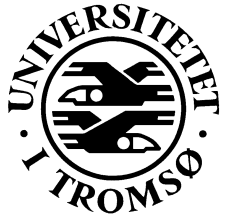

REPORT

Computer Science Technical Report: **94-16** **September 1994**

StormVideo – Digital Video in the Field of Meteorology

O. Nybø, G. Hartvigsen and D. Johansen

INSTITUTE OF MATHEMATICAL AND PHYSICAL SCIENCES
Department of Computer Science
University of Tromsø, N-9037 Tromsø, Norway, Telephone +47 77 64 40 41, Telefax +47 77 64 45 80

StormVideo – Digital Video in the Field of Meteorology

O. Nybø, G. Hartvigsen and D. Johansen

Department of Computer Science,
Institute of Mathematical Sciences,
University of Tromsø,
N-9037 Tromsø, Norway

Abstract

Visual observations constitute important input to different meteorological tasks. Previously, these observations were made by human observers and manually reported to the meteorologists. With the employment of video technology, this kind of observations can be automated. This paper presents one approach to visual weather observations. This involves software compression and transmission of digital video. The paper presents a modified version of the PVC algorithm and shows that this algorithm is well suited for software compression of color video for use as visual information in the field of meteorology.

Keywords: Video, weather forecasting, video compression, video transmission.

1 Introduction

Accurate weather forecasts are crucial for many activities in a modern society. Commercial industries such as airplane traffic, deep sea oil drilling and commercial fishing rely on the prognoses of meteorologists. Visual observations are an important source of information when the meteorologist develops a weather forecast. Rationalization and automation in society have swept away many sources of visual observation, such as meteorological outposts and manned lighthouses. On site visual observations in critical areas such as airports are expensive.

In the StormCast project (Harvigsen and Johansen, 1988; Johansen, 1993) the current focus is to construct a meteorological workbench. One feature in the StormCast meteorological workbench (Hartvigsen and Johansen, 1993) is to provide a tool for visual observation through the use of on site cameras. We have called this tool StormVideo (Nybø, 1993).

StormVideo is supposed to operate on-shore and off-shore in climatically extreme conditions. This, of course, put certain restrictions on the infrastructure, including low-bandwidth transmission lines, light-intensive remote controlled out-door cameras, etc. In addition, we have to use the current hardware and software platform in StormCast version 2.1.

These parameters give the following problem definition for the StormVideo project: *"How can remote out-door video cameras connected through low-bandwidth transmission lines be utilized to provide still pictures and full motion video from different geographical areas?"*

This problem can be further divided into digitizing analog video, compression of digital video and transmission of compressed video in an error prone network. This paper shows how this tool are implemented in software and hardware.

2 Video and video transmission

The StormVideo is based on existing video compression methods. The two most well-known video compression methods are MPEG (Le Gall, 1993) and H.261 (Liou, 1991). MPEG is an ISO standard for video transmission and storage. H.261 is a CCITT standard for compression of video for teleconference and video telephones. They are both based on discrete cosines transformations (DCT). They also both incorporate motion compensation. They both achieve high compression rates at fairly low loss of quality.

The problem with these two methods are that the compression is to slow in software implementations of these algorithms, even on workstations as HP 9000/735[1]. Several other techniques have also been proposed. These include among others vector quantization (Gersho and Ramamurthi, 1982), fractal compression (Fisher, 1992) and sub-band coding (Woods and O'Neil, 1986). None of these non-standard methods have been widely used and are still in an experimental stage.

3 StormVideo

StormVideo provides the user with two modes of operation, camera control mode and observation mode. Camera control mode will be used when the user wants to remotely change the camera angle. In this mode the user will need high frame rate to get feedback on current camera position while the camera is moving. Video quality is not important in this mode, but the user must be able to recognize what he sees. In observation mode the video quality is the most important parameter. Contrasts are very important, e.g., to see thin skies on a sunny

1. HP 9000/735: 64 Mbyte RAM, 840 Mbyte disk, 124 MIPS, 40 MFLOPS.

day. Due to quality considerations, we have chosen to use lossless compression in observation mode.

Figure 1 shows connections in a typical StormCast network. In this network there will be video servers at meteorological important sites. A video camera will be connected to each video server. The camera can be controlled remotely from any workstation in the StormCast network and visual information from any camera can be transferred to any workstation (video client) in the network. The video signal from the camera will be compressed by the video server before it is transmitted to a video client. The video client decompress the video signal when it is received. A modified version of the PVC algorithm (Huang et al., 1993) for video compression are used. Our results show a much higher compression speed than for instance MPEG (Le Gall, 1993) or H.261 (Liou, 1991).

