-format video digitizer (PAL/NTSC)
- full-frame rate MJPEG video compression
- single channel digitizer of microphone or line-level audio at standard sampling depths (e.g., 8 or 16 bits per sample) and rates (e.g., 8KHz, 32KHz, 44.1KHz, 48KHz, etc.)
- full-/half-duplex 10/100Mbps Ethernet interface
- hardware support for gated video compression
- less than 15W total power consumption
- less than 300cc total volume
- 24VAC external transformer, on-board power supply
- 512KB boot PROM
- 512KB FLASH
- 8MB DRAM

The NetCam hardware was designed to be highly flexible and a modular design approach was taken where each major function is implemented on a roughly 3"x3" printed circuit board. The boards which make up the NetCam are: the CPU board, the 10/100Mbps Ethernet board, the audio codec board, the video digitizer/compressor board, and the power supply board. These boards and their external connectors are shown in Figure 4.

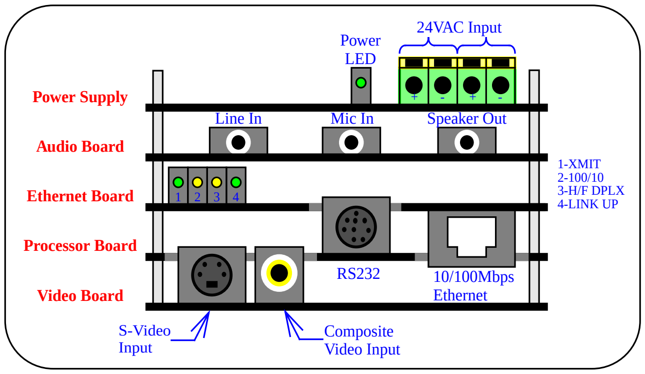

Figure 4 The NetCam Front Panel Connectors

This modular construction makes it possible to replace any of the major components with a minimal amount of effort. It is possible to easily exchange the Motion JPEG compression unit with an MPEG compression board should it be so desired. It is also possible to swap out the network interface board and have the device work with any number of different network types, as opposed to just Ethernet — e.g., Token Ring, ATM, etc. Likewise, the processor module can be replaced with a different type of CPU. The firmware is similarly modular in that there is very little effort involved in changing the specific implementation of any of the major components.

In order for this device to be a simple, self-contained, stand-alone appliance, yet be able to function in a fully general Internet environment, it is required that all of the necessary settings and parameter changes be remotely controllable across the network. The NetCam allows all of the internal settings to be read and written over the network, instead of requiring the device be connected to a control panel, or external computer in order to change the unit's parameters. In addition, all administration and update functions are available over the network as well. An initial set of configuration values are placed in the CPU board's flash memory when the system is first installed, and then all subsequent changes to the configuration values of the unit can be done over the Internet. For true unattended remote operation, in addition to all this, the unit must be able to automatically detect and recover from failures. The NetCam does this with a watchdog timer which must be successfully reset periodically or the system will restart itself. Also, the device automatically resets and restarts itself following any power failure or other exception condition. As an additional security feature, the NetCam periodically emits a keep-alive message that indicates whether a valid video input signal is present.

A diagram illustrating the main components on each of the NetCam boards is given in Figure 5.

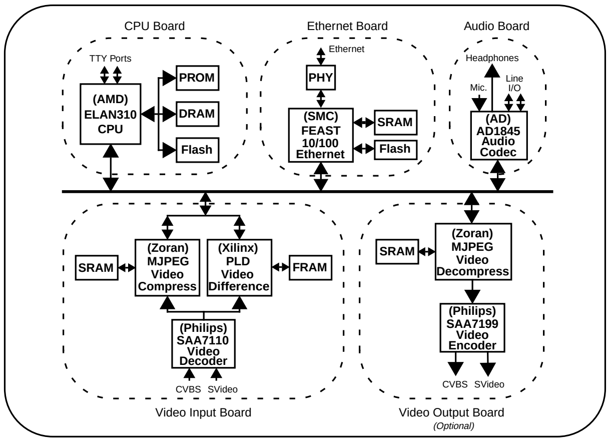

Figure 5 NetCam Hardware Block Diagram

B. NetCam Software Features

The major software components within the NetCam include the PROM-resident monitor, the executive and the application program. An overview of the key features of each of these components is provided below.

