A Scalable Video Rate Camera Interface

Jon A. Webb[1,2], Thomas Warfel[3], and Sing Bing Kang[2]

26 September 1994
CMU-CS-94-192

[1]Computer Science Department
[2]Robotics Intsitute
School of Computer Science

[3]Electrical and Computer Engineering Department

Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

Abstract

We survey the state of the art in high-speed interfaces for video input to high performance computers and note the difficulty of providing video at rates appropriate to modern parallel computers. Most interfaces that have been developed to date are not scalable, required extensive hardware development, and impose a full frame time delay between the moment the camera captures video and the moment it is available for processing. We propose a solution, based on a simple interface we have developed, which has been integrated into the iWarp parallel computer developed by Carnegie Mellon University and Intel Corporation. The interface takes advantage of iWarp's systolic capabilities, does not impose any frame buffer delay time, was simple to design, and is readily scalable to provide up to 32 camera ports, from all of which data can be captured at full video rate, on a system that fits in a 19" 6U rack. We have applied the system to multibaseline stereo vision, and provide performance figures.

This research was partially supported by the Adanced Research Projects Agency of the Department of Defense under contract number F19628-93-C-0171, ARPA order number A655, "High Performance Computing Graphics," monitored by Hanscom Air Force Base. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Advanced Research Projects Agency, the Department of Defense, or the U.S. government.

ACM Computing Reviews Keywords: C.1.2 Multiple-instruction-stream, multiple-data stream processors, parallel processors. C.3 Real-time systems. I.2.10 Vision and Scene Understanding. I.2.9 Robotics. I.3.1 Input devices. I.4.1 Digitization. I.4.2 Compression.

1. Introduction

Parallel computers now have the raw processing power to perform interesting computer vision calculations on full-sized images at video rate. However, interfacing cameras to these machines in a way that allows the full exploitation of their capabilities is difficult. As a result, these algorithms are usually applied to pre-stored images, and high performance is rarely observed in the real world.

We survey state of the art video interfaces, and present a new approach which we have successfully implemented. We have demonstrated continuous video rate capture for full-size (480×512) images from four cameras. Our approach is scalable; using it, we are able to capture video rate data from as many as thirty-two cameras in a system that fits in a standard 19" 6U rack. A chief application of this system is in multibaseline stereo vision; we shall describe this and provide early results from our implementation.

2. Problems with video data

Video data is difficult to deal with for several reasons:

- It requires high bandwidth (a conventional black and white television camera produces 7.5 MB/s of useful data.)
- It is uninterruptable (the scanning of the data is controlled by the camera, not the computer, so the data must be captured as soon as it arrives.)
- It is large (an ordinary television camera produces a 1/4 MB image).
- It is readily scalable (the introduction of color increases the volume by a factor of three; stereo introduces an independent source of extra channels.)

For these reasons conventional designs have dealt with video by introducing a frame buffer (FB) between the computer and the camera, as illustrated in Figure 1.

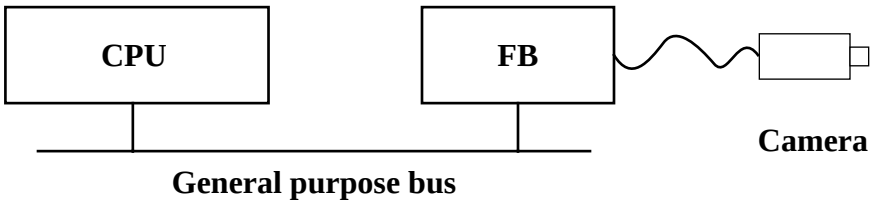


Figure 1. Conventional camera interface

This design is well-established, widely available commercially, and works for general-purpose serial computers. However, it is not suitable for high-performance parallel computers. First, it imposes a full frame time (33 ms) delay between the time of the capture of the video and the time it can be processed. This delay is unacceptable in low-latency applications like robot control. Second, it does not scale; while a general purpose bus these days may be able to sustain video rate data from a single camera or even a color camera, multiple color cameras quickly swamp a common general-purpose bus.

Thus we see that we cannot simply use serial camera interface designs for parallel computers. We now turn to designs for parallel computer camera interfaces.

3. Parallel Computer Camera Interfaces

We consider three approaches to interfacing cameras to parallel computers:

- Workstation network
- Hardware distribution
- Software distribution

3.1 Workstation network

In workstation networks we simply generalize the design of Figure 1 to multiple host computers, as shown in Figure 2. Here a network interface (NI) allows all the workstations to act in concert and exchange video data they capture in parallel.