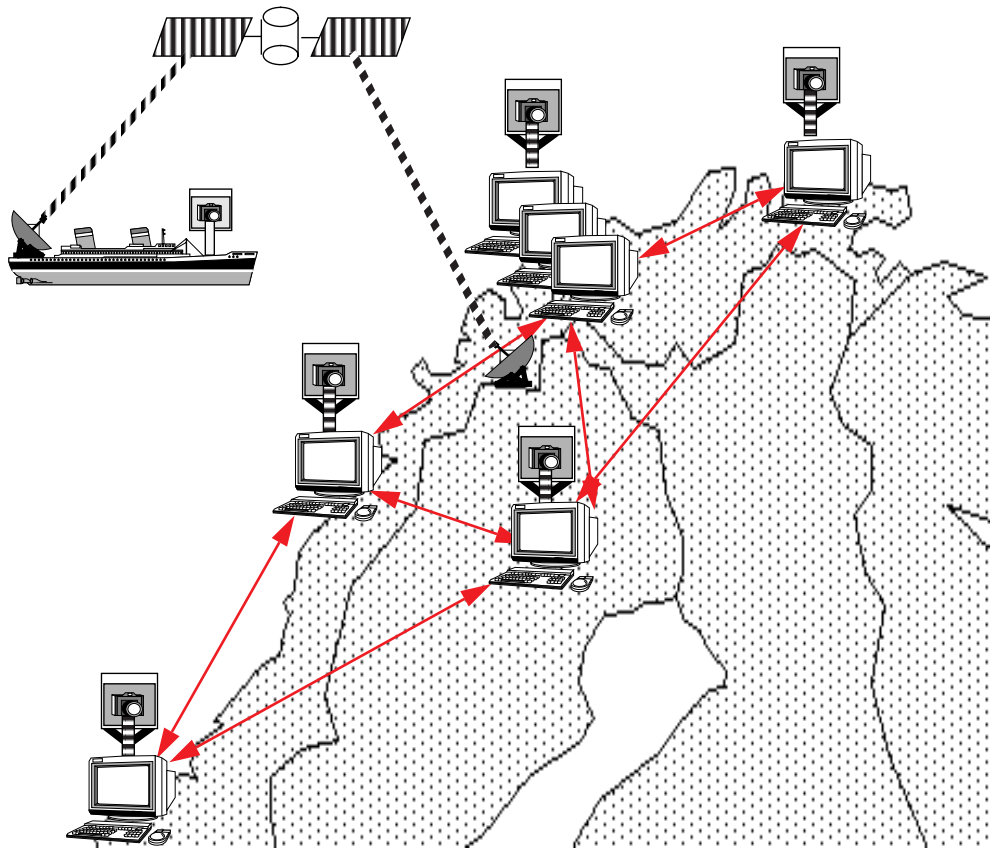

FIGURE 1. Connections in a typical StormCast network (Hartvigsen and Johansen, 1993).

4 Construction of the StormVideo

In this section the construction of the StormVideo is described. The StormVideo is implemented and tested on HP 9000/720[1] and HP 900/735[2] workstations running HP-UX 9.01. We have used RasterOps videolive card for the grabbing of images from an analog video camera. Currently we use a SONY CCD-808E Hi8 indoor camera, but in future versions this will be replaced by a motorized outdoor camera. However, this has no significant influence on the construction and experimentation. The application interface is implemented in X11 release 5/Motif 1.2.

4.1 System architecture

We have chosen to design StormVideo as 6 separate modules. Each module communicate with the next module through a clearly defined module interface. We have found this strongly modularized approach very efficient for experimenting with the different modules.