The PROM monitor in the NetCam provides the following functions:

- peek/poke registers, i/o, and memory locations
- download (S-Records) over serial line
- manipulate simple flash file system
- start/stop program execution

The NetCam's real-time executive includes the following functionality:

- threads — enables preemptive multitasking in single and multiple processor systems
- mutexes — synchronize thread access to resources
- message queues — enables inter-thread communications
- memory management — supports allocation and freeing of memory
- real-time support — provides priority scheduling with full preemption and priority inheritance to ensure critical tasks are handled in a timely manner
- device driver architecture — handles low-level interrupt processing, etc.
    - drivers implemented for Ethernet, serial port, clock, and video and audio subsystems
- debugging support — trap handling with symbolic back-trace, thread profiling, assertions
- application support, including:
    - libc functions: printf, sprintf, memset, bcopy, bzero, memcpy, strlen, and memcmp
- packet-filter-based network protocol stack, including:
    - ARP/RARP, ICMP, DHCP, IP, UDP, TCP, TFTP, and RTP/RTCP

Each NetCam unit is initialized to contain the following set of application programs pre-loaded in the simple flash file system:

- *netconfig* — a network parameter configuration tool
- *tftp* — an implementation of the trivial file transfer protocol
- *netcam* — the audio/video transmitter application program

IV. Experimental Results

A prototype Java-based distributed security video application was created (in cooperation with Sun's Corporate Security group) to explore the effectiveness of the NetCam and its implementation of gated video compression. This application was designed as a prototype for a network-based digital replacement for the functionality currently being provided by a traditional analog camera and coaxial cable distribution system.

A. An Example Corporate Security Video System

The existing analog security video system is a fairly typical installation with analog video cameras monitoring all of the entrances and exits of each building. A typical building has between 6 and 12 security video cameras. The video from each group of buildings is transported to a central collecting site where it is recorded and stored. The output from each camera is recorded in a time-lapse fashion, at between one and five frames per second, on a dedicated (industrial-grade) video recorder.

A full-time security staff exists to change video tapes, maintain the cameras, the video distribution network, and the video recorders. The security staff is responsible for retrieving and reviewing recordings as part of security investigations, as well as monitoring the video from selected cameras in real time.

This sort of installation is clearly quite costly both in terms of the amount of equipment required, as well as the manpower costs required by such a system. While this may represent the current state of the art in video security systems, it leaves much to be desired in terms of both cost and effectiveness. The sheer number of video feeds that must be provided to support a campus with a dozen buildings, each of which having a dozen cameras, presents a serious management problem. In practice, the ability to provide investigators with useful information is limited both by the technical limits of this kind of system and the limits of a highly human-interaction-intensive process. It has been reported that it takes approximately one hour of investigator time for each hour of recorded video reviewed[9].

Clearly, an opportunity exists to provide improved effectiveness, at substantially lower cost, with a digital solution. However, a examination of the quantity of video that must be handled leads quickly to the conclusion that existing compression techniques are not up to the task. This is an example of where the NetCam and gated video compression can provide a superior solution.

B. A NetCam-Based Security Video Application

A prototype NetCam-based security video application was created, and is illustrated in Figure 6. In this system, a NetCam unit is paired with each of the existing security cameras in a set of buildings, and connected to a private local-area subnetwork. The video is collected at a server which archives the data to tape. This archive server is connected to both the private security video subnet and the campus backbone network. The archive server maintains a recording of the camera outputs over a given interval of time and behaves much like a video FIFO where each hour the oldest hour of video from each camera is replaced with the most recent hour's video.

The archive server not only records the video emitted from all of the connected NetCams, but also responds to requests to playback stored video sequences. A Java-applet-based web application is provided to allow the security staff to recall the video for a given building and camera, for a defined interval of time, from any place on the network.