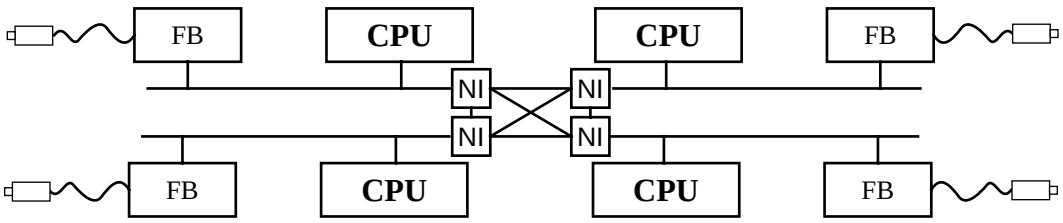


Figure 2. Workstation network

This design is scalable and uses commercial components. However

- It is costly (high speed networks that allow the exchange of data at rates sufficient to support video processing are only now becoming available, and are still expensive),
- It does not address the frame buffer delay problem, and
- It can be surprisingly difficult to synchronize the capture of images from workstations interconnected in such a network. A conventional operating system such as Unix can create large, unpredictable delays in user-level process response to events, making it difficult to ensure that all workstations get data from the same frame, even if the cameras themselves are synchronized through an external genlock signal. (The use of conventional operating systems like Unix is important in this context to achieve the cost savings in development time offered by this design.)

3.2 Hardware distribution

By hardware distribution, we mean hardware systems that directly distribute the video data to a large number of processors. This method has been used in several high performance computing systems. Most notably, the Goodyear MPP included a video interface that allowed images to be fed in a row at a time in parallel to all processors [Potter 1985].

These interfaces solve the problem of providing data at video rate, and can be designed in such a way as to overcome the frame buffer delay problem (as in the BVV series of machines [Graefe 1990].) However, they do not scale with the processor array, since they use an independent data distribution mechanism. Moreover, an independent distribution mechanism just for video data implies some waste of development time, since two similar data distribution networks must be created.

A somewhat more general approach was taken by MasPar. They developed a HiPPI interface that allows data to be fed at high speed directly to the MasPar switch. Video data can be supplied from a HiPPI framebuffer

(such as the PsiTech HFB24-110.) This approach makes use of reusable commercial components. However, it is not scalable in number of cameras. This is because each camera would require its own interface, and a limited number of HiPPI interfaces can be attached. In addition, the frame buffers available are not suitable for many computer vision applications. For example, the PsiTech frame buffer imposes the frame buffer delay discussed earlier, it is expensive, and it is primarily designed for high-resolution display, not video capture from standard cameras. This adds to its cost but not its utility in many computer vision applications.

3.3 Software distribution

A third approach has been taken in MIMD parallel computers, such as the Meiko (in the Mk027 frame buffer [Goulding 1988], among others.) Here one or a small number of the nodes are interfaced directly to the frame-buffer. The interprocessor communications network is used to distribute the captured images under software control of the nodes.

This approach is similar to the workstation network except that a parallel computer network is used to distribute the data. Since parallel computer nodes are designed to be interconnected, the per-node cost of the high speed interface is lower. Moreover, parallel computer nodes are more easily synchronized than workstations because the operating system imposes less latency between the communications network and the program. It is also similar to the specialized hardware approach except that the image is provided to only one or a few nodes. This eliminates the need to develop a full-scale hardware distribution mechanism and lessens the commitment to non-commercial products. It also enhances scalability since interfaces can be attached at multiple points in the array to support multiple cameras.

This design overcomes many of the problems with workstation networks and hardware distribution, and forms the basis of our approach. In order to get best performance we must overcome the frame buffer delay problem and use bandwidth most efficiently.

First, we would like to overcome the frame buffer delay problem. In order to do this, the data must be distributed from the video interface as it arrives rather than first being stored in a frame buffer, whether in the processor's memory or in an auxiliary board.

If the data is to be taken by the processors directly from the video interface, bandwidth must be available continuously throughout the video capture time, since the video stream cannot be interrupted, though the use of a line buffer memory can relax this constraint (since no pixel data is sent from the camera during the 12 μs horizontal retrace, allowing time to "catch up" transferring data among processors.) Additionally, interprocessor bandwidth must be great enough to support the distribution of video data.

A second issue motivates the use of a systolic approach. Consider a design like that shown in Figure 3. In a conventional message-passing computer the processor (P) would read data from the video interface (VI) and store it into the local memory (M.) It would then form a message and send the data to other processors, through the network interface (NI).

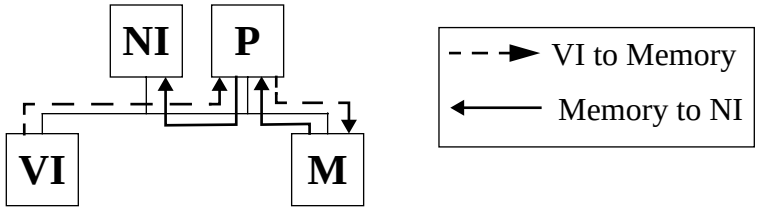


Figure 3. Data flow in a message passing system.

With this design the data travels twice (at least) across the memory bus, doubling the required bandwidth. For

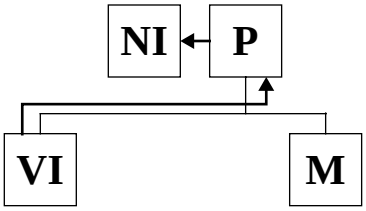


Figure 4. Data flow in a systolic system.

video rate data this is significant. In a systolic design the data would be read by the processor from the video interface and then written directly to the network interface, as shown in Figure 4.

