PUBLICATIONS OF THE ASTRONOMICAL SOCIETY OF THE PACIFIC, **110**:1067–1080, 1998 September

Data Acquisition for a 16 CCD Drift-Scan Survey

C. N. SABBEY, P. COPPI, AND A. OEMLER[1]

Department of Astronomy, Yale University, New Haven, CT 06520-8101;
sabbey@astro.yale.edu, coppi@astro.yale.edu, oemler@ociw.edu

Received 1998 May 15; accepted 1998 June 15

ABSTRACT. The QUEST (QUasar Equatorial Survey Team) collaboration is using an innovative 16 CCD mosaic camera at the prime focus of the 1 m Venezuelan Schmidt telescope to conduct a drift-scan survey of the equatorial sky (≈4000 deg^2 with $|b| > 25$). The data products, which consist of *UBV* photometry to $m_B \approx 21$ and objective prism spectroscopy to $m_B \approx 19$, will have various applications in addition to producing a homogenous sample of $\sim 10^4$ quasars. In this paper we describe the online data system, which is responsible for controlling the camera, and acquiring and managing ≈30 GB of raw image data per night. The data system, designed to be highly scalable and cost effective (under $10,000), distributes the data I/O and processing in parallel across a cluster of commodity PC components. We currently use seven Pentium-class processors (75–133 MHz), interconnected with parallel Ethernet networks, running the QNX real-time, network-wide operating system. To ease the data handling, fast algorithms provide various levels of online analysis and data compression, including a new lossless compression scheme based on adaptive linear prediction.

1. INTRODUCTION

The development of large charge-coupled device (CCD) mosaics has enabled next-generation digital sky surveys similar in scope to the venerable Palomar Survey but with the many benefits of using CCDs (Kron 1995 reviews several projects underway). In particular, the increase in detector quantum efficiency (DQE) of CCDs over photographic plates translates into a proportional improvement in the *survey efficiency* η (inversely related to the time to completion), given by

$$\eta = \Omega D^2 q \epsilon,$$

where Ω is the solid angle of sky on the detectors, D is the telescope diameter, q is the DQE, and ϵ is the observing efficiency. With this in mind, the QUEST (QUasar Equatorial Survey Team) collaboration, with members from Yale University, the Centro de Investigaciones de Astronomía (CIDA), the Universidad de Los Andes (ULA), and Indiana University, was formed to design and commission a mosaic camera to "tile" the focal plane of the 1 m Venezuelan Schmidt telescope for use as a dedicated survey instrument.

The 4 × 4 CCD array camera is shown in Figure 1 and discussed in Snyder (1998). The 2048^2 format Loral CCDs have a readout noise of ≈10 e^-, a full well capacity of ≈100,000 e^-, and typically ≈10 bad columns per chip. The 15 μm pixels set a scale of 1.″0 pixel^{-1}, such that each CCD subtends 0.°6 square on the sky, and the array of CCDs images a total of 2.°4 × 2.°4 of the Schmidt's 5° field of view. The CCDs are front-illuminated (thick) devices, but coated with a wavelength-shifting compound to provide ≈10% DQE in the UV. Mounted above each column of CCDs is a standard Johnson filter (see Fig. 1) or, in objective prism mode, a hot mirror with a 7000 Å cutoff to limit the sky background. A custom-designed, distortion-free field flattener serves as a front window for the dewar, resulting in high image quality over the entire array.

The camera is operated in drift-scan mode (also known as time-delay integration or TDI scanning; McGraw, Cawson, & Keane 1986), a very efficient ($\epsilon \approx 1$) and stable technique for conducting large area surveys to moderate depths. The telescope is parked and the CCDs, oriented so that the clocking direction is aligned with the east-west motion of the sky, are read out continuously at the apparent sidereal rate. Each CCD thereby images a strip of sky as it passes over the telescope's field of view, with the array covering a total of ≈2.°4 × 15° hr^{-1} × cos δ through four filters. The effective exposure time, set by the CCD crossing time, is ≈140 s on the equator [(2048 pixels × 1″ pixel^{-1})/15″ s^{-1}]. Because the telescope is used as a transit instrument, and each point on the sky is sampled with the mean sensitivity of the 2048 pixels in a CCD column, very uniform astrometry and photometry are possible. In addition, drift scanning can effectively be done with CCDs that have a variety of blemishes, making the cost of this project feasible.

The two primary obstacles encountered in drift scanning are the curved stellar paths across the focal plane and the differential drift rate. The radius of curvature of the stellar path is

[1] Also Carnegie Observatories, 813 Santa Barbara Street, Pasadena, CA 91101.

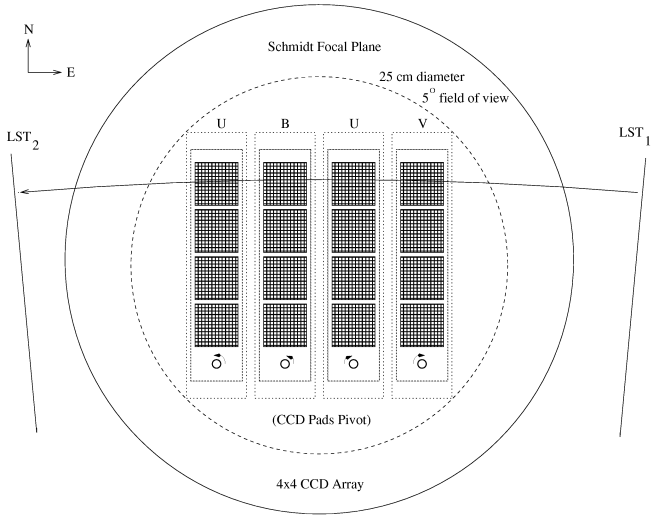

FIG. 1.—The QUEST camera has 16 CCDs mounted on invar pads that individually pivot, optimizing the CCD positions for drift scanning at different declinations. A filter covers each column of CCDs, so that nearly simultaneous photometry in *V*, *U*, *B*, and *U* is obtained as an object passes successively through each filter.

$r \approx f/\tan\delta$ (for small declinations δ and focal length f), while the differential traversal rate is $dv/d\delta \approx 15'' \text{ s}^{-1}\, d\cos\delta/d\delta \sim \sin\delta$. The result is that both difficulties become more severe at declinations farther from the equator. One possible solution is to scan along great circles, minimizing the transit-time difference across the CCD array, although this requires precision telescope tracking or mounting the CCDs onto movable stages (e.g., Zaritsky, Shectman, & Bredthauer 1996). Instead, we opted to simply extend the accessible declination range for parked-telescope drift scanning.

To correct for the curvature of the stellar paths, the CCDs are physically fixed to north-south pointing pads that individually pivot to align perpendicular to the star paths. The remaining sagitta, *s*, during an object's traversal of the length of the CCD, $l \approx 3$ cm, is given by $s \approx l^2/(8r)$. For $|\delta| < 20$ and *r* defined above, $s < 1$ pixel, thus an object essentially remains centered in a given CCD column as it crosses the chip. To compensate for the varying drift rate, CCDs scanning at different declinations are effectively clocked at different rates (see § 2.3). Even so, the size of our CCDs and the image scale limit our scan regions to $|\delta| \leq 6^\circ$; higher declinations will suffer image smearing greater than 1 pixel because of the change in sidereal rate across a single chip.

Drift scanning also imposes requirements on the data acquisition system. In part, this is because of the "real-time" response demands on the operating system to handle data transfers synchronized with the Earth's rotation. But drift scanning mainly poses interesting data management problems because of the sustained high data rate (16 CCDs × 15 rows s^{-1} × 4 kB $\text{row}^{-1} \approx 1$ MB s^{-1}) and the large data volumes that accumulate (≈30 GB per 9 hr night). Although 30 GB of data per night has recently become fairly tractable (consider Moore's law, that the doubling time for microprocessor performance is $\lesssim$2 yr), a planned second-generation camera will have an increased data rate (by a factor of 6), reemphasizing the importance of fast algorithms and online data processing and compaction.

We describe a very low-cost solution (~$10,000 for all computer hardware and well under 1 person-year of programming), made possible by distributing the workload and I/O across a cluster of commodity PC components and taking advantage of the relatively unique features of the QNX operating system (such as optimized, network-transparent message passing and full support for user-space device drivers). The system has been in use for about a year (including lab work and commissioning runs in early 1998) and has proved very reliable. Section 2 discusses the hardware setup, the use of QNX, and the online software (data acquisition, online analysis, and the user interface). The following sections present a fast, lossless image compression algorithm (§ 3), some initial data products (§ 4), and the conclusions (§ 5).

2. ONLINE DATA SYSTEM

2.1. Hardware Setup

The hardware for the online data system was purchased almost 3 years ago, based on a design driven by several constraints and strategies:

1. Minimize cost, even if this prohibits the use of traditional acquisition schemes (e.g., intelligent interface cards mounted on VME-bus Suns) and implies that the full raw data set cannot be captured on permanent storage (see § 2.4.2).
2. Harness commodity PC and networking technologies (which have the best price-to-performance ratio), by distributing workload and I/O in parallel across many subsystems.
3. Exploit the inherent parallelism of multiple (identical) detector arrays (i.e., load balancing is trivial).

