The Light Field Video Camera

Bennett Wilburn, Michal Smulski, Kelin Lee and Mark A. Horowitz

Computer Systems Lab, Electrical Engineering Department, Stanford University

ABSTRACT

We present the Light Field Video Camera, an array of CMOS image sensors for video image based rendering applications. The device is designed to record a synchronized video dataset from over one hundred cameras to a hard disk array using as few as one PC per fifty image sensors. It is intended to be flexible, modular and scalable, with much visibility and control over the cameras. The Light Field Video Camera is a modular embedded design based on the IEEE1394 High Speed Serial Bus, with an image sensor and MPEG2 compression at each node. We show both the flexibility and scalability of the design with a six camera prototype.

Keywords: image based rendering, CMOS image sensor arrays

1. INTRODUCTION

Over the past few years, the ability of Image Based Rendering (IBR) techniques to create photorealistic images of real scenes has generated great interest in capturing environments from multiple viewpoints. The Light Field Rendering and Lumigraph work both involved one moving camera recording images of a static scene .[1, 2] For real scenes where no depth information is known, these methods assume a constant depth from the camera plane to the object plane and require many images from closely packed camera locations to produce alias-free results.[3] When approximate geometry is incorporated into the rendering process, View Dependent Texture-Mapping provides another IBR technique to generate novel views of objects using a small number of captured images.[4] Recently, these methods have been extend to dynamic environments. The 3D-Room at Carnegie Mellon uses 49 video cameras to generate three dimensional scene models for rendering different views of dynamic scenes.[5] Image Based Visual Hulls uses silhouette based methods to derive approximate object geometries from a small number of cameras.[6]

At the same time that IBR methods improve, semiconductor technologies are also advancing. Ever increasing processing power is available for ever decreasing cost, power, and chip area. We have also witnessed the advent of CMOS image sensors, which are inexpensive and easy to use because of their digital interface. Furthermore, because they are manufactured in a CMOS process, processing power can be placed on the sensors itself. This raises the question, what can we do with many inexpensive CMOS image sensors, equally inexpensive optics, and a lot of processing power? Can we use more cameras of lesser quality to enable more robust IBR algorithms? Can we use clusters of inexpensive imagers and processors to create virtual cameras that outperform real ones?

In this paper, we describe a CMOS image sensor array, called the Light Field Video Camera, for investigating these questions. The device is designed to record a synchronized video dataset from over one hundred cameras to a hard disk array using as few as one PC per fifty image sensors. It is intended to be flexible, modular and scalable, with much visibility and control over the cameras. The Light Field Video Camera is a modular embedded design based on the IEEE1394 High Speed Serial Bus, with an image sensor and MPEG compression at each node. We show both the flexibility and scalability of the design and present a six-camera prototype implementation of the array.

Further author information: (Send correspondence to B.W.)
B.W.: E-mail: bennett.wilburn@stanford.edu, http:www-vlsi.stanford.edu/˜wilburn, M.S.: E-mail: msmulski@reardensteel.com, K.L: kelin@stanford.edu

2. PRIOR WORK

Nearly all camera acquisition systems for IBR applications have used a very small number of cameras, typically five or less. Light Field Rendering, the Lumigraph and Concentric Mosaics all used one camera moving around a static environment to sample the scene's light field. More recent IBR work has focused on systems capturing raw uncompressed video data from a small number of cameras. One notable exception, and probably the existing system most similar to our design, is the 3D-Room at Carnegie Mellon.[5] This facility can record uncompressed video from each of 49 NTSC cameras, but requires 17 PC's with three frame grabbers each, a time code generator, and 49 time code translators. The total recording time is also limited to less than one minute.

3. HARDWARE GOALS AND OVERVIEW

3.1. Design Goals and Tradeoffs

Our fundamental goal for the camera array is to record a video dataset from a large number of cameras. We require 640 by 480 pixel resolution, 30 frames per second synchronized progressive scan video. In order to properly orient the cameras, we also need to display live video from any of the cameras. We would like this tool to enable a broad range of research efforts, so ideally it will be flexible. For example, we need low level control over camera characteristics like exposure time and color correction, and we must be able to place the cameras in many different configurations—tightly packed, spread out, and so on. Furthermore, we want a design that scales easily and does not require extra machinery like numerous PC's or specialized video equipment. Finally, we would like a solution that is relatively inexpensive per camera.