We now propose a modification of the software distribution approach that takes advantage of the insights above.

4. Our Approach

In order to overcome the frame buffer delay problem, we distribute the frame grabbing and frame buffering operations across a tightly-coupled systolic processor array, using a simple A/D interface to connect to the cameras. We now describe the hardware design of the video interface.

The main objective was to acquire and distribute 8-bit 480×512 images generated by an NTSC interlaced black and white video source. Secondary objectives included less than one pixel horizontal jitter, multiple video sources, and minimal hardware. Our intent was to keep the video interface hardware simple and rely on the iWarp processor as much as possible.

4.1 Sampling

A typical line of NTSC video is roughly 63.5 microseconds long [Jack, 1993]; when the sync and blanking interval is removed, though, the useful video portion is only about 51.2 microseconds long. Thus, a 10MHz sampling rate suffices to acquire an image at 512 pixels per line. As iWarp provides direct access to the processor's local bus, a memory-mapped ADC allows 10 MHz sampling purely under software control.

While sampling at 10 MHz is not a problem, since the iWarp clock rate is 20 MHz, knowing *when* to begin sampling is more difficult. Sampling must commence less than one pixel-time after the horizontal sync, otherwise, the acquired image will have a noticeable jitter from line-to-line. Sampling 512 pixels per line implies a 100 nanosecond window within which sampling must start. Since our machine has a 50 ns clock, we could not use software alone to detect the edge of the horizontal sync since a basic read-compare-branch loop would take at least 5 clock-periods, resulting in a 2-pixel jitter.

Fortunately, the iWarp supports two different memory-access modes, one for fast memory (which guarantees response so no wait states are required) and one for slow memory (which allows an arbitrary delay.) By connecting our ADC to the iWarp memory interface as if it were slow memory we were able to wire the “memory ready” interface line to a sync-detect chip. When the chip detected the edge of the horizontal sync, it would enable (and keep enabled) the memory ready line on the iWarp. Thus, the first read of a video line stalls until the edge of the horizontal sync; thereafter it samples the line at a 10 MHz rate under software control.

4.2 Multiple sources

The iWarp has a wide data path on the external memory interface that can be treated as 64, 32, 16, or 8 bits wide. Since we are acquiring 8-bit video images, we adopted the 32-bit wide interface (allowing four channels per board) as a compromise between number of sources per board vs. board and distribution complexity.

Since most modern CCD cameras have built-in genlock support, we added the restriction that multiple video sources going to the same acquisition board would be genlocked to one another. This simplified the acquisition problem since only one signal need be watched when acquiring/distributing data; all the other cameras will have the same timing.

4.3 Distribution to other processors

iWarp's systolic design [Borkar, Cohn et al. 1990] allows network connections to be mapped to special on-chip registers called "gates." Data can be read from memory and "stored" to a gate just as any other register. One "read" from the ADC returns the four video signals as one 32-bit word; each byte represents the grey level of a single pixel, one from each video source. If data from all four sources goes to the same place, distribution is fairly simple. The cell that acquires the data simply sends it along a network connection that guarantees the necessary bandwidth and latency. Cells downstream pick off their data as it arrives, and allow the remaining data to continue flowing downstream.

This feature allows low-latency acquisition; a cell has the video data within a few microseconds (rather than milliseconds, if a frame buffer was used) of the time it arrived at the acquisition board. If data is being distributed over N cells, each cell has about 16K * (N-1)/N μs to process its data before the next frame arrives.

The major disadvantage is that since NTSC video is interlaced, the individual processors are responsible for re-assembling the even/odd fields into the composite images. It is possible to set aside several processors to buffer the incoming data before redistributing the assembled images, but this would re-introduce a one or two frame latency.

4.4 Circuit implementation

The circuit, shown in Figure 5, was designed for minimal variety of parts, simple power requirements, and simple implementation. It is identical for the four video channels.