The current configuration, which has undergone only minor upgrades since then, consists of seven single-board Pentium processors (each with multiple Ethernet interfaces), SCSI peripherals, and "off-the-shelf" ISA-bus cards for generating all system timing and interfacing to the stepper motor devices (the camera shutter and CCD pads). The parallel use of multiple processors, communicating over dedicated networks and writing to disks on separate SCSI buses, effectively provides performance scaling without resorting to systems with increased price-to-performance ratios. The hardware setup is shown in Figure 2 and outlined below,[2] while the choice of operating systems is rationalized in the next section.

1. Instrument control computer (4):

[2] Price at the time of purchase, from 1995 to 1998, is indicated.

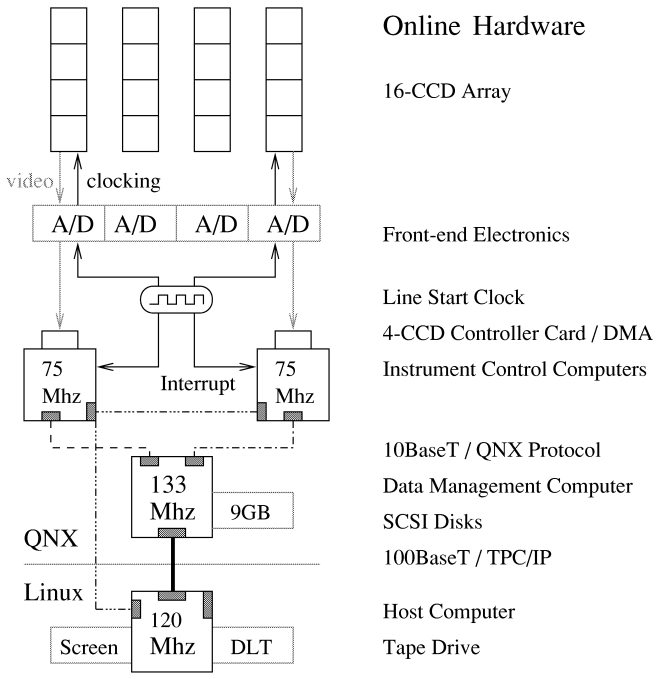

FIG. 2.—The data system hardware for controlling half of the CCD array (eight CCDs) is shown. There are seven dedicated Ethernet networks: a 10BaseT wire connecting the four ICCs (two are shown) and the host computer, a 10BaseT wire connecting each of the four ICCs to a DMC (one of two are shown), and a 100BaseT wire linking each of the two DMCs to the host computer. All networks run both TCP/IP and the QNX Networking Protocol.

a) 1 32 MB, 75 MHz Intel Pentium ($200 × 4)
b) 1 0.5 GB IDE disk (surplus)
c) 2 NE2000 Ethernet interfaces ($60 × 4)

2. Data management computer (2):
   a) 1 64 MB, 133 MHz Cyrix 686 ($250 × 2)
   b) 2 8 GB SCSI disks ($1000 × 2)
   c) 2 NE2000 Ethernet interfaces ($60 × 2)
   d) 1 SMC EtherPro100 Ethernet interface ($100 × 2)
3. Host computer (1):
   a) 1 64 MB, 120 MHz Cyrix 686 ($250)
   b) 2 4 GB SCSI disks ($2000)
   c) 2 SMC EtherPro100 Ethernet interfaces ($200)
   d) 1 NE2000 Ethernet interface ($30)
   e) 1 19 inch 1280 × 1024 DEC color monitor (surplus)
   f) 1 DLT 2000XT drive ($2500)
4. Device interfaces:
   a) 4 ISA-bus 4 CCD controller card
   b) 1 Oregon Microsystems Board ($900)
   c) 1 PC-TIO-10 National Instruments Board ($350)

Four of the processors are instrument control computers (ICCs), each dedicated to one row of four CCDs. Each has an ISA-bus CCD controller card (with onboard Direct Memory Access [DMA] hardware) that interleaves video output from four CCDs. To avoid designing our own custom interface boards, we adopted an "off-the-shelf," PC-based design by F. Harris. Serial connections over coaxial cable between the front-end electronics and the controller cards transfer CCD control commands and data. In addition, two of the ICCs contain a second ISA-bus card: an Oregon Microsystems Stepper Motor Controller board for positioning the camera shutter and CCD pads, and a National Instruments Timing I/O board for clocking the CCDs (§ 2.3.1).

Two of the processors are data management computers (DMCs), with a total of four 8 GB SCSI disks (disk I/O performance tests under QNX with a loaded CPU strongly favored bus-mastering SCSI over IDE). As described in § 2.3.2, each DMC collects and logs to disk data from two ICCs (eight CCDs total). At the end of the night's observations, the data are archived to a Digital Linear Tape device (DLT 2000XT, with 15 GB of raw data storage) mounted on the host computer. Point-to-point fast Ethernet links between the DMC and host computer ensure that the tape streams at its full 1.6 MB s^{-1} data rate, allowing 30 GB of data (compressed by a factor of 2) to be written in a few hours. The use of two DMCs sufficiently distributes the workload so that the full data set can be compressed and written to disk, while simultaneously running interactive analysis tools for the observer.

In addition, the use of the DMCs avoids the need to write to disks local to the ICCs, which would interfere with the CCD readout cycle. A bus-mastering device, including many SCSI disk and Ethernet interfaces, which does not relinquish the system bus within ~10 μs, can result in dropped data lines. (Timing constraints, due to the lack of buffering on the CCD controllers, are discussed in § 2.3.1.) However, during data acquisition, the only devices active on each ICC (in addition to the CCD controller card), are two NE2000 compatible, ISA-bus Ethernet cards with shared memory interfaces.

The final processor (the host computer), with a color monitor, provides a Tcl/Tk[3] (Tool Control Language/Toolkit; Welch 1995) graphical interface for system control and monitoring (see § 2.5.1). All of the PCs have operated reliably for years and survived shipment to the Observatorio Nacional de Llano del Hato (operated by CIDA) near Mérida, Venezuela; the only replacements required were for two SCSI disks that failed under warranty.

2.2. QNX Operating System

The QNX operating system, produced by QNX Software Systems, Ltd.,[4] since 1981, has been central to the feasibility of this project. QNX is used in applications ranging from nuclear reactor monitoring to machine vision tools on the Space Shuttle. We initially looked into QNX as an alternative to DOS for satisfying the time-critical data acquisition constraints. The numerous benefits uncovered in this work are that QNX (1) is

[3] http://www.tclconsortium.org.
[4] http://www.qnx.com.

a true real-time operating system (interrupt latency of ≈5 μs and context switch speed of ≈2 μs on a Pentium 100), (2) provides a familiar UNIX programming environment and shell interface, (3) has strong support for user-mode device drivers (see below), (4) is fully distributed (see below), (5) is an "Open" system, supporting numerous standards (e.g., POSIX, ANSI C/C++, and TCP/IP), and (6) is inexpensive for qualified academic projects.

The QNX operating system is designed as a set of cooperating (user-mode) processes, with the ~10 kB microkernel responsible only for process scheduling and low-level, network-transparent communication (message passing). As a result, "installing" a device driver is as simple as executing a user program. In fact, writing a device driver follows the normal program development sequence: the running driver can be stopped, modified, recompiled, stepped through with a debugger, and rerun (without rebooting). And any application level program, such as the driver, can access the hardware directly (i.e., without linking new code into the kernel, developing a special DLL or VxD layer, or sacrificing multitasking with memory protection). For example, functions are provided for device port I/O, and interrupt handlers for servicing hardware interrupts can be entirely written in C with access to all the global variables of the program in which they are embedded.

Because all operating system services (e.g., the filesystem manager) are accessed by message passing, and message passing is network transparent (i.e., independent of whether the recipient is local or remote), QNX is fully distributed. For many purposes, the numerous QNX nodes act as a single, logical computer. For example, process fork()-ing (with full inheritance of the environment), peripheral sharing, and interprocess communication are network transparent. In addition, process communication across the network is optimized: QNX avoids the overhead of TCP/IP, reliably providing the full network bandwidth (≈1.1 MB s^{-1} on 10BaseT Ethernet), with dynamic load balancing across multiple networks. Services (processes) on the network can be looked up by name, and messages can be broadcast to processes on all hosts or directed to a specific process (with an unique, network-wide process ID).

To take advantage of the distributed nature of QNX, the online software was created using the client-server model. By sending and receiving messages, multiple processes (local and remote) can coordinate work on some task. When a client requests a service (such as compressing data and storing it to disk, § 2.3.2), the client does not need to know whether the message is handled locally or remotely. In fact, the number of nodes on the network can change without requiring modification of the software. In recent years, some of these features of QNX have become available in an add-on, network messaging software layer (e.g., PVM), although with QNX it is native (efficient) and involves less development work.