In pursuit of these goals, we made three major design decisions. First, we elected to use CMOS image sensors instead of CCD's. Second, we decided to capture MPEG2 compressed video from each camera instead of raw digital video. Third, we chose to use the IEEE1394 High Performance Serial Bus for communication between the cameras nodes and a host PC. For the rest of this section, we explain the reasoning behind those choices. In the next section, we proceed to a more detailed explanation of the Light Field Video Camera array hardware and operation.

3.2. CMOS Image Sensors

Although CMOS image sensors in general have worse noise performance than their CCD counterparts, CMOS image sensors are suitable for this design for many reasons. Unlike CCD's, they do not require any uncommon voltage supplies or clock signals, or any extra circuitry like analog to digital converters. CMOS chips can incorporate digital processing and interfaces, which makes them much easier to use and to control. Many CMOS image sensors on the market, for example, have an I2C interface for controlling and monitoring parameters like exposure time, the gain of different color channels and gamma correction. Such control is critical for our application because we have constraints on our cameras that the average user does not. We combine data from multiple cameras, so we need to be able to measure and control the variation in response across all the sensors. With CMOS image sensors, we can easily set appropriate exposure times and color gains, turn off gamma correction, white balancing and auto-exposure algorithms, and so on. Finally, for simplicity and low cost, we use single chip, color sensors with Bayer pattern color filters.

3.3. MPEG2 Video Compression

The major tradeoff we made in this design was to store MPEG 2 compressed video instead of raw video from each camera. The high bandwidth of raw video taxes two resources—the bandwidth of the connection from the cameras to any storage or processing devices, and the total storage capacity of the system. Assuming one byte per pixel for Bayer pattern data, a 640x480 pixel, 30 fps progressive scan camera generates 70Mb/s of data. Although the theoretical maximum transfer rate for the PCI bus is 132MB/s, 80MB/s is a more typical sustained rate, meaning that the raw data from about nine cameras could be transferred to a PC over a PCI interface. Of course, this assumes that the data will be stored in the PC's DRAM and not written to a hard disk, severely limiting the amount of data that can be captured. If the data is output to a disk drive, again over the PCI interface, then the throughput of the system is halved again, and data from only four cameras can be acquired. Finally, the total bandwidth to disk to save raw video from one hundred sensors would be close to 1GB/s. This limits the total duration of video that can be stored and requires many hard drives in parallel to supply the needed bandwidth.

Using MPEG2 video compression, we can reduce the output bitrate of our cameras from 70Mb/s to anywhere between 1Mb/s and 15Mb/s. Assuming a relatively standard value of 5Mb/s constant bitrate MPEG2 video, sixty cameras would create a total bandwidth of only 300Mb/s, or 38MB/s, so storing to disk through a PC is feasible. The disadvantage of using MPEG2 compressed data, obviously, is that the stored images will suffer from compression artifacts which could interfere with IBR or vision algorithms. Although we have to carefully investigate this issue, we believe it will not be too significant. We do not move the array while we film, so the background doesn't change, resulting in fewer artifacts at a given bitrate. We are also free to double the camera bitrate to 10Mb/s and record from thirty cameras to each PC in our system. In order to measure the effects of compression artifacts on our algorithms, we designed the array to store multiple synchronized raw video frames from all the cameras to their local memory for subsequent download. Locating features with subpixel resolution for geometric calibration also requires raw images.

3.4. IEEE1394 High Performance Serial Bus

The IEEE1394 High Performance Serial Bus is a natural choice for our array because of its guaranteed bandwidth for streaming applications. The IEEE1394 bus currently accommodates maximum data transfer rates of 400Mb/s, but more importantly, a default of 320Mb/s of that bandwidth is designated for isochronous transfers–bandwidth that is allocated on the basis on 63 "channels" and is guaranteed to be available every 125μsec IEEE1394 cycle. The IEEE1394 bus is also well suited for a modular, expandable design. Each IEEE1394 bus can have 63 nodes, one of which will be our host PC, meaning that we can put 62 cameras on a single bus. The cameras can each be assigned an isochronous channel and guaranteed 1/62 of the total bus bandwidth each cycle. Assuming a typical 5Mb/s constant bitrate MPEG2 compressed video from each camera, we can theoretically put the maximum of 62 cameras on each IEEE1394 bus. If we do not quite reach this due to overhead at the PC or possibly other effects, we have the option of reducing the bitrate or the number of cameras connected to the bus.