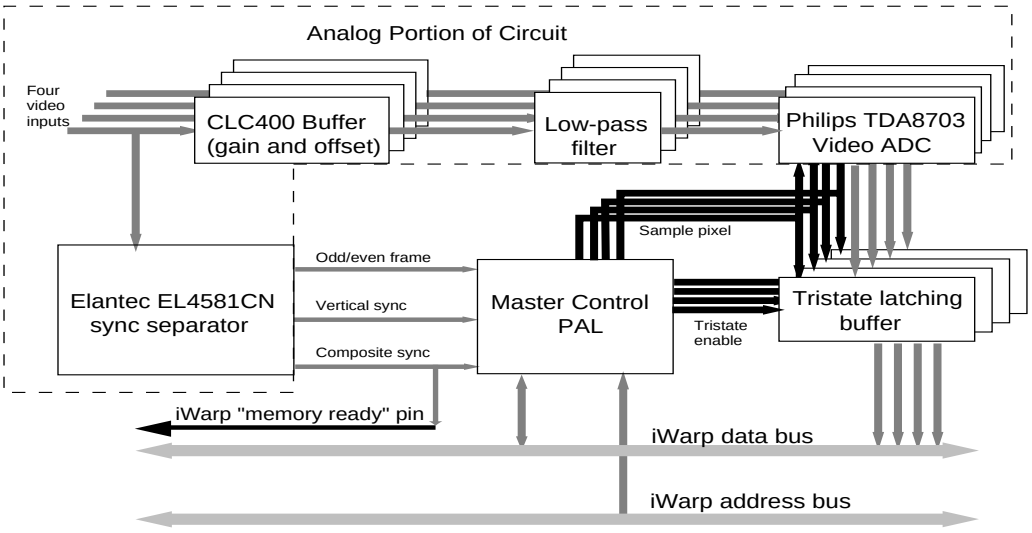


Figure 5. Video channel circuit.

Video input is DC-coupled, buffered, and level-shifted through a ComLinear CLC400 op-amp. This is a relatively low-cost amplifier that has sufficient bandwidth for 20 MHz signals and uses just +/-5 V. Since our video sources have a fairly constant output, small, manual potentiometers were used to set the buffer gain and offset for each channel. The output of the amplifier is passed through a low-pass filter with a 5 MHz cut-off frequency.

Filter output goes to a Philips TDA8703 8-bit ADC, which digitizes the signal and presents the output to a PAL which simply acts as a fast, tri-stated latch. Both the Philips ADC and the latch-PAL receive a 10 MHz clock signal from a master control PAL.

The master control PAL provides address decode for the video board and generates a clock from the read request signal. It also provides read access to an Elantec EL4581CN Sync-separator chip. This chip is connected to one of the incoming video signals and provides vertical sync, composite sync, and even/odd video frame information. The composite sync bit is also connected directly to the iWarp external memory interface "memory ready" line. To keep the implementation simple, no FIFOs or PLL-derived clock signals were used. All timing is under pure software control. This limits the sampling rate to processor clock multiples, i.e., 1024, 512, 341, or 256 pixels per line.

Pixel acquisition is a one-pixel deep pipeline, as shown in Figure 6. A "sample control line" is derived from the AND'ing of the address decode logic with the iWarp "memory read" signal. When the iWarp processor does a "read" from the memory address to which the ADC is mapped, the "sample control line" goes high. At the end of that cycle the control line goes low again until the next "read" from that address. This "sample control line" is used as the clock signal for the ADC's themselves. It also controls the latching and tri-stating of the buffer PAL's.

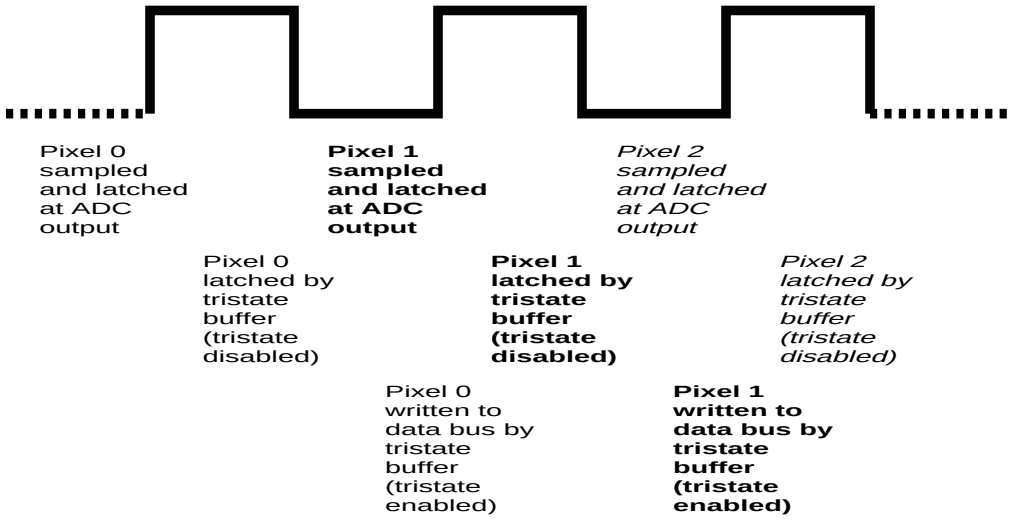


Figure 6. Pixel acquisition pipeline.

4.5 Storage

We have configured an 8×8 iWarp array with the video interface described above and 16 MB DRAM memories on the first 32 cells (for a total of 512 MB, plus 512 KB/cell for operating system, program, and auxiliary variables), as shown in Figure 7. (Each iWarp cell is connected to its four neighbors, and the edges are connected as a torus. These connections are not shown.) In order to store images, we must route data from the video input cell to the large memory cells in a way that ensures reliable delivery of the 40 MB/s peak bandwidth needed.

