Durham adaptive optics real-time controller

Alastair Basden,* Deli Geng, Richard Myers, and Eddy Younger

Department of Physics, Durham University, South Road, Durham DH1 3LE, UK

*Corresponding author: a.g.basden@durham.ac.uk

Received 3 May 2010; revised 6 October 2010; accepted 13 October 2010;
posted 15 October 2010 (Doc. ID 127850); published 9 November 2010

The Durham adaptive optics (AO) real-time controller was initially a proof of concept design for a generic AO control system. It has since been developed into a modern and powerful central-processing-unit-based real-time control system, capable of using hardware acceleration (including field programmable gate arrays and graphical processing units), based primarily around commercial off-the-shelf hardware. It is powerful enough to be used as the real-time controller for all currently planned 8 m class telescope AO systems. Here we give details of this controller and the concepts behind it, and report on performance, including latency and jitter, which is less than 10 μs for small AO systems. © 2010 Optical Society of America

OCIS codes: 010.1080, 110.1080.

1. Introduction

Adaptive optics (AO) is a technology widely used in optical and infrared astronomy, and almost all large current and planned science telescopes have an AO system. A large number of results have been obtained using AO systems that would otherwise be impossible for seeing-limited (uncorrected) observations [1,2]. New AO techniques are being studied for novel applications, such as wide-field high resolution imaging [3] and extrasolar planet finding [4].

When starlight passes through the Earth's atmosphere, random perturbations are introduced that distort the wavefronts from the astronomical source in a time varying fashion [5]. It is then no longer possible to form a diffraction limited image from these distorted wavefronts, and the effective resolution of a telescope is reduced. By sensing the form of the wavefront using a wavefront sensor (WFS) as described by Roddier [6], and then rapidly applying corrective measures to one or more deformable mirrors (DMs), it is possible to compensate for some of the perturbations and, hence, improve the image quality and resolution of the telescope. The WFS and DM together form part of an AO system.

A real-time control system is used to interpret the WFS signals and compute the commands that are to be sent to the DM [7–10]. This is a computationally intensive task, as new DM commands are computed from WFS signals typically 1000 times per second, and the shape of the DM must be adjusted in real time, before the atmospheric turbulence has changed significantly, requiring a control system that is able to operate with minimal latency. It has long been thought that central processing units (CPUs) are not suitable for real-time control of AO systems because of poor performance and large jitter. Previous systems have typically been comprised of complex arrangements of digital signal processors (DSPs) and field programmable gate arrays (FPGAs), requiring extensive development time and quick obsolescence due to the fast changing nature of new hardware. However, this is no longer the case, and here we present a modern CPU-based solution.

The Durham AO real-time controller (DARC) is a highly configurable AO control platform designed for the control of current and as-yet-unknown AO systems. It has been developed from a proof of concept system into a capable controller based around a modern architecture, running on commercial off-the-shelf hardware. It is primarily a CPU-based system, with the ability to use additional hardware acceleration where available. The reasons for developing it were

0003-6935/10/326354-10$15.00/0

twofold: to have a powerful system capable of controlling high-order laboratory experiments at Durham, and to have a control system capable of controlling a multiple laser guide star (LGS) multiobject AO (MOAO) on-sky demonstrator instrument [11] on the William Herschel Telescope (WHT) from 2010 onward, giving on-sky usage of the control system. The technology demonstrated in this experiment is likely to impact the design of future instruments.

Here, we give an overview of the DARC in Section 2, including key components, algorithms implemented, and hardware used. In Section 3 we describe the performance of the system, and conclusions are made in Section 4.

2. DARC Overview

The DARC is comprised of several key components: The real-time control pipeline (RTCP), a control interface, a diagnostic system, a graphical and scripting interface, and background tasks. The RTCP does the bulk of the work, taking WFS camera data and computing the control vectors to be sent to DMs. The control interface is responsible for allowing the user to update and control the RTCP, for example, changing reference images. The diagnostic system is an optional component recommended for large systems and is responsible for taking output produced by the RTCP, logging it, and distributing it to clients as requested. The graphical and scripting interfaces provide easy ways for a user to alter the state of the system via the control interface. An overview of the components of the DARC is shown in Fig. 1.