The IEEE1394 bus also provides flexibility in where we locate our cameras. IEEE1394 devices can be connected by cables as long as 4.5m, so we have the option of packing our cameras together as tightly as they will fit or spreading them out over relatively large distances. Only sixteen hops between devices are allowed on one bus, so spans greater than 72m would require more than one bus. We envision one PC per IEEE1394 bus, but one could use a machine with separate PCI busses to handle more than one IEEE1394 bus at a time.

4. DETAILED HARDWARE DESCRIPTION

4.1. Overview

The Light Field Camera Array is based on custom made camera nodes connected to each other and a host PC via an IEEE1394 High Performance Serial Bus. Each camera node is an embedded system with a color CMOS image sensor, an MPEG2 video encoder and a three port IEEE1394 interface. To keep the cameras synchronized, a common clock signal is distributed to all of the cameras. We also route a reset signal and two general purpose triggers to all the boards. Each camera receives these signals from a parent node and can send them to two children. Camera arrays with less than 62 nodes fit on one IEEE1394 bus, and the topology for the signal connections can mirror the 1394 network. We connect our nodes in a tree topology because the IEEE1394 bus is limited to 16 hops between devices. Arrays with over 62 cameras require multiple IEEE1394 busses. We will use two or more PC's as hosts, each managing a separate bus. In this case the clock, reset and triggers must be routed between the busses as well as to all the boards in a given bus.

4.2. Block Diagram

Figure 1 shows a block diagram of a camera node. Each node consists of two custom printed circuit boards, one with just the image sensor and passive components, and another with everything else. The image sensors boards are very small, only 54x30mm, so we can pack them very tightly together if we so desire. They are connected

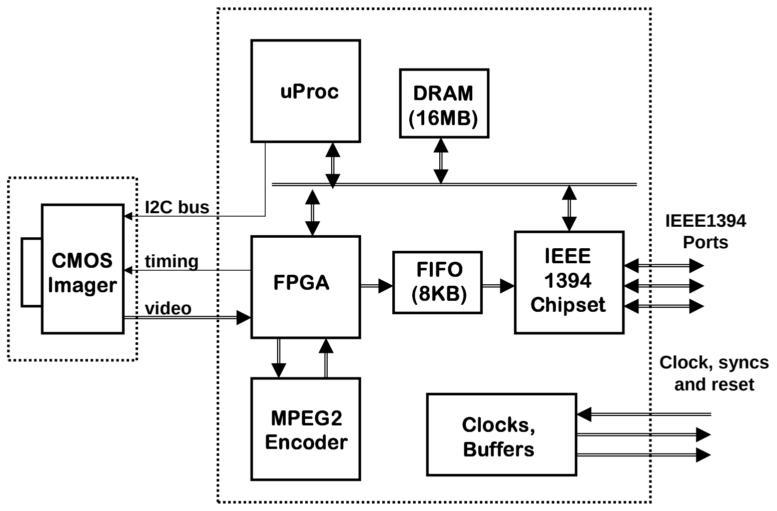

Figure 1: Block diagram of camera node.

to the main boards by meter long ribbon cables. We chose Omnivision 7620 image sensors for two reasons. Most importantly, the vertical and horizontal sync signals can be driven into the chip, allowing complete control over the video timing. Synchronizing cameras, traditionally a difficult feat with analog cameras, is simple with CMOS imagers. The Omnivision chip generates 640x480 pixel progressive scan video at 30fps and, conveniently, can be programmed to output YUV4:2:2 format video. This is the format required by our MPEG2 encoder.

The main boards have a Motorola Coldfire MCF5206e microprocesssor, a Texas Instruments IEEE1394 chipset, a Sony CXD1922q MPEG2 video encoder, and a Xilinx Spartan-XL Field Programmable Gate Array (FPGA). They also have 16MB of DRAM for storing the microprocessor's executable with its heap and stack, and raw image data from the sensors. Using its built-in I2C serial interface, the microprocessor can query and program control registers on the OV7620 image sensor. With the exception of a small boot EEPROM that initializes the microprocessor and the IEEE1394 interface, all executables, bitfiles and microcode are downloaded from the host PC over the 1394 bus, making code iterations quick and simple.