The principal disadvantage that we encountered in using QNX is that certain software tools and hardware are not supported. For example, IRAF is not supported under QNX, and there were complications in using our DLT drive with QNX (although this problem has apparently been solved with newer releases of QNX). A simple remedy for these problems was to run Linux (Redhat) on the host computer and provide a TCP/IP connection to the QNX network (the coexistence of multiple networking protocols on the same physical medium is not a problem).

2.3. Data Acquisition

2.3.1. System Timing

Underlying the drift-scan data acquisition is accurate digital timing. Because "tracking" is accomplished by shifting deposited charge across the CCD, as opposed to moving the telescope, clock instability problems (>1 in 10^5) will noticeably degrade the image quality and astrometry (in the R.A. direction). In addition, the CCDs need to be clocked simultaneously, to within ~1 μs, to avoid CCD cross-talk and increased readout noise, but the required line clock rate (i.e., the apparent sidereal rate in pixels) varies greatly over the ≈3° declination extent of the CCD array.

Our solution, suggested by S. Shectman, is to clock the CCDs synchronously at the same rate but occasionally drop clock pulses at different intervals to achieve an effective variation in the readout rate with declination. That is, a given CCD drops every Nth clock pulse, where N is determined from the difference between the "base" clocking rate for the array ($\Omega_{\rm base}$, in rows s^{-1}) and the desired effective clocking rate ($\Omega_{\rm eff}$) for that CCD: $N = \Omega_{\rm base}/(\Omega_{\rm base} - \Omega_{\rm eff})$. The desired effective rate follows from the Earth's rotation rate ($\Omega_\oplus = 15\farcs041069\ s^{-1}$), the pixel scale ($p = 1''$ $pixel^{-1}$), and the central declination δ of the CCD: $\Omega_{\rm eff} = \Omega_\oplus/p \times \cos\delta$. The base rate is chosen as the largest desired effective rate. A typical line drop interval is $N \sim 1000$ lines (i.e., ~1 minute), during which time the measured R.A. coordinates of objects will deviate linearly from their true positions (by 0 to almost 1 pixel) and must be corrected.

To accomplish the above clocking scheme, we use the National Instruments Timing I/O (TIO) board with some external logic (AND gates and line drivers). The 5 MHz clock provided by the crystal oscillator on the TIO board is divided down to the required base rate using a programmable counter on the board. This base clock (~15 Hz) is fed into four counters (one per row of CCDs) programmed to change signal level after a certain number of counts (N given above). These provide the line "drop" signals which are then AND-ed with the computer and CCD controller signals to suppress the line starts at the required intervals.

With a base clock rate of ~15 Hz, the TIO board generates line start signals for the front-end electronics and the data acquisition PCs (ICCs) every 60 ms. To avoid an extra digital I/O board per ICC, the line start is delivered directly to the IRQ pin on the parallel port of each ICC. The line start signal for the CCDs is delayed relative to that delivered to the PCs

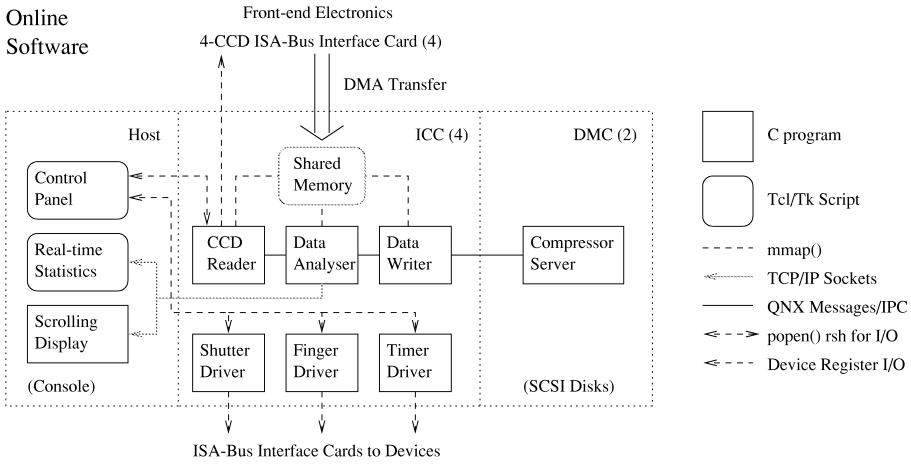

FIG. 3.—The data system software, described in the text, is distributed across the seven online computers

by about 5–10 ms to allow the software to prepare for the DMA data transfer (§ 2.3.2).

The data comes in bursts, during which the CCD is clocked out at 50 kHz, giving a total readout time for one line of 40 ms and a time between individual DMA writes of 20 μs. Since the interface card had no data buffer until recently (and now only has a small one), the timing of the data places stringent requirements on the PC operating system and data acquisition software: the software has only a 20 ms window to setup the DMA registers for the next line transfer. In addition, no other device in the PC can grab the system bus for more than ~10 μs.

The base clock is also used as a trigger signal for eight other counters (four for the CCD electronics, four for the ICCs) that are set up in a one-shot count mode driven off the 5 MHz board clock. These counters produce a square wave of arbitrary duty cycle and arbitrary start time relative to the trigger signal, allowing us to generate *all* desired electronics signals with digital accuracy (1 in 5×10^6).

2.3.2. DMA Data Transfer

The conventional technique of "single-buffered" data acquisition is a sequential process in which the data samples are first transferred from the I/O device to system memory and then processed and/or logged to disk. Drift scanning, however, relies on the continuous acquisition of quantities of data greatly exceeding the available memory, hence "double-buffered" (continuous) mode data acquisition is used. In this mode, the memory buffer is arranged as a circular buffer divided into sections called blocks. The processor can then concurrently access and process blocks of already acquired data while data transfer proceeds uninterrupted in the background to other blocks.

The actual data transfer from the CCD controller to system memory is initiated by an interrupt service routine that is triggered at the apparent sidereal rate by the clock pulse on the ICC's parallel port. The interrupt handler programs the DMA controller on the card with the transfer size and address of the next block, allowing the controller to bypass the CPU and transfer four interleaved CCD rows directly to memory. The Watcom C/C++ environment under QNX facilitates these basic data acquisition tasks by providing C library calls (accessible from user-space programs), for allocating DMA-safe buffers, installing interrupt vectors, reading/writing device registers, etc. In fact, the basic data acquisition software[5] was written in several days.

On each ICC, three concurrent processes access the (shared-memory) circular DMA buffer asynchronously with respect to the line-by-line readout (and each other). The processes are informed of incoming data using QNX Proxies, a type of buffered, nonblocking message used for event notification. The processes (depicted in Fig. 3, and outlined below) are the CCD reader, the data handler, and the data analyzer. Multiple processes are used to increase modularity and fault tolerance (the data acquisition continues even if the analysis stalls or crashes). Also, the data analysis and handling can easily be offloaded to remote CPUs (by replacing the shared memory reference with message passing).

The basic tasks of the three concurrent processes are

1. CCD reader:
   a) initialize CCD controller (binning etc.)
   b) preallocate DMA buffers
   c) spawn data handler/analyzer
   d) register interrupt handler
   e) register QNX proxy
   f) send status reports to host computer
2. Data handler:
   a) name lookup of disk server

[5] ftp://www.astro.yale.edu/pub/sabbey/daq.tar.gz.

b) pack data lines into QNX messages
c) send to disk server

3. Data analyzer:
a) descramble interleaved data row
b) data analysis (histograms, objects, etc.)
c) send data/statistics to host computer

The data handler packs the data into messages for transmission to a server process running on a DMC, which compresses the data (see § 3) before writing to disk in pseudo-FITS format (i.e., an ASCII FITS header followed by compressed image data, with a ".fits.enc" filename extension). The data analyzer process does object detection and histogram analysis (discussed in § 2.4), updating the Tcl/Tk image statistics window on the host computer. It also sends the rebinned and byte-scaled data over a TCP/IP connection to the scrolling image server (see § 2.5.2).

2.4. Online Analysis

There are many reasons for wanting real-time (or near real-time) data analysis:

1. To monitor the data quality, both to confirm the camera operation and to make observing strategy decisions (useful diagnostics include the seeing and background level as a function of time for each CCD).
2. To identify brightness- or position-variable objects for immediate follow-up work (e.g., transients such as supernovae and near-Earth asteroids).
3. To provide inputs for a "fast-start" of the off-line software.
4. To restrict the data flow near the source, reducing the demands on the downstream data pipeline.
5. To avoid data backlog and indigestion.