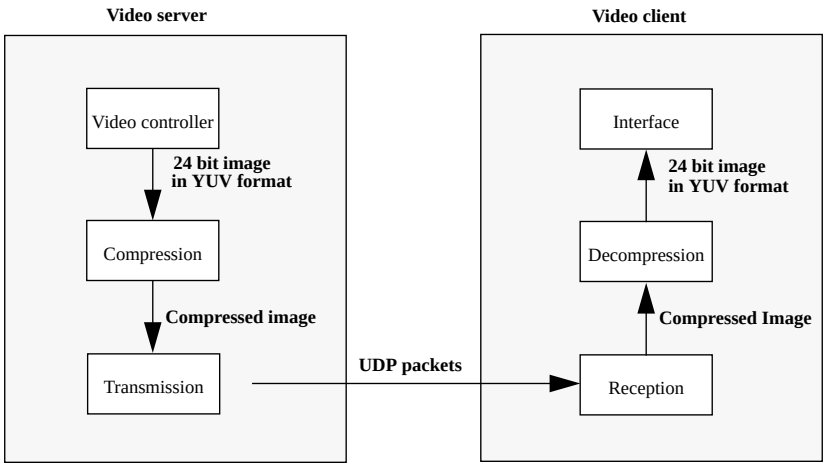

FIGURE 2. High level architecture of StormVideo

Figure 2 shows the 6 modules of StormVideo and the data flow in normal operation. The video controller reads images from the RasterOps card and transform them into YUV color space. The YUV transformation includes a 4:1 subsampling of the chrominance signal. The video controller also sends control information to the camera such as steering of camera angle and zoom in and out. The compression module compresses the image. The transmission and reception module handle the data transfer on the available network connection. These two modules also do some error handling. The decompression module decompresses images and handles errors not detected in the reception module. The decompress module

1. HP 9000/720: 32 Mbyte RAM, 840 Mbyte disk, 57 MIPS, 17 MFLOPS.
2. HP 9000/735: 64 Mbyte RAM, 840 Mbyte disk, 124 MIPS, 40 MFLOPS.

also handles requests for video storage. The interface module receives decompressed images and displays them on the screen.

4.2 Compression

We have modified the PVC algorithm (Huang et al., 1993). The adaptive Huffman coding in the final step of the PVC encoder is replaced with a Lempel-Ziv coder (Ziv and Lempel, 1977). This was done to increase the speed of the coding process. The compress module inserts CRC data into the compressed data. The decompress module uses this to perform CRC check on the data to detect errors not detected by the reception module.

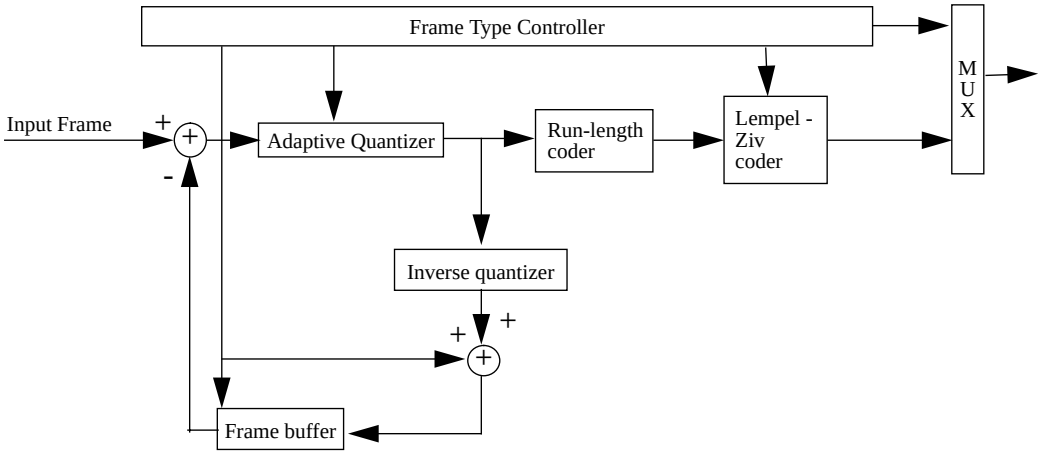

FIGURE 3. The block diagram of the modified PVC encoder

The decoding process is the inverse of the encoding process shown in Figure 3.

4.3 Communication

The network communication in the present version StormVideo is based on UDP/IP. Local IPC is implemented using shared memory because this is a very effective IPC mechanism (Haviland et al., 1987). The communication module does some error detection as opposed to PVC. This is done because the transmission module has to know the length of the incoming frame to read it from the shared memory. This knowledge is passed to the reception module which then is able to check the length of each incoming frame.

4.3.1 Protocol

Figure 4 shows the communication protocol of StormVideo. Normal operation involves only m_x messages. The m_x messages contain image data, which is sent from the video controller module on the video server to the interface module on the video client. When an exception

occurs (e.g., an error is detected or the user wants to change camera direction), a control message (c_x) is sent from the module where the exception arose to the appropriate module.