The FPGA handles the interfaces between the image sensor, the MPEG2 encoder, the microprocessor, and the IEEE1394 isochronous transfer mechanism. It generates the sync signals for the image sensors and can be synchronized by an IEEE1394 command or a trigger signal passed between the boards. Video input from the image sensor is synchronized to the system clock via an asynchronous FIFO. The FPGA can store sequential frames to the on-board DRAM in real time using DMA transfers. After a preset number of frames are stored to DRAM, they can be uploaded to the host PC by IEEE1394 asynchronous read requests. The microprocessor services these requests and transfers the data from DRAM to the 1394 chipset. Setting aside some memory for program code, heap and stack space, we can easily save 20 sequential YUV4:2:2 images of 40 raw Bayer mosaic images in the local DRAM.

To stream raw or MPEG2 compressed video, the FPGA routes the appropriate data through an 8KB FIFO to the isochronous data interface of the IEEE1394 chipset. Here, the microprocessor sets up the IEEE1394 interface but is not involved in transferring the data. Streaming continues until the board receives a IEEE1394 asynchronous command to stop. Currently, our prototype only uses the FPGA to provide the "glue logic" between the different chips in our design, but we are including an additional FPGA with dedicated memory in our final design in order to perform some basic image processing locally at each camera.

Although all the boards in the system are physically identical, one board in the system is treated differently and designated the"root" board. The root board generates the clock, reset and trigger signals which are sent to the other boards in the array. We guarantee frame accurate starts for the snapshots and video acquisition by sending a command from the host PC to the root board, which then waits until a specific portion of the frame to either assert a trigger signal or broadcast a 1394 command to tell the other boards to start the desired operation.

4.3. Host PC

The host PC in our system is a linux box with a PCI IEEE1394 interface card and two 18GB Ultra160 SCSI disk drives. Using software RAID 0 striping, we can write over 50MB/s of data at a sustained rate to the SCSI drives. Open source linux IEEE1394 code and drivers greatly accelerated our board bringup time. The standard linux IEEE1394 drivers only provide DMA assisted transfers from up to four isochronous channels at any given time, so we modified them slightly to work with up to all 64 channels as described in the OHCI1394 specification. This is critical for letting us stream MPEG2 data from all the cameras on a given bus at the same time. If we stream 40MB/s over the IEEE1394 bus to the disk array (over 5Mb/s for 60 cameras, or 10Mb/s for 30), we can store over 15 minutes of data.

As mentioned earlier, our final array of more than 100 cameras will need two or more host PC's. Each one downloads the executables for the boards and the FPGA bitfiles in parallel to all the camera nodes on its bus. Sending the Coldfire executable and the FPGA bitfile over IEEE1394 interface lets us iterate quickly when designing code and also lets us easily configure the cameras for different purposes. A user on the host PC can also view live raw video from any camera on that PC's 1394 bus and interactively change image sensor attributes like exposure time, color gains, and so on.

4.4. Physical Camera Configuration

Figure 2 shows a closeup view one of our small image sensor boards and the six camera setup we are using now. The lenses are fixed focal length and must be manually adjusted. We are using optical mounts and 80/20 bars for convenience. The mounts actually prevent us from packing the cameras as tightly as the board dimensions allow, but they are very easy to reconfigure for different spacings.