Here we discuss the fourth issue, which was of overriding importance during the design phase of this project. Our original storage devices (16 GB of disk space and one DAT tape drive) were inadequate for handling the 30 GB of raw image data per night. Our current solution is to archive the full (compressed) data set to DLT tape. However, extracting a hierarchy of data subsets (such as image "stamps" described in § 2.4.2) is still useful for distribution and fast-access purposes (corresponding to a computer's memory hierarchy), and could become crucial to the online system for a more ambitious second-generation camera.

We very briefly describe the online analysis, which includes fast catalog creation and object stamp cutting.

2.4.1. Fast Catalog Creation

Most software for the analysis of extragalactic survey data includes the following steps (e.g., Jarvis & Tyson 1981; Irwin 1985; Bertin & Arnouts 1996):

1. Analyze image brightness histograms to determine the background level and sigma.
2. Convolve image with a two-dimensional, Gaussian-like detection filter.
3. Identify sets of connected pixels above some brightness threshold.
4. Examine object contours at multiple brightness thresholds to deblend overlapping objects.
5. Measure positions, shape parameters, and magnitudes.

We followed this procedure, with provisions to run the code online. In particular, the software needs to process 4×2048^2 pixels every ≈140 s, on a 32 MB (75 MHz) Pentium that is managing data acquisition for a row of CCDs. This requires fast algorithms that can tap into the data stream (before it has reached permanent storage) and analyze data in small sliding windows from four CCDs in parallel.

A histogram per CCD column is incremented with each incoming row of data and analyzed every 4096 rows (a time step of about 5 minutes). The histograms are 200 bins wide, centered on the estimate of the background level from the previous time step (overflow and underflow bins register outliers). The linearly interpolated quartiles—the 25th (q_{25}), 50th (median), and 75th (q_{75}) percentiles—and the first moment of the 2.3 σ clipped histograms (to approximate the mode, see Da Costa 1992) are measured to obtain the following:

1. Column i relative sensitivity for flat-fielding:

$$s\,[i] = (m\,[i] - b\,[i]) \,/\, \langle \boldsymbol{m} - \boldsymbol{b} \rangle \,,$$

using column i mode $m\,[i]$, column i bias $b\,[i]$, and the corresponding arrays (all columns), $\boldsymbol{m}$ and $\boldsymbol{b}$.

2. Mean sky brightness B:

$$B = \langle \boldsymbol{m} - \boldsymbol{b}/\boldsymbol{s} \rangle_{\rm mode} \,,$$

where $\boldsymbol{s}$ is the sensitivity array corresponding to $s\,[i]$ above.

3. Column i quartile sigma:

$$B_\sigma\,[i] = 0.7415 \times (q_{75}\,[i] - q_{25}\,[i]) \,,$$

following Weir et al. (1995).

4. Column i object detection threshold:

$$t\,[i] = (B + N_\sigma \langle \boldsymbol{B}_\sigma \rangle) \times s\,[i] + b\,[i] \,,$$

where N_σ is adjusted to set the threshold.

The image is convolved on the fly with the canonical, Gaussian-like detection filter (see Irwin 1985), and a one-pass object detection routine is applied. Objects are represented as linked lists of horizontal runs of pixels above the detection threshold. With each incoming (convolved) row of data, the new "hot" runs are identified and used to either (1) extend an existing

object, (2) start a new object, or (3) merge previously disconnected objects (by pointing the tail of one linked list to the head of another). To track the locations of objects being built up, an array of pointers as wide as the image indicates which object, if any, is continuing in each column. An object that was not accessed while processing the current data row has completed and will be kept if it contains more pixels than some threshold.

The bias and flat-field corrections (vectors) are then applied to the object pixels (only), and moments are measured (e.g., Stobie 1980) to determine the object centroid and shape parameters (the ellipse major and minor axes and position angle). In addition, an isophotal magnitude is calculated from the sum of the background-subtracted counts of the object pixels, and an object region description, represented as a bitmap with object pixels (only) set to 1, is created.

As an improvement over the isophotal magnitudes, fast multiaperture photometry was developed for the online software (as part of a data verification pipeline to run in the morning). The objective is to measure total magnitudes using circular aperture photometry with curve-of-growth corrections (see Stetson 1990) as a function of time and position (i.e., varying within an R.A.-decl. grid). There is a trade-off however between calibrating the curve of growth with the high signal-to-noise ratio measurements of the brightest (unsaturated) stars and providing a grid of aperture corrections with good resolution (which relies on the more numerous fainter stars).

The procedure developed is to measure all objects through a sequence of apertures of increasing diameter (e.g., 3″–30″), terminating the sequence when the standard error of the aperture measurement exceeds a threshold of, e.g., 0.1 mag. All of the resulting magnitude measurements within a given grid block are then used to obtain the curve-of-growth corrections for that block. In particular, the magnitude "correction" between consecutive annuli in the sequence is obtained from the mode of the magnitude differences between those two annuli (see Stetson 1990 for a sophisticated treatment). For each object, the local corrections are then applied to the aperture magnitude of smallest error to normalize the measurement to the largest aperture.

To obtain multiaperture photometry for $\sim 10^6$ objects in a few minutes on a slow PC, an unconventional technique of adding "shell sums" onto a "core sum," with a mask to track the pixels already added to the core sum, was used:

```
clear mask
core_sum = 0
shell_sum = 0

foreach aperture, small to large
{
 foreach pixel in aperture bounding box
 {
  if (pixel is masked)
    Continue foreach
  else if (pixel entirely within aperture)
    Update core_sum, mask the pixel
  else if (pixel partly within aperture)
    Update shell_sum using fractional weight
 }
 sum = shell_sum + core_sum
 shell_sum = 0
 calculate mag, magErr (using sum)
 If (magErr gt MAXERR) done
}
```

This is efficient because the typical (i.e., faint) object will be measured only through the few smallest apertures (after which the magnitude error becomes large), and the procedure can "short-circuit" early. The more common (but reasonable) approach is to maintain a separate aperture sum for each aperture size and loop over all pixels in the bounding box of the largest aperture, checking at each pixel which sums need to be updated.

A prototype for fast object deblending was also developed (but is currently relegated to offline processing). The deblending follows the isophotal method outlined in the literature (e.g., Beard, MacGillivray, & Thanisch 1990; Bertin & Arnouts 1996) but is implemented using recursion with tests for multiple relative maxima to avoid isophotal analysis on objects without a saddle point. Essentially, the deblender runs the object detection routine (described above) at increasing brightness thresholds, recursively calling itself to ascend multiple brightness peaks (objects) detected above the current threshold.

If `object` is a C typedef for a data structure that contains the object region description, and the C function `detect()` searches for connected pixel objects (children) within the (parent) `object` region, then the deblender works roughly as follows (in pseudo C code):

```
int nobjs = 0;
object objs[NMAX];

main
{
 object child, parent;
 if ((child = deblend(parent, thresh)) != -1)
   objs[nobjs++] = child;
}

object deblend(object parent, float thresh)
{
 int nchildren;
 object child, children[NMAX];

 if (single_relmax(parent)) /* bail out early */
   return(parent);
```

```
 do { thresh = thresh * scale;
      nchildren = detect(parent, thresh,
        children);
      parent = children[0];
    } while (nchildren == 1);

 for (i = 0; i < nchildren; i++) /* recursion */
   if ((child = deblend(children[i], thresh))
       != -1)
     objs[nobjs++] = child;

 return
}
```

Extended objects without a saddle point will pass through the deblender untouched but can be flagged for multiple profile-fitting analysis.

2.4.2. Image "Stamp" Cutting

The option of storing image "stamps" that encapsulate the object detection isophotes, as opposed to storing our full images that are ≈90% background, was a major incentive for the online cataloging work described above. The method of data acquisition originally envisioned is "stamp cutting using lead-chip object detections." By buffering the pixel coordinates of objects detected in the (most sensitive) *V* filter (the lead chip of each row), stamps can be cut in the *U*- and *B*-band drift-scan images (at time intervals given by the chip spacing in the R.A. direction) for objects not detectable in *U* until the data has been coadded. The stamp size is obtained by enlarging the bounding box of the detection isophote by a user-specified constant factor, e.g., 1.5 (see Fig. 4).

With a surface density of ≈5000 objects deg^{-2} (to the 3 σ detection limit of $m_V \approx 22$) and ≈2 deg^2 of sky between the lead chip and last chip of each row, the list of buffered positions contains $\sim 10^4$ objects and cannot be scanned frequently for objects "falling off" the trailing chips. On the other hand, the image data are only buffered briefly in small sliding windows (512 rows), prohibiting large delays (especially for stamps comparable in size to the buffer). An efficient solution, which simplifies the coordinate transformation of object positions between chips, is to have a separate stamp cutter for each CCD access the circular list of lead-chip object positions asynchronously.