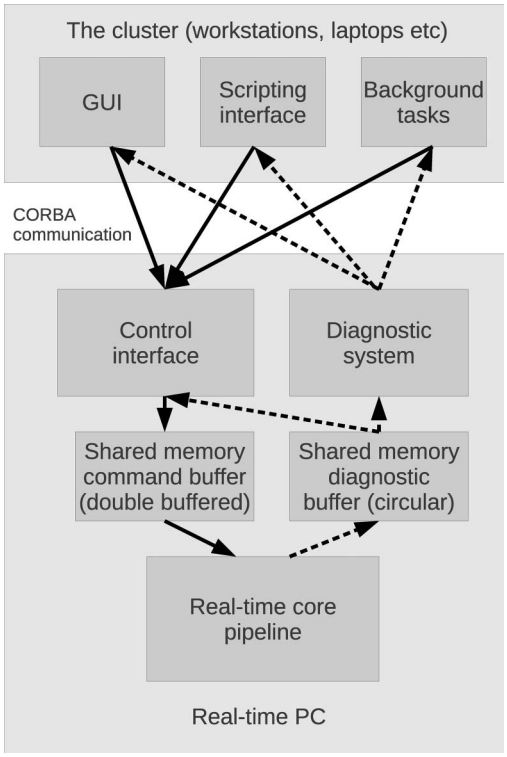

Fig. 1. Overview of the DARC system showing the key components and interactions between them: solid lines, control flow; dashed lines, data (telemetry) flow.

A. Real-Time Control Pipeline

The RTCP is a software executable running on real-time (or standard) Linux. It is multithreaded, with the number of threads being user controllable. The user specifies the priorities of each of these threads and can restrict them to certain processors (the CPU affinity), and make such changes while the RTCP is running. When running on multicore machines (currently, up to 16 cores are readily available off the shelf), this gives the user flexibility to optimize a given AO system for given computational hardware. The RTCP also has the ability to make use of the processing power found in graphical processing units (GPUs) for wavefront reconstruction, if suitable hardware is available.

Diagnostic data are written to shared memory circular buffers and includes raw and calibrated pixels, slope measurements, wavefront estimates, DM commands, and flux measurements. The RTCP cannot be delayed by the writing of diagnostic data, meaning that performance is not affected by network glitches or errors in diagnostic reading subsystems.

1. *Interfaces to the RTCP*

The data input to the RTCP can either be in the form of raw camera images, calibrated camera images, or wavefront slope measurements. An optional FPGA front end has been developed at Durham for the ESO SPARTA AO system [12], called the wavefront processing unit (WPU), which calibrates WFS images and computes wavefront slopes. This WPU front end can optionally be used with the RTCP and has very low (submicrosecond) latency and virtually no jitter, being deterministic. It should be noted that this WPU front end performs weighted center-of-gravity slope measurements; if other slope measurement algorithms are required, they must be performed by the RTCP.

The interface for handling data input into the RTCP is based on shared object libraries, which can be changed (or indeed, written) while the RTCP is running. Such interfaces currently include a serial front panel data port (sFPDP) [13] interface [used with low-light-level electron multiplying CCD (EMCCD) cameras at Durham], a gigabit Ethernet interface (used with Pulnix cameras), a socket interface, a universal serial bus interface (used with Xenics IR cameras), and a file interface for testing. A library for virtually any interface could be written as required by a user. The user has the ability to swap between these interfaces as required without stopping the RTCP, useful, for example, for changing from a physical camera into a replay mode. This is also the case for the interface responsible for sending the DM commands, and currently implemented interfaces include one for sending actuator demands to a figure sensor over sFPDP, one to control the DM combination that we have at Durham directly, and one for sending actuator demands over a socket.

Such flexibility means that the DARC can be used with practically any AO system without code changes to the core real-time system (and without the subsequent debugging that this would entail), with the user writing input and output libraries appropriate for their system.

There is also an interface for handling wavefront reconstruction, allowing the wavefront reconstruction algorithm to be changed (and even developed) on the fly without halting the RTCP. Currently implemented reconstruction interfaces include a Kalman filter, a standard matrix–vector multiplication integrator, and a matrix–vector-based algorithm for open-loop wavefront control. Additionally, a GPU-based matrix–vector multiplication interface is also available and will be presented in a separate paper. We also intend to implement iterative and Fourier-based reconstruction algorithms using this interface.