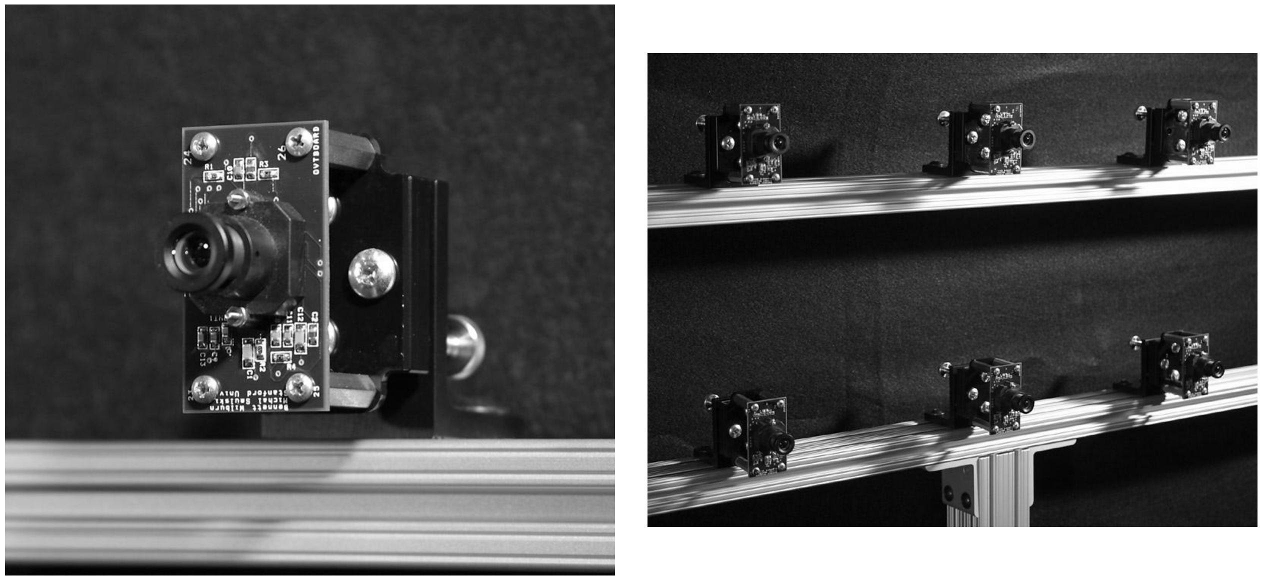

Figure 2: Individual sensor boards and prototype camera tile arrangement(without cables).

4.5. General Operation

Each camera node in the Light Field Video Camera array is a fully functional IEEE1394 device and can be queried from the host PC over the IEEE1394 bus. When the power is turned on or after a board reset, the Coldfire runs boot code that configures the processor memory address space (making the processor aware of the

DRAM, for example) and sets up the IEEE1394 interface. The host PC queries and identifies each camera and determines the bus topology. Because each device is assigned a physical ID that could change between resets, each board has a unique ID number stored in its EEPROM along with the boot code.

Once the PC has determined the IEEE1394 bus topology and identified the cameras, the user can download a more sophisticated executable to all of the camera nodes in parallel using the IEEE1394 broadcast address, physical ID 63. The executable runs a simple embedded operating system and has a full set of commands for taking snapshots, streaming live video and streaming MPEG2 compressed video. All commands are sent from the host PC to the boards as writes to certain addresses in the IEEE1394 address space. Furthermore, much of the state on the board is also mapped into this address space and can be read and written from the host. This includes the DRAM memory, control registers in the FPGA, the IEEE1394 Link Layer Controller and the MPEG2 encoder chip, and the state in the image sensor I2C registers. Bitfiles for the FPGA and microcode for the MPEG2 encoder are downloaded from the host and programmed into their respective devices in parallel.

After configuring the FPGA's, which drive the timing for the CMOS image sensors, the user can set parameters for the sensors. Currently, we turn off gamma correction in order to get a linear response from the cameras. We let the on-chip autoexposure algorithm pick a suitable starting exposure time, then turn it off to freeze that value. Finally, we disable the white balancing and set the color gains directly.

5. PROTOTYPE RESULTS

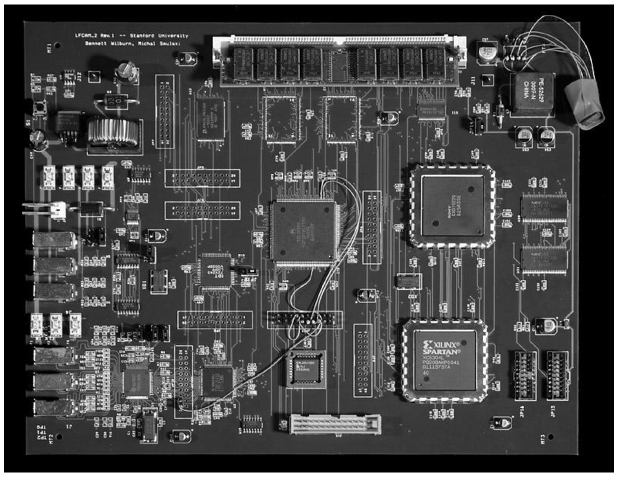
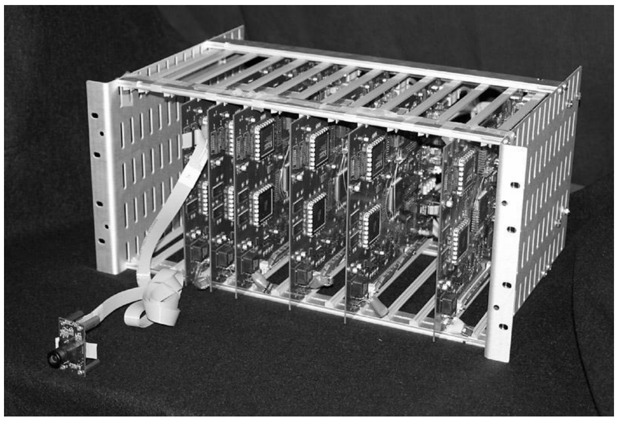