The list of lead-chip object positions is sorted by the object stamp-ending row number (the default output of object detection). On each row of incoming data, a stamp cutter for each CCD is awoken and allowed to cycle through the list, cutting stamps until it reaches an object (stamp) that has not completed in the drift-scan data for its CCD (at which point the cutter marks its position and goes to sleep). A stamp that completed in row number n_1 in the lead-chip drift-scan image is ready to cut in the kth CCD of the row when $n = n_1 + (k - 1) \times \Delta_{\rm chip}$, where n is the number of data rows that have been read by the kth CCD and $\Delta_{\rm chip}$ is the CCD spacing in the R.A. direction in pixels.

For the prism data (and low Galactic latitude photometry), stamp cutting would not be effective (in fact, the data size could be expanded due to overlapping stamps). Our original plan was to co-add prism data from each row of CCDs online (reducing the output data size by a factor of 4). For the current camera, however, lossless compression (§ 3) was found sufficient and the co-adding will be done offline.

2.5. User Interface

2.5.1. Tcl/Tk Control Panel

Although the online data system is distributed across more than 15 processes running on seven networked PCs, a Tcl/Tk graphical user interface (GUI) running on the host computer allows the user to remain pleasantly unaware of this. More generally, the purpose of the GUI is to

1. provide a self-obvious, high-level abstraction for controlling and monitoring the entire system (using scrolling status logs, image displays, and analysis statistics);
2. act as a central repository of state information that is fed to starting programs (such as the observation index, the CCDs being read, the clock rate, the CCD-pad rotations, etc.);
3. receive periodic feedback from all processes for monitoring the health of the system;
4. handle basic sequencing logic (e.g., rotate the CCD pads, set the line clock, open the shutter, then simultaneously kick four CCD readers), or to preprogram observing sequences; and
5. check for reasonable inputs and guide the user into noncontradictory selections (e.g., when "drift-scan" mode is selected the exposure time entry box is disabled, and an entry box for the number of lines to read is enabled).

A potential mode of failure in a distributed system with many communicating processes is when a process enters a "hung" state without actually crashing. Deadlock ensues as the cooperating processes wait for a message or some other event triggered by the stalled process. If this occurs, 3 above can be useful in identifying the problem, although as a result of 2, a global restart is trivial and only takes seconds. To accomplish 3, all processes either report directly to the GUI (see below) or update a global log file to which the GUI has a file I/O event handler attached (Tcl command `fileevent`).

The basic parameters that are specified (using the GUI) for each observation are the data acquisition mode (drift-scan, snapshot, or simulation), the compression type (lossless, stamps, catalog, or none), the chips to read (various combinations are possible), the chips for real-time analysis (lead chips, all chips, or none), the number of rows in the drift scan (typically $\approx 4 \times 10^5$ for an ≈8 hr scan), the number of rows per frame (typically 2048), the observation index (a running

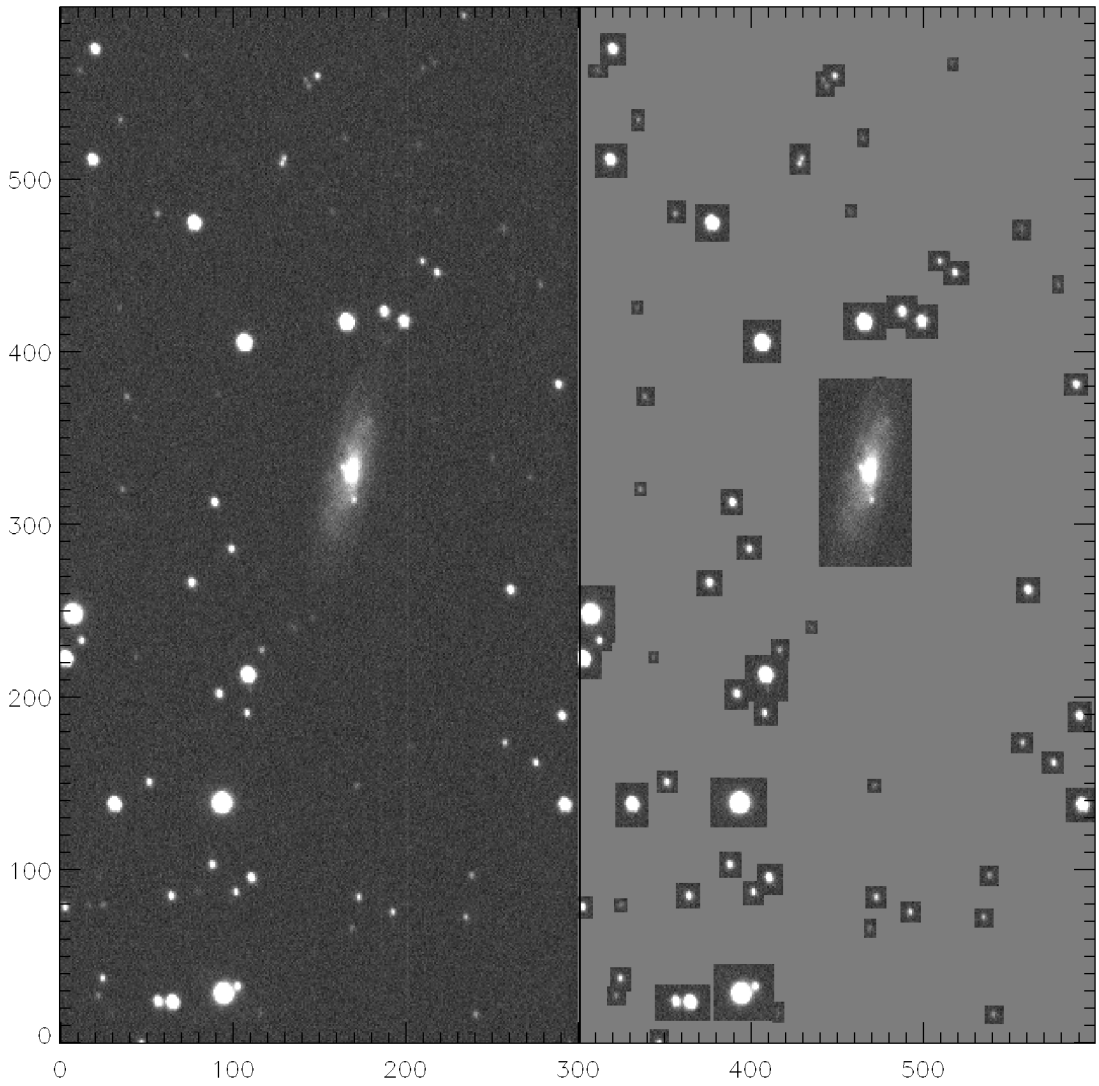

FIG. 4.—An example of image "stamps" that can be extracted in real time from the drift-scan data to reduce the quantity of data to be archived by factors of ≥10. Object detections in the lead chip (*V* filter) of each row are buffered to cut stamps for all chips in the row. Each stamp is accompanied by measurements of the encapsulated object (see the text) and a bitmap indicating the object pixels. An automated triangle-matching routine (based on Valdes et al. 1995) is run at the completion of data taking to convert pixel coordinates to sky coordinates.

integer), the clock rate, and the CCD-pad positions. (The drift-scan image from each CCD is stored as a sequence of FITS frames, of some convenient number of rows specified by the user, that fit together seamlessly.)

The GUI interpolates the parameters into a command string of the form "`| rsh qnx1 ccdReader -mode scan -nlines 90000`" on which it does a (process) open for I/O and uses the Tcl `fileevent` command to register a procedure to be invoked when data (e.g., status messages) are available on the pipe. This was the simplest solution for allowing the GUI to control the preexisting, interactive command line CCD reader (and other hardware-dependent programs) without modification, over the network. A potential drawback is that a broken connection cannot be reestablished (without aborting the current observation), although this has not been a problem.

2.5.2. Real-Time Scrolling Image

The real-time scrolling display, written in ANSI C using X11 library calls, presents a view of the sky drifting over the telescope from the perspective of a given CCD. (To deal with having only one console for the online system, the user can "change the channel" as desired, indicating the CCD to display data from.) The select() system call is used to avoid blocking I/O on socket file descriptors within the X event loop. Repeatedly copying the image in the X window to an offscreen

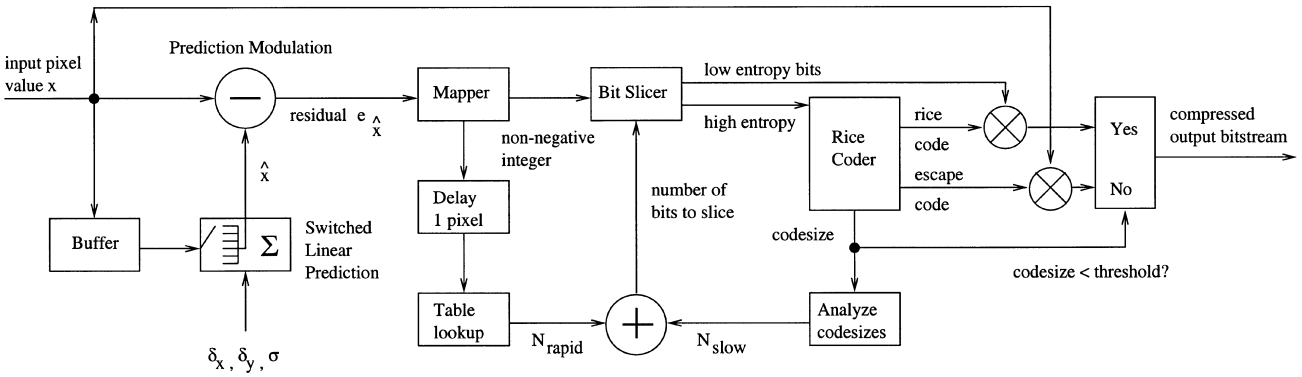

FIG. 5.—A block diagram of the adaptive compression algorithm described in the text

buffer (a pixmap) using XCopyArea(), and then redrawing to the window with an offset of one row and using XPutImage() to fill in the new data row allows for smooth animation.