In addition, the fact that the RTP encapsulated video from the NetCams is being multicast allows multiple simultaneous listeners to be active on each video stream. For example, the archive server's recording process, a monitoring process that generates an alarm when a camera goes away, and any number of live viewing applications could all be monitoring the output of a given camera's NetCam.

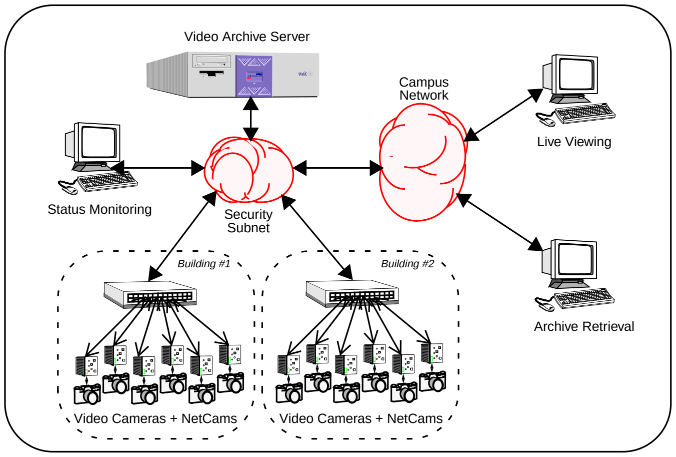

Figure 6 The Prototype NetCam-Based Security Video Application

C. Measurements from the Prototype System

The results of this initial experimental use of the NetCam demonstrate the overall effectiveness of both the gated compression and rate limiting techniques described above. These measurements demonstrate that there is great opportunity to exploit the advantages of gated compression, and the combination of gating and rate-limiting provides an effective means of reducing continuous video streams to manageable levels.

The highly regular and stylized traffic patterns which are typical of this application domain can be seen clearly in the graph shown in Figure 7. This graph shows the number of frames captured by a representative camera, for each hour of the day, over a four week period. The activity patterns of a common office building can be seen both within the daily cycle of activity which begins with the cleaning staff working around midnight, progresses to a quiet period leading up to the arrival of the building occupants in the morning, then ramping up to a sustained level of activity throughout the day, and then ramping down to another quiet period at the end of the day. In addition to this daily cycle of activity, the relative lack of activity during weekends is also evident in this graph. This data illustrates how periods of inactivity in the observed environment translate into natural opportunities for compression of the video stream. Whereas without the use of gated compression (and rate limiting), a video camera would capture over 200,000 fields of video per hour, the example shown in Figure 7 captures a maximum of less than 4,000 fields of video (and averages less than 1,000 fields) per hour. This behavior provides an additional two orders of magnitude in compression over whatever other video compression technique is used with the gating method.

The plots given in Figure 8 indicate that each of the example building's NetCams emit video fields with the same characteristic envelope. This is somewhat unexpected as the cameras are focussed on different entrances to the building, with obviously different usage patterns. These measurements reinforce the notion that a significant opportunity for gating video off exists within security video applications, regardless of the specific scene being monitored.

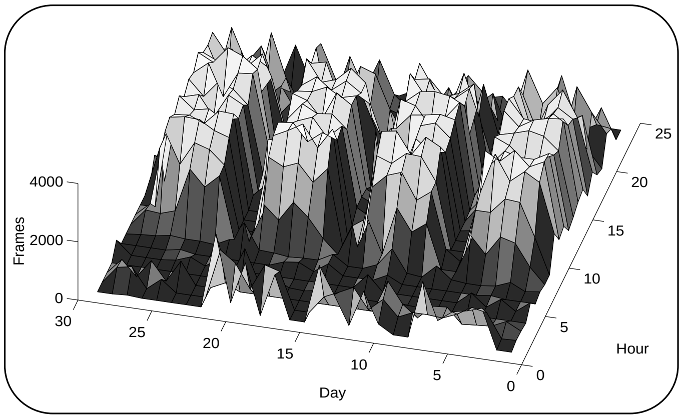

Figure 7 Video Fields Emitted from a Single NetCam Over a Month

The effectiveness of the NetCam's gating and rate-limiting functions can be seen in the plot given in . This graph shows the number of bytes of data emitted by a representative NetCam over the course of a ten week period. Both the representative daily and weekly characteristics are reflected in this graph. In this example, the subject NetCam was configured to limit its output to less than 1GB per day. The cumulative data volume over the given period was measured to be 78GB, with an average of 46MB/hour, a minimum of 600KB/hour, and a maximum of 146MB/hour.