Figure 3: Light Field Video Camera main board and cage with one camera attached.

Figure 3 shows one of our prototype main boards and the cage of boards with one camera attached. The full prototype has six camera nodes connected to one host PC and is fully functional. We can display raw data from any one camera at a time on the host PC and store up to 20 synchronized sequential snapshots from the cameras to their local memory. Finally, we can stream MPEG2 compressed video simultaneously from all six cameras to the SCSI disk array. Like the snapshots, the video data are synchronized and all start on the same frame.

Figure 4 shows six images from video captured with the prototype array. The images are all from the same frame number (in this case, number 105) of the MPEG2 video recorded from their respective cameras. The pictures were recorded from the three by two camera setup shown in Figure 2 and are arranged in order so that the top left image corresponds to the top left camera from the perspective of a camera looking into the scene. From these images it appears that the video is synchronized and frame accurate. In fact, we have verified with a logic analyzer that the video synchronization signals for the cameras are coincident to within 100μsec and the data acquisition starts on the same frame for all cameras. The 100μsec uncertainty occurs because we are currently using a broadcast IEEE1394 command to synchronize the cameras, and the microprocessors

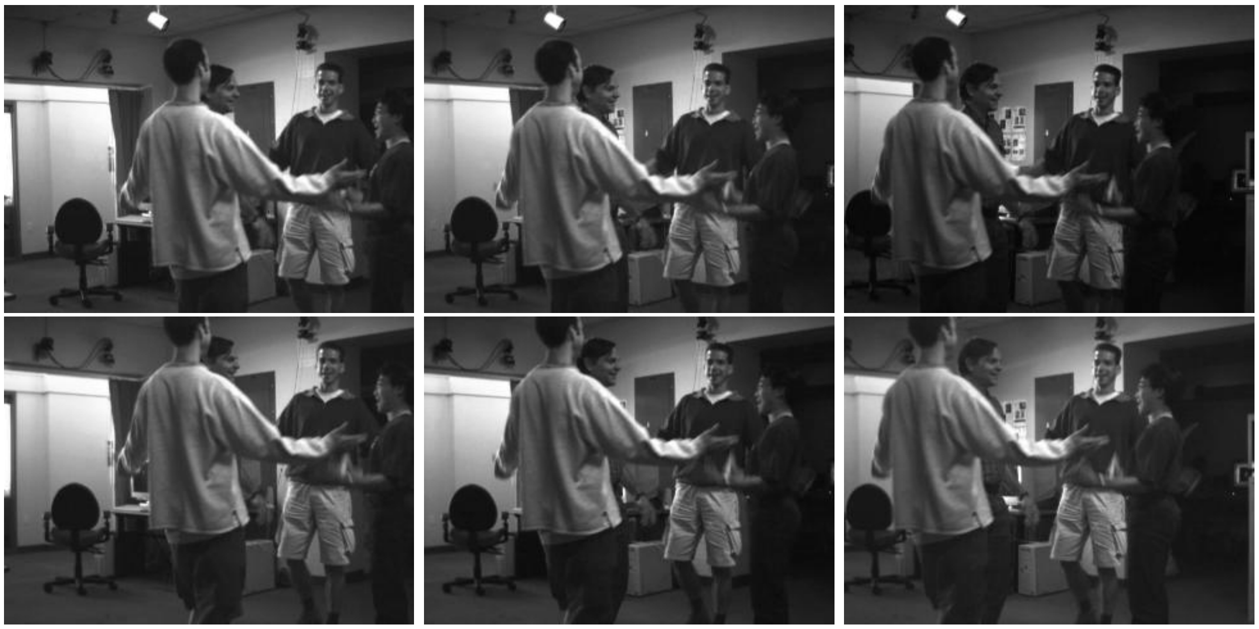

Figure 4: MPEG2 encoded image of same the frame from six cameras.

take a variable amount of time to respond to interrupt requests from the IEEE1394 chipset. Using the trigger signals will reduce this uncertainty by at least an order of magnitude because we will bypass the microprocessors entirely.