To keep network traffic to a minimum, token passing on the ICCs coordinates which client transmits data to the server, and the data are rebinned by a factor of 4 × 4 and byte-scaled before transmission. The byte-scaling uses the robust determination of the background level and σ as a function of time from the histogram analysis to linearly map 2 byte pixel values in the range $-3\ \sigma$ to $+10\ \sigma$ onto a byte value. For full resolution, interactive inspection of the data, several X applications (run on the DMCs, but displayed on the host) were created. Useful tools include radial profile fits to determine the FWHM, curve-of-growth aperture photometry, measurement of basic image statistics, and plotting of one-dimensional profiles of image regions selected using the mouse.

An example of multiplexing input from several network socket connections onto a real-time scrolling display is available in a stripped-down program on anonymous ftp.[6] The program is written in ANSI C with X11 library calls, so it is quite efficient and portable to most UNIX boxes and provides a generic BSD socket interface for clients anywhere on the local network or even the Internet. A toolkit was not used because high-level, prefabricated widgets were less important than having fast, direct access to the Xlib graphics primitives.

3. COMPRESSION

The basic principle behind data compression is to eliminate redundancy, that is, to store a decorrelated representation of the data (see Rabbani & Jones 1991 and Sayood 1996 for general introductions). Numerous techniques have been developed for use in astronomy, including methods that encode the coefficients of transformed image blocks (e.g., White & Percival 1994; Press 1992), methods that segment the image by bitplane to encode each layer separately (Veran & Wright 1994), and methods that first segment by picture elements (Huang & Bijaoui 1991). Lossy programs achieve compression ratios of ~100 by thresholding noise (and image features that resemble noise) to a constant level, while lossless programs are exactly reversible but compress by only a factor of 2 or 3.

A lossless program was created for incorporation into the online system, improving the effective data storage and throughput by more than a factor of 2. The design constraints called for a program that is fast (~1 MB s^{-1} on a 133 MHz Pentium), easily implemented, has a small memory footprint, allows streaming of data to disk or tape, and compresses a variety of astronomical images well, including high and low surface density photometry and slitless spectroscopy. Our approach was to make a rapidly adaptive version of a technique[7] (Rice 1991) that is based on Rice coding (Rice & Plaunt 1971) the prediction-modulated pixel data.

More specifically, the basic sequence of operations to compress the current pixel is (see Fig. 5):

1. Form a prediction, p, for the current pixel value, x, using a linear combination of already transmitted pixel values (specified below).

2. Map the prediction residual, $\Delta = (x - p)$, onto the nonnegative integers, sequenced by decreasing frequency of occurrence (i.e., "Standard Form"). Prediction residuals are often assumed to follow a symmetric Laplacian distribution with zero mean, for which the following simple mapping ($\Delta \rightarrow \delta$) is effective:

$$\delta = \begin{cases} -2\Delta - 1 & \text{if } \Delta < 0; \\ 2\Delta & \text{if } \Delta \geq 0. \end{cases}$$

3. Partition the mapped residual, δ, into N low entropy and $(16 - N)$ high entropy bits. N is calculated as defined below;

[6] ftp://www.astro.yale.edu/pub/sabbey/xscroll.tar.gz.

[7] See also M. W. Richmond & N. E. Ellman (1996) (http://a188-1004.rit.edu/richmond/rice/rice.html).

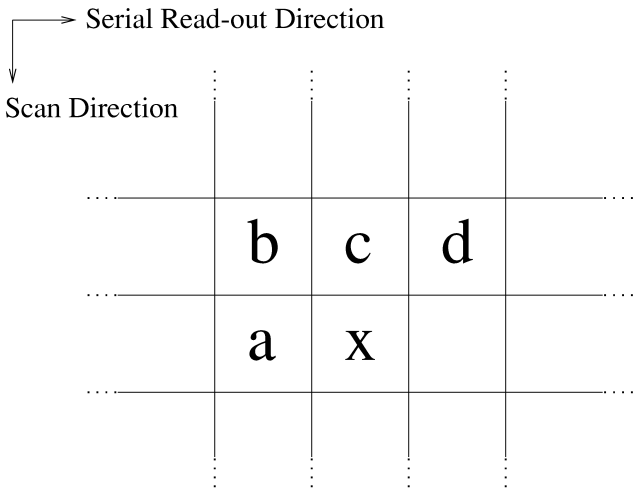

FIG. 6.—The pixel labeling used in the text to explain the adaptive selection of the predictor used in the compression algorithm.

see Yeh, Rice, & Miller (1991) for a discussion of the optimal partition size as a function of the entropy of the data.

4. Direct the N low-order bits (the "noise bits") of the mapped residual to the output stream and encode the remaining $(16 - N)$ high-order bits using Rice coding, a variable length prefix code (no code is a prefix of another). If the Rice code is larger than 12 bits (set empirically), then an escape code and the uncompressed pixel are transmitted instead.

The decoder program applies the above steps in reverse to uncompress the data stream: the sliced noise bits are concatenated with the Rice-coded value to form the mapped prediction residual and the current pixel value is deduced.

We use the simplest Rice codebook, a Fundamental Sequence, for which codeword i is given by $(i - 1)$ 0's followed by a 1. For example, the four most probable values will be assigned the bit representations (codes): 1, 01, 001, 0001. The decoder can "instantaneously" parse the code stream by splitting on 1's. If the probability distribution function (PDF) is heavily skewed toward the most probable value, then this representation will require fewer bits than the fixed length binary code: 00, 01, 10, 11. For example, a four symbol memoryless source with symbol probabilities $P_{1,2,3,4} = (0.70, 0.15, 0.10, 0.05)$ will require an average of only 1.5 bits per value (with the above Rice codes) instead of 2.

Rice coding is one of a spectrum of statistical coders that trade off simplicity and speed for the ability to optimally encode samples from a more continuous range of PDFs. The definition of optimal follows from Shannon's noiseless coding theorem, which says that the minimum average codelength is bounded from below by the source entropy:

$$H = -\sum_i p_i \log_2 (p_i) \text{ bits symbol}^{-1},$$

where $-\log_2 (p_i)$ is the amount of "information" (in the sense of uncertainty) and p_i is the probability of the ith symbol. Thus, an optimal encoder will assign a codeword of length $-\log_2 (p_i)$ bits to a symbol of probability p_i, that is, the length is proportional to the information content. For a PDF of the form $P_n = 2^{-n}$ $(n = 1,2,3, \ldots)$, the optimal codeword lengths will be integers. In this case, prefix codes such as Rice and Huffman (Huffman 1952), which have a single binary codeword for each symbol, will be optimal.

To recover optimality in the general case, arithmetic coding (see Witten, Neal, & Cleary 1987) can be used to effectively assign a single probability and "codeword" to the entire data sequence, making the overhead of a fractional bit negligible. In practice, however, Rice coding achieves a coding rate (average number of bits per pixel, or bpp) close to the source entropy (Yeh et al. 1991), and arithmetic coding is crucial only for highly skewed PDFs in which the entropy is $\ll 1$ bpp (prefix codes require a minimum of 1 bpp). In addition, Rice codes, which are a special case of Huffman codes, are easily made adaptive to changing source probabilities, unlike Huffman codes.

To extend this technique of Rice coding the most significant bits of the differential signal, we allow (pixel by pixel) adaptivity in the linear prediction (step 1 above) and in the partitioning of the mapped residual (step 3). To adaptively select an appropriate model for prediction, approximations to the local gradient in the x- and y-directions are formed using two-point differences and compared with a threshold based on the noise level: $\epsilon = 4 \times N_s$ (see below). Thus, with the pixel labeling shown in Figure 6, the prediction p, given $\delta_x = |c - b|$ and $\delta_y = |a - b|$, is

$$p = \begin{cases} (a + b + c + d)/4, & \delta_x < \epsilon,\ \delta_y < \epsilon; \\ a, & \delta_x < \epsilon,\ \delta_y \geq \epsilon; \\ c, & \delta_x \geq \epsilon,\ \delta_y < \epsilon; \\ a + (c - b)/2, & \delta_x \geq \epsilon,\ \delta_y \geq \epsilon,\ \delta_x < \delta_y; \\ c + (a - b)/2, & \delta_x \geq \epsilon,\ \delta_y \geq \epsilon,\ \delta_x \geq \delta_y. \end{cases}$$

Higher order switched predictors were not more effective, as might be expected from studies (e.g., Habibi 1971) showing greatly diminished returns when more than three neighboring pixels are used in the prediction.