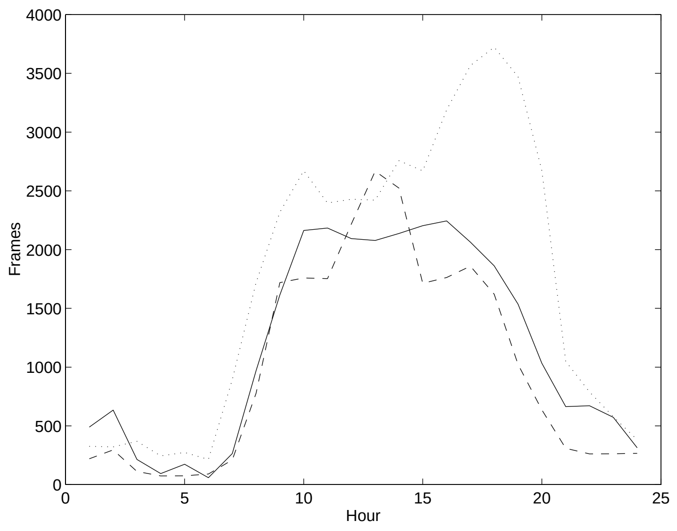

Figure 8 A Month's Average Daily Output of From Three NetCams

A measure of the degree to which the gating and rate-limiting mechanisms distribute the sampling of a video stream can be seen in the distribution of field capture times. An illustration of this for a representative NetCam is shown in Figure 10. In this plot, the approximately 800 frames captured during the hour interval shown are fairly well distributed over time. There is only one instance where the maximum of 30 frames is captured during a one second interval, and there was, on average, less than four frames captured during each second where activity was detected by the NetCam

.

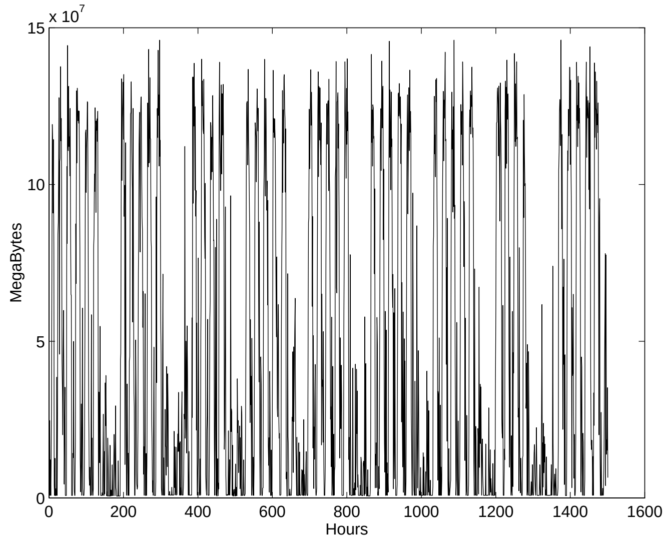

Figure 9 Amount of Data Emitted by a NetCam Over Nine Weeks

Finally, to put these figures in perspective, Table 1 is given which illustrates the storage required per camera for raw (YUV422 format[8]) video, MJPEG[1] and MPEG[2] compressed video, and gated motion-JPEG video.