In addition to demonstrating the functionality of the design, the prototype also shows its scalability. New cameras are added to the system by simply connecting them to the IEEE1394 bus. Regardless of the number of cameras in the system, they can all be booted, programmed and configured in the same amount of time. More importantly, the array is already operating within the bandwidth constraints of the final system. Snapshots of raw data to local memory and streaming raw video from any one camera does not get any more difficult as more boards are added to the system. Streaming MPEG2 compressed data does, though, because it uses more and more IEEE1394 bandwidth. Currently, each of our camera nodes sends 30 bytes of compressed video each isochronous cycle. At this rate, we could put 45 cameras on the IEEE1394 bus. The other potential bottleneck is streaming over the PCI bus to memory and back out to the disk array. We clearly have not streamed data from tens of cameras to disk yet, but we have recorded 18MB/s of raw video from one camera to the striped disks in real time. The total default isochronous bus bandwidth is only a little more than twice that much.

System debug for this design was relatively painless because we included many test headers and because of the support available in the open source community for the Coldfire Background Debug Mode (BDM) port. Connecting the host PC to it through the serial port, we are able to single step through assembly or source level code running on the board. From the host PC, we can simultaneously debug both the host PC code and the embedded processor code. Furthermore, we connected headers to some FPGA pins to route internal signals out to a logic analyzer.

6. FUTURE WORK

The first piece of future work is to construct the final 128 node Light Field Video Camera. We are currently revising the board design to fix some minor design flaws and add another FPGA with memory that will be dedicated for basic image processing operations. At the time of the writing of this paper, we are working on the layout for the new design and anticipate sending the finished design out and starting fabrication of a 128 camera array in a couple weeks. At the same time, we are working on developing applications for the array, such as disparity-based methods for view interpolation. Preliminary results indicate that disparity estimation will work well on the MPEG compressed data. We have basic geometric calibration routines finished, but still need to investigate the best way to calibrate, geometrically and radiometrically, one hundred cameras. The individual image sensors show much variation in their color responses, even to the naked eye. Furthermore, much work will be required to parallelize algorithms we develop so they will run in reasonable amounts of time on data from so many cameras.

ACKNOWLEDGMENTS

The authors would like to thanks various members of the IMTV project at Stanford for their help and inspiration, especially Marcus Magnor, Vaibhav Vaish and professors Marc Levoy and Pat Hanrahan. We are also grateful to the online Coldfire Development Resources and IEEE1394 for Linux communities for tools, code and tips that greatly aided this project.

REFERENCES

1. M. Levoy and P. Hanrahan, "Light field rendering," *Proc. ACM Conference on Computer Graphics (SIGGRAPH'96),* New Orleans, USA , pp. 31–42, Aug. 1996.
2. S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The Lumigraph," *Proc. ACM Conference on Computer Graphics (SIGGRAPH'96),* New Orleans, USA , pp. 43–54, Aug. 1996.
3. J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum, "Plenoptic sampling," *Proc. ACM Conference on Computer Graphics (SIGGRAPH'00),* New Orleans, USA , pp. 307–318, Aug. 2000.
4. P. Debevec, C. Taylor, and J. Malik, "Modelling and rendering architecture from photographs," *Proc. ACM Conference on Computer Graphics (SIGGRAPH'96),* New Orleans, USA , pp. 11–20, Aug. 1996.
5. T. Kanade, H. Saito, and S. Vedula, "The 3d-room: Digitizing time-varying 3d events by synchronized multiple video streams," Tech. Rep. CMU-RI-TR-98-34, Carnegie Mellon University, 1998.
6. W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan, "Image-based visual hulls," *Proc. ACM Conference on Computer Graphics (SIGGRAPH-2000),* New Orleans, USA , pp. 369–374, July 2000.