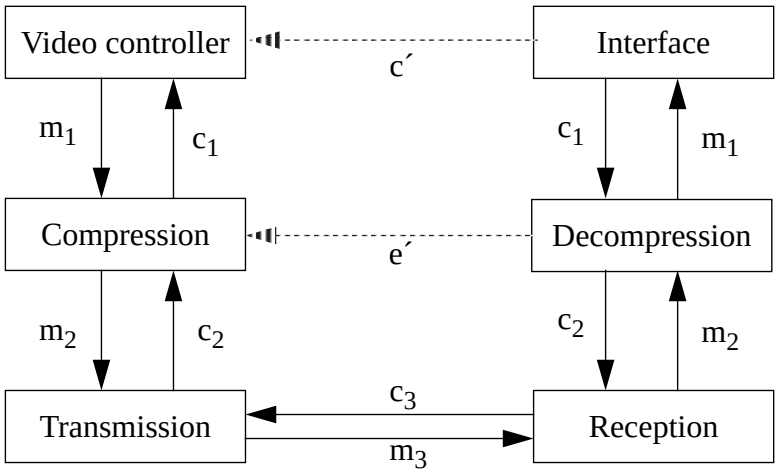

FIGURE 4. The communication protocol of StormVideo

4.3.2 The m_x data format

The m_1 messages contain uncompressed or decompressed messages. The data format is as shown in Figure 5. The first 4 bytes contains the width and height, two byte for each. The angle of the camera is stored in the next two fields. The horizontal angle is stored in v_1 and the vertical angle in v_2. The last field is the frame itself.

2 byte	2 byte	2 byte	2 byte	Width * Height * 1.5 byte
Width	Height	v_1	v_2	Frame on YUV format

FIGURE 5. The m_1 data format.

The format of m_2 and m_3 is almost the same and is shown in Figure 6. The m_3 messages have an extra field to indicate wether the packet is the start of a new frame or not. This is necessary because a m_2 message containing a frame may be split up in smaller m_3 messages which fits into the packet size of the transport protocol being used. (We have used UDP/IP, but ISDN will also be considered to be used in future versions). The start field of the m_3 message is used for detecting packet loss in the reception module. The type field indicates the frame type. This may be one of the types defined in PVC (refresh, coarse, medium or

fine). The length field indicates the total frame length. The last field, the data field, contains the compressed image data.

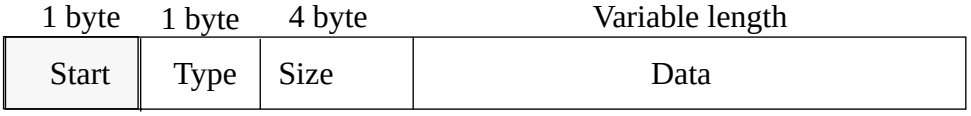

FIGURE 6. The m_2 and m_3 data formats. (The shaded field is present only in m_3 messages.)

4.3.3 The c_x data format

There are four types of control messages (c_x). These include two requests from the user interface to the video controller (indicated as c´ in Figure 4), one request from the user interface to the compress module, and error messages from the decompressor (indicated as e´ in Figure 4) or the reception module to the compress module. The requests from the user interface include requests for change of size of video frames, requests for change of camera angle, and start/stop quantization. The first two control messages are received by the video controller and the last is sent to the compress module. Figure 7 shows the data format of the c_x messages. The first field indicates the type of control message. The data field holds the information needed for each type of control message. When change of size or angle is requested from the video controller the data field will hold new width and height or new horizontal and vertical angle respectively. The data field is not used for the two other message types.

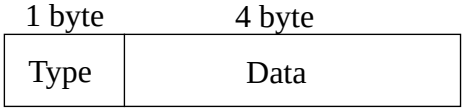

FIGURE 7. The c_x data format

4.4 Local IPC

The vertical lines in Figure 4 indicate the local IPC of StormVideo. Local IPC in StormVideo is implemented using shared memory. A shared memory segment for each communication line is creates. This segment is divided into slots which can hold one message each. We have implemented access control for the shared memory for the two main message types (control and data messages) differently.

