Report Number: WUCS-90-12

1990-01-01

A Simulation Testbed for Image Compression Algorithms

Andrew Francis Laine

This paper presents an overview of the design and development of a real-time (30 frames/sec) simulation testbed for evaluating and comparing image compression algorithms. The system was motivated by the need to visualize the performance of a novel compression algorithm when operating on moving pictures originating from "live" video sources. The simulation utilities are designed to exploit the parallelism of a Pixar Image Computer and high-throughput of a parallel disk assembly. The design of two key utilities are discussed: (1) A program to format precomputed four channel (RGBA) 256 X 256 color frames onto a parallel disk assembly. (2)...
Read complete abstract on page 2.

A Simulation Testbed for Image Compression Algorithms

Andrew Francis Laine

Complete Abstract:

This paper presents an overview of the design and development of a real-time (30 frames/sec) simulation testbed for evaluating and comparing image compression algorithms. The system was motivated by the need to visualize the performance of a novel compression algorithm when operating on moving pictures originating from "live" video sources. The simulation utilities are designed to exploit the parallelism of a Pixar Image Computer and high-throughput of a parallel disk assembly. The design of two key utilities are discussed: (1) A program to format precomputed four channel (RGBA) 256 X 256 color frames onto a parallel disk assembly. (2) A parallel program to read the stored frames off the disk and display them on the Pixar screen at video rates up to 30 frames/second.

A SIMULATION TESTBED FOR IMAGE COMPRESSION ALGORITHMS

Andrew Francis Laine

WUCS-90-12

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

ABSTRACT

This paper presents an overview of the design and development of a real-time (30 frames/sec) simulation testbed for evaluating and comparing image compression algorithms. The system was motivated by the need to visualize the performance of a novel compression algorithm when operating on moving pictures originating from "live" video sources. The simulation utilities are designed to exploit the parallelism of a Pixar Image Computer and high-throughput of a parallel disk assembly.

The design of two key utilities are discussed: (1) A program to format precomputed four channel (RGBA) 256 x 256 color frames onto a parallel disk assembly. (2) A parallel program to read the stored frames off the disk and display them on the Pixar screen at video rates upto 30 frames/second.

Acknowledgments: This work was supported by Southwestern Bell Telephone and SBC Technology Resources Incorporated. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of Southwestern Bell Telephone or SBC Technology Resources Incorporated.

1. Introduction The objective of our work has been the development of a real-time animation testbed for the evaluation and comparison of image compression algorithms. We have designed and implemented a strategy to exploit the parallel transfer disk of the Pixar Image Computer and associated hardware/software interfaces, seeking real-time (24-30 frames/second) animation capability for 256 x 256 RGB color frames. The system was motivated by the need to visualize the performance of a novel compression algorithm when operating on moving pictures originating from "live" video sources. A significant advantage of this purely digital approach is that the number of analog to digital (and D/A) conversion steps needed to show a "movie" is minimized. Thus noise and other artifacts that might affect the visual quality of a particular compression algorithm are prevented from entering the production loop. The digital simulation utilities are designed to exploit the parallelism of a Pixar Image Computer and high-throughput of a parallel disk assembly.

The design of two key utilities are discussed: (1) A program to format precomputed four channel (RGBA) 256 x 256 color frames onto a parallel disk assembly. (2) A program to read the stored frames off the disk and display them on the Pixar screen at video rates upto 30 frames/second.

2. System Overview

At present, there are two components to the system. Input to the first component "mk_movie", consists of a file containing previously digitized frames (either ca from a frame-grabber or precomputed synthetically). Each frame is stored in row major order, the first scanline mapping to the upper righthand corner of the Pixar screen. Each color channel (RGBA) is stored sequentially within the file. The maximum size of each color frame is limited to 256 x 256 pixels.

The program "mk_movie" creates a single fast-disk file on the Pixar Fast Disk, large enough to contain the entire set of precomputed frames. Each Fast Disk assembly may store approximately 1.2 Gigabytes of data. Thus animations of duration 90 seconds are possible (256x256x4 pixels/frame) x (30 frames/second). If the input is monochrome, the frame is copied to the R,G and B channels and treated like a color frame (the alpha channel is blanked).

By creating a single fast-disk file, such that all frames are contained within one tile, we need only make one *read* system call to retrieve the entire set of frames from the fast-disk. Thus we avoid the system overhead experienced if multiple *reads* were used to retrieve each frame (reguardless of whether the frames were stored as individual tiles within a fast-disk file or stored as a set of independent fast-disk files).

This approach was motivated by an earlier investigation that showed sustained throughputs (of 8 MB/second) could only be obtained by minimizing the setup costs for each frame. In particular, while sustained throughputs of 8.2 MB/second were measured when reading frames larger than 4MB, a greater than expected penalty (50%) was observed for reading smaller frames (ranging in size from 1.0 - 0.25 MB). Thus, for smaller sized frames, measured throughput was 4.2MB/second (on the average), insufficient to support real-time animation. The degradation in performance was principally due to the cost of setting up the HSI (High Speed Interface) and starting up a Chap program associated with each read operation (not in reading the data off the the fast-disk itself).

Our present strategy, requires exactly one setup per animation sequence, and provides the sustained throughput required to drive the entire animation at video rates. Recall that we treat the entire animation (a sequence of frames) as one logical "image". Where the sequence of frames has been precomputed and reformatted for contiguous storage on the Fastdisk.

This approach requires custom programming of the Chap (Channel Array Processor), to map the continuous flow of pixels from the Fastdisk (actually the Yapbus) onto a predefined subrectangle in video memory. The program "fd_movie", provides the framework for running the animation. It sets up the HSI (High Speed Interface) to read from the Fastdisk and loads a Chap program (mloop) to read pixels from the Yapbus. The actual video display may be displaced by an offset within the framebuffer.

To show a movie, the Chap program "mloop" is first loaded into the Pixar Image Computer and runs asyncronously, listening for pixels to arrive from the Yapbus. Next, the HSI is configured to receive pixels from the Fastdisk and route them onto the Yapbus in 8-bit RGBA format. Thus, four bytes arriving from the fastdisk, yields one pixel out on the Yapbus. In addition, the program must instruct the HSI how many scanlines to read, and the length of a scanline. Finally, the fastdisk controller is instructed to seek to start of the first tile in the image and then to read the entire tile from the fastdisk.

The Chap program then collects a scanline worth of pixels from the Yapbus and writes the scanline onto the framebuffer at a user specified offset. An internal counter is incremented for each scanline displayed and the process is repeated until all the scanlines for a frame are copied to the framebuffer. The program then waits until the next vertical blanking period and begins to draw the next frame. Figure 1 shows an overview of the data flow from the fastdisk assembly, HSI, CHAP, and finally to the frame buffer for video display.

3. Future Work

While the present program assumes all frames are the same size, and does not provide for the transmission of variable sized "packets", it is possible to attached a small header in front of each frame specifying is offset within the framebuffer and it's size (in rectangular coordinates). Thus, it should be feasible to support variable sized (rectangular) sub-frames within the same animation.

A new Chap program capable of routing larger size frames (512 x 512) of monochrome images should be feasible. In this program, four consecutive black and white frames could be kept within each color (RGBA) frame stored on the fastdisk.

Figure 1. **Diagram of the Visualization System**
A Simulation Testbed for BPN Compression Algorithms