Since each iWarp physical link promises a peak bandwidth of 40 MB/s, this would seem to be trivial, but in fact a flaw in the iWarp design limits the peak bandwidth of a physical link to slightly under 40 MB/s. The flaw momentarily interrupts the video stream and results in unacceptable jitter in the image. As a result we adopt the more complex routing strategy shown in Figure 7. The video input is divided into two streams (alternate pixels within a row follow different routes) and each half-field is stored in a different bank of cells. The half-fields are stored in successive memory locations within each cell's memory, with each cell storing as many half-fields as

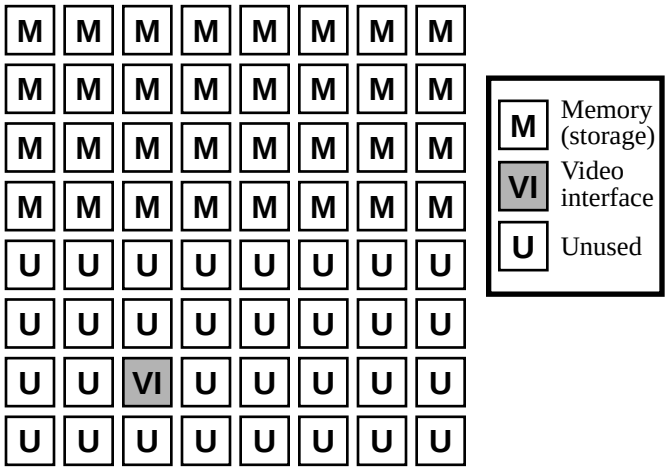


Figure 7. iWarp array with memories and video input.

it can before passing the rest of the fields onto later cells in its bank (for more information on how this is done see [Webb 1993].) With this strategy, and using the full 512 MB of memory available in the large memory cells, we can store over 500 480×512 images at full video rate. In our current system, these images are copied onto disk before further processing.

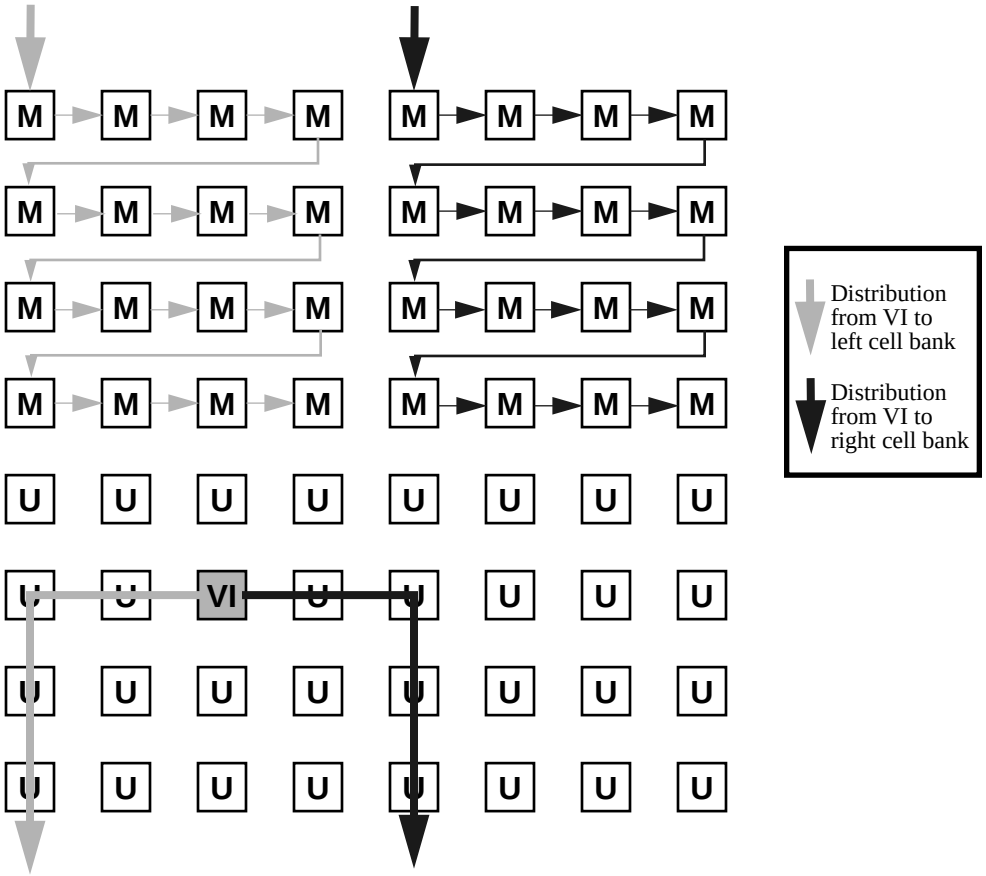


Figure 8. Routing data from the video interface to the memories (taking advantage of the toroidal configuration of iWarp.)