4.4.1 Local m_x message exchange

The local m_x message exchange is implemented as the classic producer-consumer problem. Consistent access control for the exchange of m_x messages is implemented using semaphores (Dijkstra, 1968). We use two semaphores for each slot in the memory, *free* and *used*. *Free* is initialized to 1 and used to 0. To insert a message into a slot the sender performs a *down* (p) operation on the *free* semaphore of the memory slot in question, then the message is stored in the memory slot and a *up* (v) operation is performed on the *used* semaphore. The receiver makes the following steps to fetch a message: *down* on *used* semaphore, read the message from the memory slot and *up* on the *free* semaphore. In this way a process will only block if the specific memory slot is being used by the other part of the message exchange.

4.4.2 Local c_x message exchange

Since most of the exchanged messages in the application will be m_x messages we have chosen to implement the exchange of c_x as exceptions. When a module wants to send a c_x message to another module it enters the message into the shared memory and sends a signal to the receiver which then read the message from the shared memory. Since signals may be lost, we compensate for this by checking the incoming m_x message to see if the request has been fulfilled. If the c_x message was lost the sender will time-out and retransmit the message. The m_x messages cannot be sent the as described in section 4.5.1 because an interactive user interface cannot block to receive an incoming message. Nor can signals be used because sending signal to an X process may cause problems. Instead, a method similar to the signaling for exchange of c_x messages is used. We use a pipe and set this up to generate an X event each time the sender (decompress module) write to this pipe. The interface module reads the shared memory segment when it receives an X event on the pipe.

5 Results

This section is divided in two parts. Firstly, the performance of the system is presented. Then the compression ratios obtained are presented. The video quality has by Norwegian meteorologists been judged as good enough for feedback during camera movement.

The first test goal is to locate the bottlenecks in the application. This information is necessary to be able to improve the performance. The test shows where the effort should be put in order to improve the application.

The second test goal is to confirm the design goals. This includes examination of picture quality.

5.1 Method

The performance testing are conducted with the use of the library function gettimeofday(). This function have been used to measure the time to the different segments in the modules. Within the functionality available, we have chosen three kind of sequences.

The first measurement represents transmission of a still picture inside the laboratory. This is similar to having a static camera. The second measurement is based on full-time movie (generated from a TV). This is similar to movement of both camera and object. The third measurement shows the results from a moving camera.

5.2 Performance

Table 1 shows the performance of the user interface. The 91% of real-time[1] is good enough for proper feedback when the camera angle is changed.

TABLE 1. Dithering and display speed of 354x288 pixels images.

msec/frame	frames/sec	real-time
44 ms	22.7	91%

TABLE 2. Decompression (S=16, Z=16, 354x288 pixels, ca 40 Kb compressed size)

msec/frame	frames/sec	real-time
184 ms	5.4	22%

Table 2 shows that the decompression achieve 22% of real-time. This is acceptable for camera control. The user will be able to see how the camera moves. Other tests done, but not reported in this paper, have shown that most of the time (43%) is spent on the Lempel-Ziv decoding. We believe that this can be optimized to run faster than the current version.

TABLE 3. Compression (S=16, Z=16, 354x288 pixels, ca 40 Kb compressed size)

msec/frame	frames/sec	real-time
403 ms	2.48	10%

Table 3 shows the results for compression of video. This module achieves only 10% of real-time. The Lempel-Ziv coding account for 72% of the time spent. Again the Lempel-Ziv coder is not fully optimized and some gains could be achieved doing so.

TABLE 4. Grabbing and YUV conversion (354x288 pixels)

msec/frame	frames/sec	real-time
276 ms	3.62	14%

Table 4 shows that the grabbing is rather slow. The YUV conversion is optimized by removing all multiplication using table lookup instead.

1. We define real-time as 25 frame/sec

5.3 Compression ratios

We have tested to types of video sequences which will be typical in operational use of StormVideo. The two sequences are still picture and moving camera. The result of the still picture sequence shows what the compression ratio is likely to be in observation mode. The moving camera shows some indication of the compression ratio in the camera control mode. The results presented here are created using a hand hold camera indoors, but we believe that the results are close to what we can expect from a motorized outdoor camera. A motorized outdoor camera will of course move more steadily, but we believe that our test are a good approximation of operational use. Each test consist of a sequence of 20 frames where the first frame is a refresh frame and the following are coarse frames.

Figure 8 shows the results for loss-less compression. We can see that the compressed size for still images is low. This compression will be used in observation mode.