In homogenous image regions (the first case above), the mean of several neighboring pixels will be selected as the predictor; otherwise the direction and order of the predictor will be selected based on the local correlation (an extension of Graham's Rule; see Netravali & Limb 1980). For example, pixel values in a CCD bad column will be predicted using the adjacent previous row value, while a higher order predictor in the dispersion direction will be used to predict bright spectrum pixels in the objective prism data. Only previously transmitted data are used so that the decoder can apply identical logic without the need for transmission of side information (backward adaptive prediction).

To allow adaptivity in the partitioning of the mapped residual (step 3), the partition size N is obtained from the sum of slowly varying and rapidly varying terms ($N = N_s + N_r$). The slowly varying contribution accounts for the background noise level; the rapidly varying term accounts for residual structure not captured by the linear prediction. The slowly varying term is updated by $+1/0/-1$ every $P \times Q$ image block, e.g., 16×16, based on a count, c, of the 1-bit Rice codes sent in the previous image block. Empirically, a simple rule was found to be effective: if ($c < 0.25 \times P \times Q$) then increment N_s, else if ($c > 0.75 \times P \times Q$) then decrement N_s. In other words, N_s is varied slowly such that the probability of Rice codeword 0 is kept near the optimal value of 0.5.

The rapidly varying (pixel by pixel) contribution, N_r, is obtained by a table lookup using as an index the mapped prediction residual from the previous pixel, right-shifted by N_s bits. For prediction residuals not dominated by noise, the PDF is often assumed to resemble a Laplacian of zero mean and variance σ^2, with the discrete probability function

$$p_{\sigma^2}(\Delta) = \int_{\Delta-1/2}^{\Delta+1/2} \frac{1}{\sqrt{2\sigma^2}} \exp\left(-\sqrt{\frac{2}{\sigma^2}}|\Delta|\right).$$

The σ^2 that maximizes the probability of the prediction residual Δ is obtained by solving the equation $\partial p_{\sigma^2}(\Delta)/\partial(\sigma^2) = 0$, yielding an optimal variance of $\sigma^2 = 2\Delta^2$. With the PDF specified, one can evaluate the entropy and use Yeh et al. (1991) to specify N_r as a function of the (mapped) prediction residual (i.e., $N_r = 0$ for $H < 2.5$ bpp, $N_r = 1$ for $H = 2.5$–3.5 bpp, etc.).

The value of N_r is typically 0, except in image regions containing relatively bright sources (i.e., where large prediction residuals become more probable). In these regions, N_r responds quickly to the increased entropy caused by the residual structure, and increases the number of "noise" bits sliced. This often allows the program to avoid sending very large (nonoptimal) Rice codes and escape codes. Similarly, the calculation of N_r effectively allows the encoder and decoder programs to predict when an escape code would be used, often making the escape code unnecessary. For example, if $N = N_s + N_r > 15$ bits, then the encoder will simply transmit the full pixel value (this case is not shown in Fig. 5).

The performance of the compression program was evaluated using a set of "standard" astronomical images[8] compiled for testing photometry software (Murtagh & Warmels 1989). Compared with the transform-based `hcompress`[9] program (used in lossless mode), the adaptive predictor gave increased compression ratios on all images (from 2% to 22% improvement, with median and average improvements both of 11%). Compared with the lossless output of `fitspress`,[10] the improvement ranged from -1% to 67%, with median and average improvements of 12% and 25%, respectively.

With respect to algorithms developed for online work (see the design criteria above), the improvements are more significant. However, a central reason for developing our own routine is that we were unable to find any lossless compression program that was fast enough and suitable for use in the online system. For example, the adaptive predictor program is ≈11 times faster than `fitspress` on our data (≈6 times faster than `hcompress`), and does not depend on user-tuned parameters to maintain high performance. A stand-alone version of the adaptive predictor program (in beta release), and the evaluation procedure and results in detail, are available from anonymous ftp.[11]

4. INITIAL DATA PRODUCTS

The camera was installed and commissioned in early 1998 at the Observatorio de Llano del Hato, near Mérida, Venezuela. The site has an altitude of 3600 m and a latitude of 7° north, for excellent coverage of about $\frac{1}{3}$ of the sky centered on the celestial equator. All systems have undergone extensive testing and were shown to be fully operational for extended observing runs. In addition, the drift-scan data obtained has a circularly symmetric PSF, without signs of trailing or other distortions. However, the FWHM of the PSF is typically 2.″5 in the V band, whereas the typical atmospheric seeing expected is 1.″5. Studies attribute this to "dome seeing" (due to excessive heat generated by the front-end electronics box), and steps are currently being taken to duct the heat away from the telescope and increase the total air flow in the dome.

Nonetheless, from the inspection of several hours of science-grade commissioning data taken in 1998 late January, we can confirm the performance of the camera and identify hundreds of interesting emission-line objects. A small portion of a drift-scan image with the objective prism mounted is shown in Figure 7. Additional examples of one-dimensional (extracted) spectra are shown in Figure 8. The 3.°2 objective prism provides 401 Å mm^{-1} dispersion at Hϵ (∼10 Å spectral resolution in the blue). The data shown were taken without any filter (i.e., a bandpass of ≈3800–10000 Å), although we are currently using a hot mirror to cut off beyond 7000 Å (reducing the sky background and the problem of overlapping spectra). Software to automatically extract and co-add spectra from different CCDs and scans is in progress.

The selection of quasars and other interesting sources using slitless spectroscopy complements well the selection by photometry. For example, a reddened quasar lensing system, or a $2.5 \lesssim z \lesssim 3$ quasar close to the stellar locus in a color diagram, might go unnoticed in the photometry but be identified from the emission lines in the spectroscopy. In addition, a high-dispersion prism survey (sufficient to resolve clearly the Balmer absorption series) with broad bandpass coverage (≈3800–7000 Å) and CCD detectors (as opposed to photographic plates) is unique and promises a diversity of applications.

[8] ftp://iraf.noao.edu/iraf/extern/focas.std.tar.Z.

[9] http://www.stsci.edu/software/hcompress.html.

[10] ftp://cfata4.harvard.edu/pub/fitspress08.tar.Z.

[11] ftp://www.astro.yale.edu/pub/sabbey/encode.tar.gz.

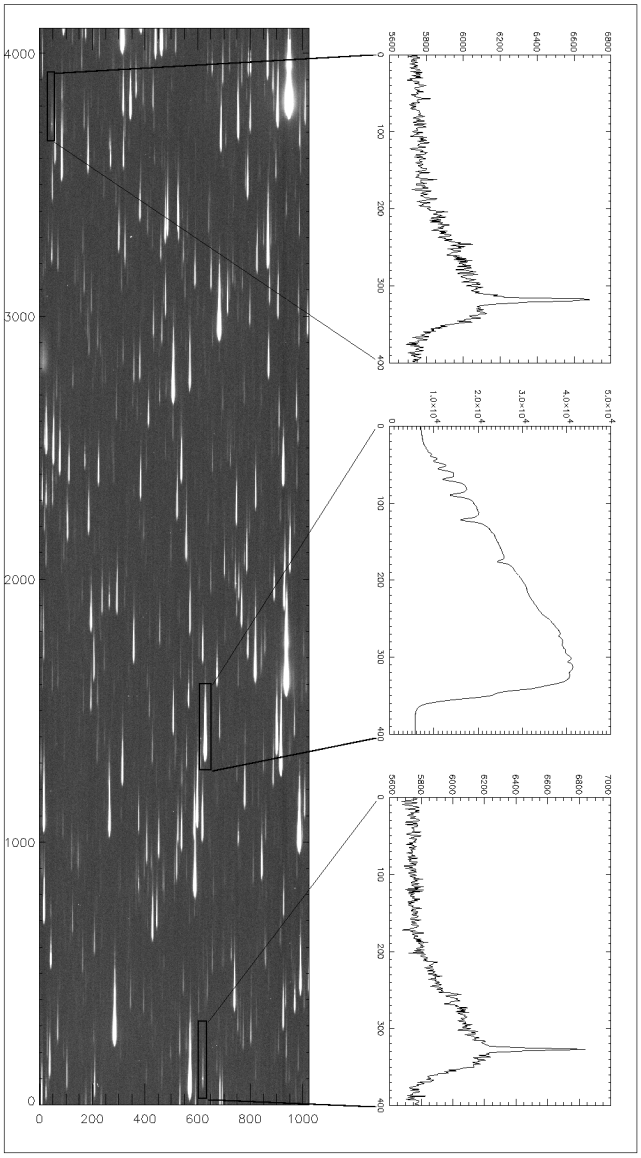

FIG. 7.—A small portion of an objective prism drift-scan image from one CCD is shown. The prism is used to obtain spectra for all objects in the field simultaneously. The dispersion is set parallel to the scan direction to avoid clipping the spectra (which are typically ≈200 pixels in length). Examples of some extracted spectra are shown on the right.