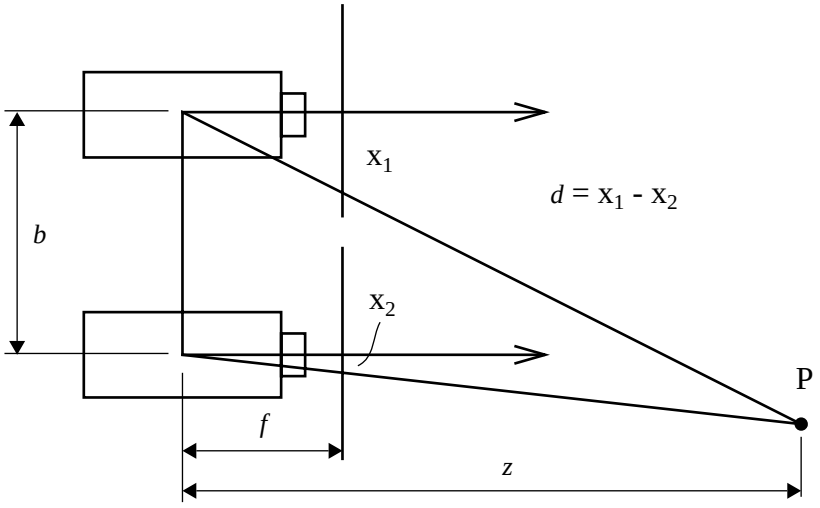
We now turn to the application which has driven much of this system development, multibaseline stereo vision.We trade off multiple cameras (and correspondingly higher bandwidth) for higher accuracy, which is a suitable way to take advantage of high performance computers.

5. Application to stereo vision

Computer vision deals with the extraction and analysis of 3D scenes. Binocular stereo vision is a relatively inexpensive method by which 3D information of the scene can be extracted from triangulation using two cameras. Its primary drawbacks of the problems of image point correspondence (for a survey of correspondence techniques, see [Dhond and Aggarwal 1989]) and baseline[1]/ease of correspondence trade-off have been mitigated using multiple cameras or camera locations; such an approach has been termed multibaseline stereo. Stereo vision is computationally intensive, but, fortunately, the spatially repetitive nature of depth recovery lends itself to parallelization. This is especially critical in the case of multibaseline stereo with high image resolution and the practical requirement of timely extraction of data.

5.1 The principle of multibaseline stereo

In binocular stereo where the two cameras are arranged in parallel, depth can be easily calculated given the disparity[2]. If the focal length of both cameras is *f*, the baseline *b* and disparity *d*, then the depth *z* is given by $z = f*b/d$, as shown in Figure 9.



From similar triangles,

$$\frac{d}{b} = \frac{f}{z}$$

Figure 9. Relationship between the baseline *b*, disparity *d*, focal length *f*, and depth *z*

In multibaseline stereo, more than two cameras or camera locations are employed, yielding multiple images with different baselines [Okutomi and Kanade 1993]. In the parallel configuration, each camera is a lateral displacement of the other. From Figure 9, $d = f*b/z$ (we assume for illustration that the cameras have identical focal lengths).

---

1 The baseline is the distance between two camera optical centers.

2 The disparity is defined as the shift in corresponding points in the left and right images.

For a given depth, we then calculate the respective expected disparities relative to a reference camera (say, the left-most camera) as well as the sum of match errors over all the cameras. (An example of a match error is the image difference of image patches centered at corresponding points.) By iterating the calculations over a given resolution and interval of depths, the depth associated with a given pixel in the reference camera is taken to be the one with the lowest amount of error.

5.2 Multibaseline stereo in a convergent configuration

A problem associated with a stereo arrangement of parallel camera locations is the limited overlap between the fields of views of all the cameras. The percentage of overlap increases with depth. The primary advantage is the simple and direct formula in extracting depth.

Verging the cameras at a specific volume in space is optimal in an indoor application where maximum utility of the camera visual range is desired and the workspace size is constrained and known *a priori*. Such a configuration is illustrated in Figure 10. One such application is the tracking of objects in the Assembly Plan from Observation project [Ikeuchi and Suehiro 1992]. The aim of the project is to enable a robot system observe a human perform a task, understand the task, and replicate that task using a robotic manipulator. By continuously monitoring the human hand motion, motion breakpoints such as the point of grasping and ungrasping an object can be extracted [Kang and Ikeuchi 1994]. The verged multibaseline camera system can extend the capability of the system to tracking the object being manipulated by the human. For this purpose, we require fast image acquisition and depth recovery.