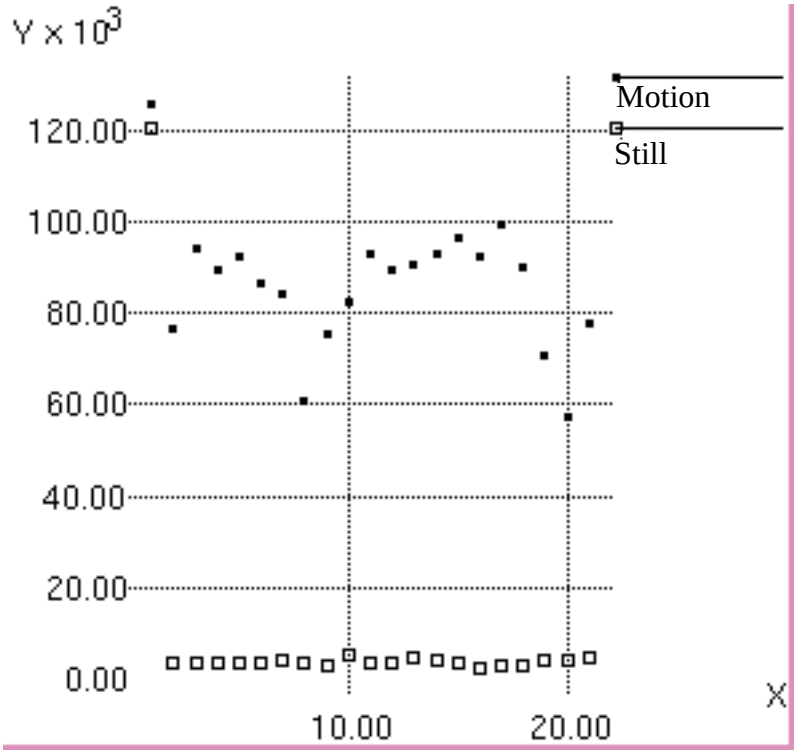

FIGURE 8. Compression of 352x288 images (S=1,Z=0)

Figure 9 shows the compressed image size for frames when lossy compression is used. For motion video the size is about 25 Kb. This is still to large for transmission of live video on 2 type B ISDN lines. The bits allocated for the run-length coding in the current version can be better tuned to make the compression better. Meteorologists consulted have judged the image quality at S=16 and Z=16 to be well good enough for camera control mode. To

increase compression ratio we could raise the S and Z values even more, but much higher than S=24 and Z=24 is not possible while still getting recognizable images.

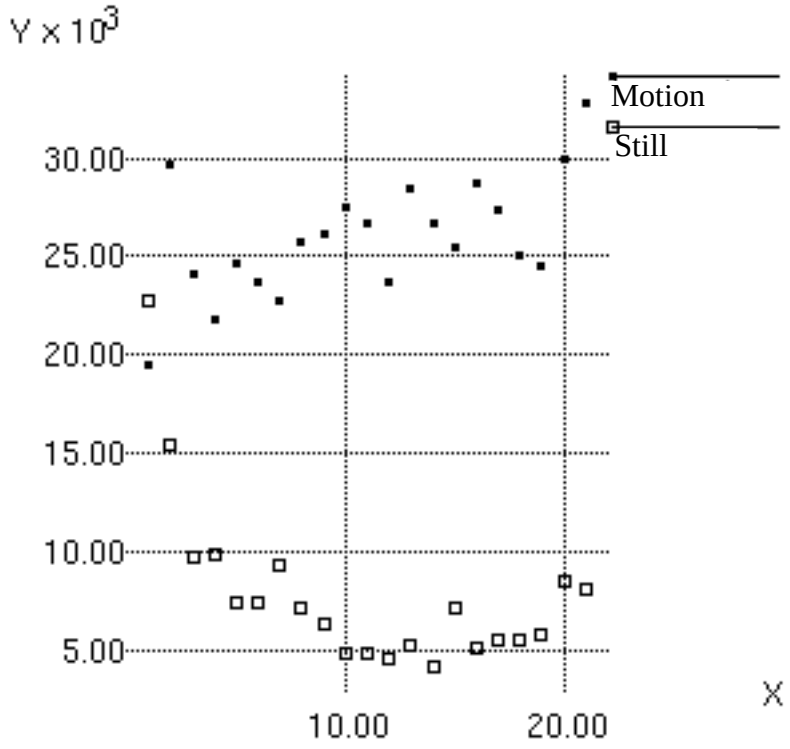

FIGURE 9. Compression of 352x288 images (S=16,Z=16)

6 Discussions and open problems

A peak out of the window will enable a trained meteorologist to give a fairly precise short-term forecast. He can se clouds which indicates presence of weather fronts in the area. The shape of the clouds gives information on the stability of the air above the observation point. Numerous observation in a large area by use of e.g. StormVideo will provide information on among others movement of fronts and location of showers.