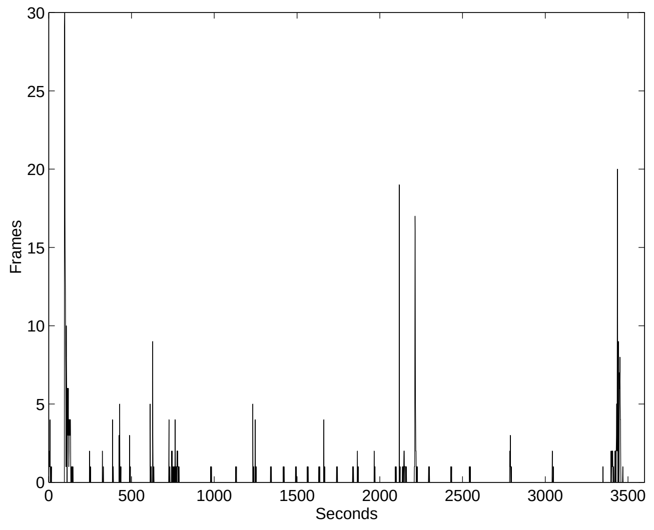

Figure 10 Distribution of Fields Captured Over a One Hour Period

	422YUV (1:1)	**MJPEG (50:1)**	**MPEG (100:1)**	**Gated & Limited MJPEG (2000:1 worst case)**
second	20 MB	400 KB	200KB	11 KB (avg.)
minute	1.2 GB	24 MB	12 MB	667 KB (avg.)
hour	72 GB	1.4 GB	720 MB	40 MB (limit)
day	1.7 TB	34.6 GB	18 GB	1 GB (max.)
week	12 TB	242 GB	126 GB	7 GB (max.)
month	52 TB	1 TB	540 GB	30 GB (max.)

Table 1 Volume of Video Data Output Over a Given Interval of Time

V. Alternative Solutions

Although a number of other techniques have been developed to detect motion in real-time video streams, each of these techniques suffers from one sort of deficiency or another.

One general class of techniques employs a variety of different adjunct means of detecting motion — i.e., by emitting or detecting some form of energy other than that in the visual spectrum (e.g., microwave, infrared, etc.). Typically in such systems, the adjunct detection system is used to switch a video camera or recording device on and off. These techniques tend to be less reliable and easier to detect and defeat than techniques based solely on the use of the information contained in the video stream itself.

The video-only techniques can be classified into those which operate in the analog domain and those which operate in the digital domain. A number of simplistic analog approaches have been employed (e.g., such as placing photocells on a television monitor) with less than the desired level of effectiveness. Analog techniques such as detecting changes in light values using one-shot timers to sample fixed locations in the video signal, or sampling fixed regions of the video signal and comparing against preset values, have been used to detect motion in video. Other analog techniques filter or integrate the incoming video signal and look for gross changes in the signal's characteristics to detect motion. These approaches tend to be inexpensive, but also provide poor results as they are dealing with some adulterated (and simplified) version of the video signal which discards the bulk of the information content of the signal. Working with a signal with so little information content, the best that can be achieved is a rough approximation that motion has occurred in the scene when the incoming signal changes in a particular way.

All of these analog techniques tend to be imprecise in what they measure and so have inherent limitations in the degree of sensitivity that they can have to actual motion, as well as being susceptible to false triggers. Digital techniques tend to be much better at reducing both false positive (i.e., detecting motion when there is none) and false negative (not detection motion when it exists) motion detection outputs. The digital approach has the property of being able to repeatably associate a numerical value with a physical portion of the video camera's field of view. This ability to quantify the light coming from an area in space makes it possible to more accurately determine when motion occur in the scene being observed than can be done by analog means.

Most digital continuous monitoring video systems tend to be variants of time-lapse video approaches, all of which are based on the assumption is that it is not practical to store all of the data, and it is not possible to know when something interesting is happening. Thus, they greatly reduce the capture frame rate (e.g., to one frame a second) and capture samples uniformly distributed in time. This approach is not very effective due to the fact that events of interest tend to be bursty and the sampling is done uniformly in time. The captured video still contains the same density of information of interest, however there are far fewer frames which contain the events of interest, and there is no indication of where these interesting frames appear within the captured data set.

A better approach would be to reduce the frame capture rate to as close to zero as possible during periods when the scene being monitored is unchanging, and then capture at full rate when something interest occurs. In this way, an equivalent number of frames might be captured, but the capture times are made to correspond with the times when events of interest occur. An additional benefit of this idealized approach is the reduction in uninteresting frames within the data set, which aids in the process of locating specific events of interest.