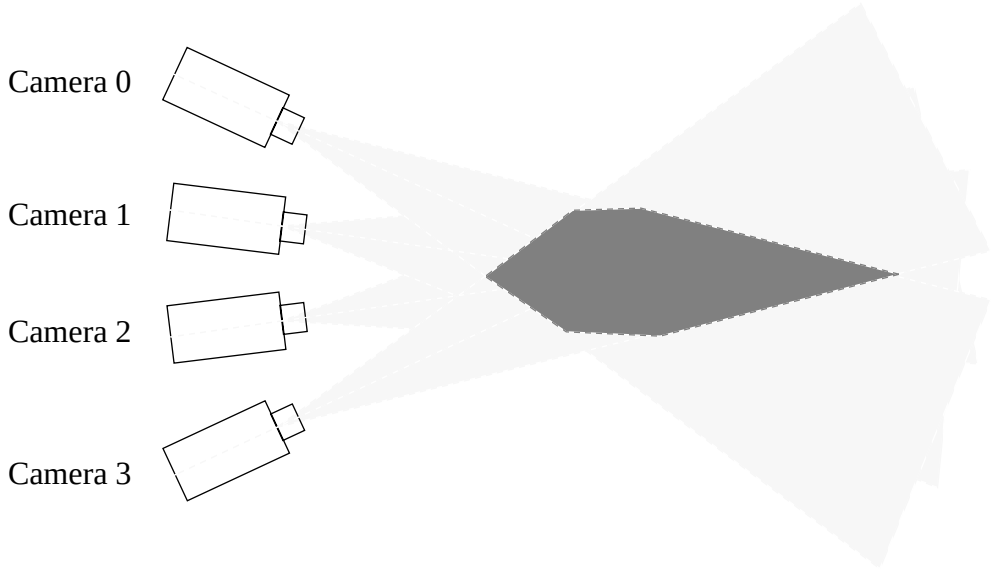


Figure 10. A verged camera configuration (dark shaded area is the common 3D space viewable from all cameras).

The disadvantage of using a convergent camera configuration is that the epipolar lines[1] at each camera image is no longer parallel to their scan lines, leading to more complicated mathematics. However, one can easily perform a warping operation (called *rectification*) on the camera images as a preprocessing step to depth recovery. The process of rectification for a pair of images transforms the original pair of image planes to another pair such that the resulting epipolar lines are parallel and equal along the new scan lines. The concept of rectification is depicted in Figure 11. Here $\mathbf{c}_1$ and $\mathbf{c}_2$ are the camera optical centers, Π_1 and Π_2 the original image

1 The epipolar lines on a pair of cameras for any given 3D point in space are the intersection of the plane passing through that point and the two camera optical centers, and the image planes. For cameras aligned in parallel, these epipolar lines are parallel and correspond to the scan lines.

planes, and Ω_1 and Ω_2 the rectified image planes. The condition of parallel and equal epipolar lines necessitates planes Ω_1 and Ω_2 to lie in the same plane, indicated as Ω_{12}. A point **q** is projected to image points $\mathbf{v}_1$ and $\mathbf{v}_2$ on the same scan line in the rectified planes.

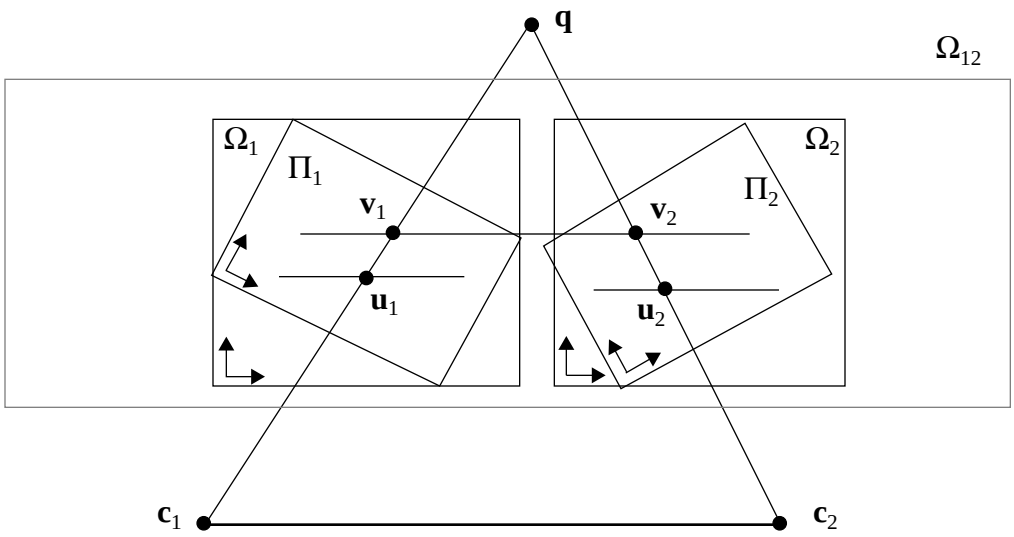


Figure 11. Image rectification

5.3 The 4-camera multibaseline system in a convergent configuration

The multibaseline system that we have built is shown in Figure 12. It comprises four cameras mounted on a metal bar, which in turn is mounted on a sturdy tripod stand; each camera can be rotated about a vertical axis and fixed at discrete positions along the bar. The four camera video signals are all synchronized by feeding the genlock output from one camera to the genlock inputs of the others.

5.3.1 Camera calibration

Camera calibration refers to the determination of the extrinsic (relative pose) and intrinsic (optical center image offset, focal length and aspect ratio) camera parameters. The pinhole camera model is assumed in the calibration process. The origin of the verged camera configuration coincides with that of the left-most (reference) camera.