The system performance in the current version is about 2 images pr. second in a 10 Mbit/sec ethernet. In most cases, this is acceptable results for a meteorologist. The user will be able to see how the camera moves. However, much can be done to improve the total system performance. Most important is the grabber implementation. The low frame rate of the grabber degrades the rest of the application. The PVC algorithm is very dependant on correlation between following images in a sequence. We have seen that The Lempel-Ziv coding is

responsible for a large amount of processing time. Both compression rates and system performance would be increased if the frame rate of the grabber is increased. We use Xv for grabbing of images from the card. This is a very cumbersome solution, because Xv is not designed for our needs. Xv can not deliver a RGB frame in a memory area, but put it into an X window structure. We had to use Xv because of lack of technical information on the videocard. A faster card which could grab an image directly into a memory area at a speed of about 15 frames/sec would probably increase overall system performance. The PVC implementation has shown advantages such as non blocking effects like MPEG or H.261 which reduces contour lines. Furthermore, our results show that it is fast, but compression ratio in the current version is not as good as we want.

7 Concluding remarks

StormVideo is an example of visual weather observations on workstations. The modular design and well-defined interfaces between the different modules have proved to be an acceptable architecture. The use of shared memory as inter-process communication has reduced the computational costs. Another observation has been that the PVS compression has appeared to be deeply dependent on a high correlation of the pictures in the video stream. This requires an high picture frequency. Thus the method will not be usable if the videodata is transmitted over an B-ISDN line since the bandwidth will reduce the system. This is especially through if the camera is moving. The feedback from the Norwegian meteorologists indicates that this kind of visual weather observations will be a helpful tool for the meteorologists in the near future.

8 Acknowledgments

We would like to thank meteorologist Bård Fjukstad for his help with defining the requirements of the application and for his evaluation of the results.

References

Dijkstra, E.W. (1968). “The structure of the “THE” multiprogramming system.”, Comm. ACM, Vol. 11, No. 5 (May), pp. 341-346.

Fisher, Y. (1992). “Fractal Image Compression”, Siggraph ‘92 course notes.

Gersho, A., and Ramamurthi, B. (1982). “Image Coding using Vector Quantization”, Proc. of Int. Conf. ASSP, Paris 1982, pp. 428-431.

Hartvigsen, G. and Johansen, D. (1993). “StormCast meteorological workbench”, Project description, June 1993, Department of Computer Science, Institute of Mathematical and Physical Sciences, University of Tromsø, Norway.

Havilland, K.F., Salama, B. (1989). “UNIX System Programming”, Addison-Wesley,

Huang, H., Huang, J. and Wu, J. (1993). “Real-Time Software-Based Video Coder for Multimedia Communication Systems”, ACM Multimedia ‘93, Proceedings, pp 65-73.

Johansen, D. (1993). “The StormCast Applied Approach to Distributed Computing”, Ph.D thesis, Department of computer science, institute of mathematical and physical sciences, University of Tromsø, Norway.

Le Gall, D. (1991). “MPEG: A Video Compression Standard for Multimedia Applications”, CACM, Vol 34 No 4, April 1991, pp 46- 58.

Liou, M. (1991). “Overview of the px64 kbit/s Video Coding Standard”. CACM, Vol 34, No 4, April 1991, pp 59- 63.

Nybø, O. (1993). “Visual Weather Observations - transmission and storing of digital video”, Graduate civil engineer thesis, Autum 1993, Department of Computer Science, Institute of Mathematical and Physical Sciences, University of Tromsø, Norway. (in Norwegian).

Woods, J.W. and O’Neil, S.D. (1986). “Subband Coding of Images”, IEEE Trans. on Acoustics, Speech and Signal Proc., ASSP-34, Oct. 1986, pp. 1278-1288.

Ziv, J. and Lempel, A. (1977). “A universal algorithm for sequential data compression”, IEEE Trans. IT-23, 337-343.