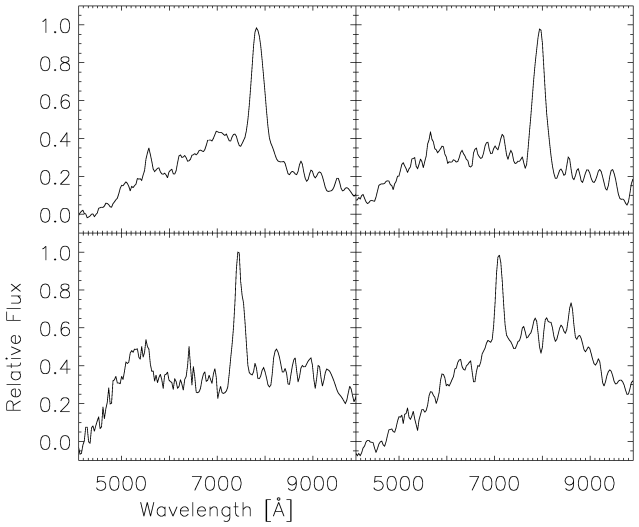

FIG. 8.—Extracted prism spectra are shown with a preliminary wavelength calibration (currently accurate to ≲1%). The top left spectrum is EXO 1134.5+0156, a low-redshift QSO ($z = 0.2$, mag = 17.4), detected with a high signal-to-noise ratio in data from one CCD (i.e., not co-added). The other spectra are previously unknown emission-line objects (mag ≈ 17.5). The detection limit for strong emission-line objects in four CCD co-added data with the hot mirror to reduce the sky background level is $m_B \lesssim 19$. Multiple overlapping scans will improve the detection limit.

With respect to the upcoming Sloan Digital Sky Survey (SDSS; Gunn & Knapp 1993), QUEST will be complementary. For example, the two surveys only partially overlap on the sky (the SDSS is focusing on the northern Galactic cap), and QUEST is obtaining slitless spectroscopy of all objects in the survey region in contrast to the (higher quality) multifiber spectroscopy that SDSS will obtain for targets selected by photometry. The objective prism data has insufficient resolution (and depth) to undertake a next-generation galaxy redshift survey (a primary goal of the SDSS) but will be ideal for identifying emission line sources and other rare (and serendipitous) classes of objects even if they populate an unexpected or dense region of the color diagram. The SDSS will provide a more sensitive and general purpose survey, while QUEST, with a much smaller membership (and budget), will be more open to special purpose projects such as a planned wide-angle variability survey.

5. CONCLUSION

We described an inexpensive, distributed data acquisition system based on commodity PC hardware that reliably manages 30 GB of drift-scan image data per night. The "front end" runs the Linux operating system, providing interactive analysis tools and a graphical interface for system control, while the "back end" (which handles the data flow) runs the QNX operating system, chosen for its real-time performance, network-transparent process communication facilities, and ease of hardware access. Fast data analysis and compaction run concurrently with the data acquisition, providing the observer with data quality feedback and a hierarchy of data subsets that can be optionally directed to permanent storage. The lossless compression algorithm presented makes possible real-time data compression using modest computing resources.

The data system is modular and scalable, so that updating the system for our planned second-generation camera (with 96 CCDs, each with 4096 × 1024 7.5 μm pixels for an aggregate data rate increase of a factor of 6) should not be a major burden. In particular, the hardware dependent tasks are well isolated from the analysis and user interface software, and the client-server model of distributing the workload adapts well to a

modified network configuration (with more processors, for example). In addition, network-distributed applications are likely to thrive in the near future with the emergence of several high performance networking options such as Gigabit Ethernet.

Although we could simply replicate components for the second-generation camera (because the system is roughly modular with respect to rows of CCDs), we plan to make several changes. First, the computer hardware will be upgraded. Tentative plans call for multiple 100 Megabit (or possibly Gigabit) Ethernet networks of 17 Pentium II processors, with SCSI hard disks and multiple DLT drives. Second, the CCD controller interface cards will be modified to allow one computer to control more than four CCDs.

The groups involved in QUEST are Yale University (Physics and Astronomy Departments), Centro de Investigationes de Astronomía (CIDA), Indiana University (Physics and Astronomy Departments), and Universidad de los Andes (ULA). The representatives are (one from each group plus an independent member) Charles Baltay (Yale Physics), Sabatino Sofia (Yale Astronomy), Jim Musser (Indiana University), Gustavo Bruzual (CIDA), Patricia Rosenzweig (ULA), and Gus Oemler. Yale Physics designed and constructed the camera. Yale Astronomy produced the online data system. Indiana University designed and constructed the "front-end" readout electronics. CIDA is providing ≥50% of the Schmidt telescope time and technical support of the camera and online systems in Venezuela under the supervision of Gerardo Sanchez. ULA is providing observers. All groups provide scientific leadership.

We would like to thank S. Shectman for advice on drift scanning and designing the data acquisition system and S. Eisenstat for helpful discussions about data compression.

We acknowledge a grant from Intel providing the online computer system for the second-generation camera (17 Pentium II processors and peripherals) and an educational discount from QNX Software Systems, Ltd., in purchasing the QNX operating system.

REFERENCES

Beard, S. M., MacGillivray, H. T., & Thanisch, P. F. 1990, MNRAS, 247, 311
Bertin, E., & Arnouts, S. 1996, A&AS, 117, 393
Da Costa, G. S. 1992, in ASP Conf. Ser. 23, Astronomical CCD Observing and Reduction Techniques, ed. S. Howell (San Francisco: ASP), 90
Gunn, J. E., & Knapp, G. R. 1993, in ASP Conf. Ser. 43, Sky Surveys: Protostars to Protogalaxies, ed. B. T. Soifer (San Francisco: ASP), 267
Habibi, A. 1971, IEEE Trans. Commun. Technol., 19, 948
Huang, L., & Bijaoui, A. 1991, Exp. Astron., 1, 311
Huffman, D. A. 1952, Proc. IRE, 40, 1098
Irwin, M. J. 1985, MNRAS, 214, 575
Jarvis, J. F., & Tyson, J. A. 1981, AJ, 86, 476
Kron, R. 1995, PASP, 107, 766
McGraw, J. T., Cawson, M. G. M., & Keane, M. J. 1986, Proc. SPIE, 627, 60
Murtagh, F., & Warmels, R. H. 1989, Proc. First ESO/ST-ECF Data Analysis Workshop, ed. P. J. Grosbøl, F. Murtagh, & R. H. Warmels (Garching: ESO)
Netravali, A. N., & Limb, J. O. 1980, Proc. IEEE, 68, 366
Press, W. H. 1992, in ASP Conf. Ser. 25, Astronomical Data Analysis, Software and Systems I, ed. D. Worral, C. Biemesderfer, & J. Barnes (San Francisco: ASP), 3
Rabbani, M., & Jones, P. W. 1991, Digital Image Compression Techniques (Bellingham: SPIE)
Rice, R. F. 1991, JPL Publ. 91-3 (Pasadena: JPL)
Rice, R. F., & Plaunt, J. R. 1971, IEEE Trans. Commun. Technol., 19, 889
Sayood, K. 1996, Introduction to Data Compression (San Francisco: Morgan Kaufmann)
Snyder, J. 1998, Proc. SPIE, 3355, in press
Stetson, P. B. 1990, PASP, 102, 932
Stobie, R. S. 1980, Proc. SPIE, 264, 208
Valdes, F. G., Campusano, L. E., Velasquez, J. D., & Stetson, P. B. 1995, PASP, 107, 1119
Veran, J. P., & Wright, J. R. 1994, in ASP Conf. Ser. 61, Astronomical Data Analysis, Software and Systems III, ed. D. R. Crabtree, R. J. Hanisch, & J. Barnes (San Francisco: ASP), 519
Weir, N., Fayyad, U. M., Djorgovski, S. G., & Roden, J. 1995, PASP, 107, 1243
Welch, B. 1995, Practical Programming in Tcl and Tk (Upper Saddle River: Prentice Hall)
White, R. L., & Percival, J. W. 1994, Proc. SPIE, 2199, 703
Witten, I. H., Neal, R. M., & Cleary, J. G. 1987, Commun. Assn. Comput. Mach., 30, 520
Yeh, P.-S., Rice, R. F., & Miller, W. 1991, JPL Publ. 91-2 (Pasadena: JPL)
Zaritsky, D., Shectman, S. A., & Bredthauer, G. 1996, PASP, 108, 104