A planar dot pattern arranged in a 7×7 equally spaced grid is used in calibrating the cameras; images of this pattern are taken at various depth positions (five different depths in our case). The dots of the calibration pattern are detected using a star-shaped template with the weight distribution decreasing towards the center. The entire pattern is extracted and tracked from one camera to the next by imposing structural constraints of each dot relative to its neighbors, namely by determining the nearest and second nearest distances to another dot. The simultaneous recovery of the camera parameters of all the four cameras can be done using the non-linear least-squares technique described by Szeliski and Kang [Szeliski and Kang 1994]. The inputs and outputs to this module are shown in the simplified diagram in Figure 13.

Figure 12. 4-camera multibaseline system

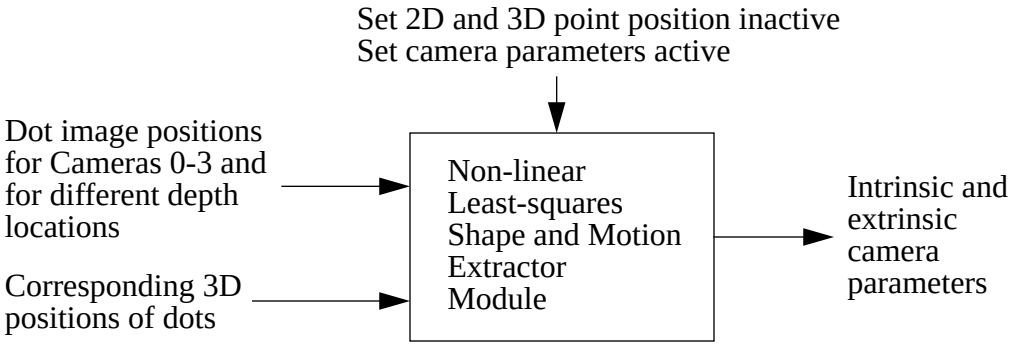
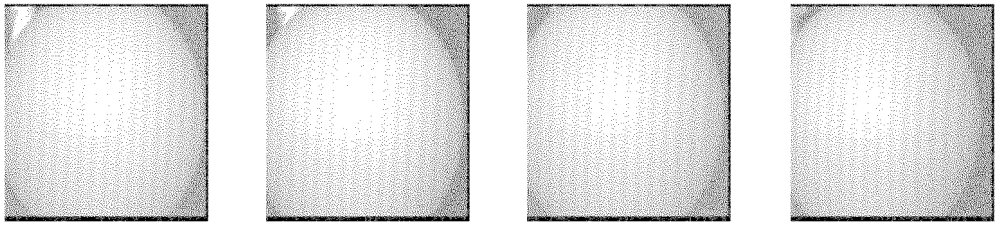


Figure 13. Non-linear least-squares approach to extraction of camera parameters

5.4 Results

In this section, we show the results of a set of images of a scene of a globe. In this scene, a pattern of sinusoidally varying intensity is projected onto the scene to facilitate image point correspondence. The four views are shown in Figure 14.

The recovered depth map is shown in Figure 15. The large peaks at the borders are outliers due to mismatches in the background.



(a) (b) (c) (d)

Figure 14. Views of the globe from the four cameras ((a)-(d))

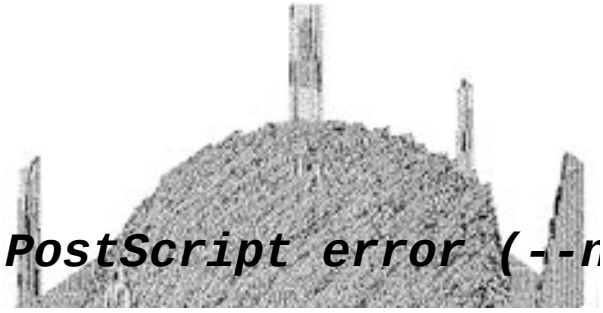


Figure 15. Recovered depth map of the scene

6. Future work

We are currently measuring and improving the accuracy of our four-camera multibaseline system.

We plan to extend the current system to allow the storage of data from as many as thirty-two cameras for more advanced stereo vision applications, using the system layout and routing shown in Figure 16. (The odd routing of data shown here is due to the necessity of achieving the required 40 MB/s output from each VI cell, and a restriction on the placement of VI boards due to the arrangement of the connectors on it and the iWarp cells.)

With this design we can store only about 2 s of video at video rate (since thirty-two cameras provide an average data rate of 240 MB/s, and our memory is still limited to 512 MB), severely limiting the utility of the system. We are therefore considering remote storage of the video data using an iWarp/HiPPI interface built as part of the Nectar project by Network Systems Corporation. With this interface we should be able to achieve 70-80 MB/s, which means that with a simple compression technique (perhaps DPCM) implemented on the unused cells we may be able to store data from all thirty-two cameras continuously, up to the limits of our storage server.

Acknowledgments

We thank Luke Tuttle for constructing the first prototypes of the video interface, Bill Ross for assisting in the design of the camera setup and jig, and Mark Wheeler for doing the analysis that led to the decision to verge