Other attempts have been made at determining when interesting events are occurring based purely on the contents of the video stream. The most promising, image-processing-based, techniques tend to be extremely hardware-intensive, and therefore too expensive for many applications. At the very extreme high end of the spectrum of approaches, many image processing techniques have been developed which automatically segment a video image into regions of pixels which correspond to objects in the camera's field of view. The motion of these object can then be detected, classified, and tracked[3]. These techniques are prohibitively expensive and rarely can be made to run in real-time — i.e., they typically cannot be used for digital video security applications.

Some digital techniques use the very computationally-intensive approach to detect motion, which involves taking regions (typically an NxM rectangle) of pixels from the incoming video stream and correlating them with the pixels in a reference image. This approach can be thought of as an approximation to the generalized image understanding approach described above. In this case, the incoming image is arbitrarily divided up into rectangles which are then compared against (to be localized on) a reference image. This is considerably simpler than trying to first segment the incoming image into objects and then compare the new location of the object against its location in the reference object. This technique is used as part of the MPEG video compression standard and is known as motion-estimation. While this approach can be quite effective in detecting motion, it is also costly and time consuming. Sophisticated and costly custom integrated circuits must be used to perform this kind of motion detection. Furthermore, this approach tends to be quite sensitive the to the quality of the incoming image; noise on the incoming video signal makes it very difficult to locate given regions in a reference image. For all of these reasons, motion estimation techniques have not been widely used for the types of applications of interest here.

Other digital techniques for motion detection in security video applications are based on the detection of edges in video images — i.e., abrupt transitions in color or brightness which serve delineate one region from another. This type of approach simplifies the process by requiring only transitions be stored and detected, as opposed to values of large numbers of pixels. This takes advantage of the fact that there is a high degree of correlation between pixels in a video image (i.e., large regions of pixels tend to share similar values). These types of devices tend to be very sensitive to false trigger events due to lighting changes. A stationary scene may appear to move as the lighting changes the location of shadows in a scene over the course of a day.

The majority of video motion detection techniques work on the principle of comparing the incoming video signal to some stored reference signal. Some devices are constrained to only use the previous frame as a reference. While this has the benefit of requiring less storage, is less sensitive to false trigger events due to slowly changing lighting, and it lends itself to a more simple implementation, it has the drawback of making the system unable to detect slow rate of change events. Ideally, a motion detection device could choose arbitrary frames as the current reference frame. This way, reference frames can be chosen periodically to adapt to changing lighting conditions, and also during a period where motion is detected, to allow the system to adapt to differing degrees of motion. This, in fact, is how the device being described here operates.

In the digital domain, a common method for detecting motion is to subtract the value of an incoming pixel from the corresponding pixel in the reference frame and accumulate the resulting difference and generate a motion indication should the difference signal exceed some preset amount. Among the shortcomings of this approach are the fact that the entire frame is being differenced against a reference frame and the result is accumulated to make the motion determination. This is a problem in that changes over the whole image field can cancel out, thereby giving a false reading. For example, a given pixel could be brighter than its corresponding value in the reference frame by amount N, and another pixel could be darker than its reference value by -N. In such a case, the changes cancel out and significant motion might not be detected.

In addition, the simple differencing of corresponding pixels is insufficient for an effective motion detector. At the very least, the system must use the absolute value of the pixel differences — i.e., a difference of N is equivalent to a difference of -N. Similarly, the magnitude of differences is significant. The output of all video cameras have noise imposed upon the video signal. This means that the value reported for a pixel of an unchanging scene may vary plus and minus some amount simply due to thermally induced noise. Most existing methods do not compensate for this and so noise on the video signal contributes to false positive responses as well as to the need to desensitize motion detectors to the point where addition false negatives are generated.

The NetCam described here uses both a pixel difference threshold (which defines the degree (in absolute value) to which a pixel must vary from it's corresponding reference pixel in order to be considered different), and a frame difference threshold (which defines the number of pixels which must be different for a motion detection indication to be given).

VI. Conclusions

This paper introduces a new application-specific video compression technique that allows a compressed video stream to be switched (or gated) off whenever the current video field does not differ significantly from the last field sent. By (serially) performing per-pixel comparisons of incoming fields with the most recently transmitted field, it is possible to determine if the new field contains new information which should be sent, or whether the field is essentially the same and therefore need not be transmitted. This approach can result in an increase in compression ratios, in addition to that given by the native compression method, of a factor of 100 or more. In video surveillance applications, the use of gated compression, in conjunction with MJPEG compression, can yield an effective compression ration of over 5000:1.

It was noted that the video in many fixed-camera applications is highly redundant. In these cases, gated compression ensures that these devices generate absolutely no network traffic when there is no activity in the camera's field of view. However, when a significant event occurs, gating allows for transmitting at full frame rate, allowing the capture of highly detailed and complete video sequences, then returning to the quiescent state when the event ends. This results in highly bursty traffic from the devices, alternately transmitting at full rate and being silent. It is possible with to set parameters which define exactly how sensitive to differences in video images such a device should be. In fact, it is possible to set the gating mechanism's parameters such that no gating is done at all and every frame of video is passed through to the network. In addition, a data-size limiting mechanism can be used in conjunction with gating function to ensure that a video source does not exceed a given amount of data in a period of time.

Gated compression can be effective even when used with broadcast television sources. For example, in cases where 30 frame per second interlaced video has been converted from 24 frame per second film, a technique called "3-2 pull-down" is used[8]. This duplicates one field two additional times followed by the display of the next field, which is then duplicated once (resulting in the following type of pattern: aaabbcccdd...). When this type of video signal is presented to a device that uses gated compression, the gating mechanism removes the redundant fields and achieves an additional compression factor of 60%, with no loss in video quality (as all repeated fields are simply redisplays of the first field).

This paper describes a small, low-cost, Internet appliance that implements the gated compression scheme and illustrates the effectiveness of this approach in an example application. This device uses digital processing techniques to quickly, accurately, and inexpensively detect motion in video streams captured by a video camera. This device is shown to be capable of detecting small amounts of motion in a scene, as well as significant changes in motion in a constantly changing scene (by dynamically adjusting its sensitivity to the current level of background motion). The results of a series of experiments were presented to illustrate the effectiveness of the gated compression technique in an actual surveillance video application.

VII. Acknowledgments

The authors wish to thank the other members of the Sun Labs Media Systems Group who contributed to the design and implementation of the NetCam, including: Jerry Wall (firmware and bringup), Alan Ruberg (RTP/RTCP), and Lawrence Butcher and Marc Schneider (hardware).

Thanks is also due to Steve Kruschke of Sun's Corporate Security group for his contributions to the system requirements definition, and Ivan Sutherland of Sun Labs for suggestions which contributed to the improvement of the gating algorithm.

VIII. References

[1] W. B. Pennebaker and J. L. Mitchell.
JPEG Still Image Data Compression Standard.
Van Nostrand Reinhold, New York, 1993.

[2] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall.
MPEG Video Compression Standard.
Chapman and Hall, New York, 1997.

[3] J. O. Limb and J. A. Murphy.
"Measuring the Speed of Moving Objects from Television Signals."
IEEE Transactions on Communications, 23(4):474-478, April 1975.

[4] I. H. Witten, A. M. Moffat, and T. C. Bell.
Managing Gigabytes.
Van Nostrand Reinhold, New York, 1995.

[5] M. R. Macedonia and D. P. Brutzman.
MBone Provides Audio and Video Across the Internet.
IEEE Computer, April 1994.

[6] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson.
RTP: A Transport Protocol for Real-Time Applications.
Internet Engineering Task Force RFC 1889, January 1996.

[7] S. McCanne and V. Jacobson.
vic: A Flexible Framework for Packet Video.
Proceedings of ACM Multimedia, November 1995.

[8] K. Jack.
Video Demystified: A Handbook for the Digital Engineer.
HighText, San Diego, 1996.

[9] S. Kruschke.
Personal Communications.