2. *RTCP Processing Strategy*

The processing of data is carried out using what we term a horizontal processing strategy, where many threads perform multiple algorithms, with each thread performing the same operations as other threads. Data are processed by the RTCP on a sub-aperture-by-sub-aperture basis, with a thread requesting work, and being assigned one (or more) sub-apertures to process (as soon as enough pixels have arrived for this sub-aperture) up to the stage of computing the influence that this sub-aperture has on the DM (if possible, dependent on a reconstruction algorithm) before returning and requesting another sub-aperture to process. This allows the processing to commence as soon as enough camera pixels have arrived for a given sub-aperture, before the whole camera frame is ready, greatly reducing the time between the last camera pixel being read out and the DM commands being ready (the latency), an important parameter for an AO control system. Once the last sub-aperture has been processed, the partial DM commands computed by each thread are amalgamated and post-processed (including clipping) by a post-processing thread and sent to the DM. While this post-processing is being carried out, the other threads are able to start processing the next WFS camera frame if it is available. All sub-apertures of a given frame must have been processed before the next frame is commenced.

A more conventional processing strategy is the vertical processing strategy, where discrete tasks are assigned to different threads, or at least, performed one at a time rather than being broken up into pieces (e.g., a thread or process for image calibration, a thread for slope computation, and a thread for wavefront reconstruction). Here, synchronization between stages occurs either after each stage has completed (greatly increasing latency since all stages must wait until all the pixel data has arrived, and then until all previous stages have completed), or after each sub-aperture has completed its stage. This, however, greatly increases the amount of synchronization required between stages (using, for example, Mutexes or locks), which, in turn, adds to the latency. Figure 2 demonstrates these strategies.

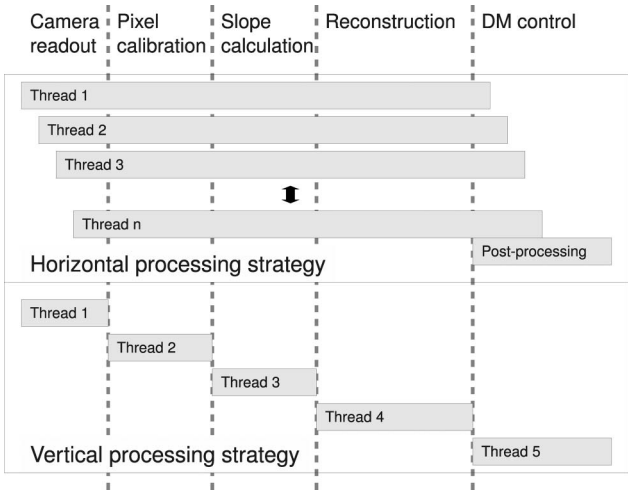

Fig. 2. Demonstration of the horizontal and vertical processing strategies. With a horizontal strategy, all the threads perform the same operations, allowing the processing load to be balanced between available processors. With a vertical strategy, the work performed by threads is unequal, load balancing becomes harder, and greater latency is introduced by the need to communicate between each stage.

A horizontal processing strategy allows far better CPU load optimization than a vertical processing strategy since each thread is assigned the same amount of work, which will allow a higher CPU utilization while there is processing to be carried out, reducing the total time taken. Additionally, processing of a sub-aperture is carried out from start to finish by one thread, reducing the need for thread synchronization that a vertical processing strategy would require (where each processing stage needs to be synchronized with the others).

The ability to compute partial DM commands (i.e., the influence of a given sub-aperture on the final DM command vector) is dependent on the wavefront reconstruction algorithm used. With most iterative and Fourier-based wavefront reconstruction algorithms, the computation of partial DM commands is not possible. These algorithms require all the slope measurements to be present before the computation of the DM command vector commences, meaning that the wavefront reconstruction algorithm only starts after all slope measurements have been computed. This has the disadvantage that a greater amount of post processing is required after all the WFS pixels have been received by the RTCP, and so can result in a greater latency than algorithms that allow the computation of partial DM commands.

With a matrix–vector implementation (where the DM command is updated with the dot product of a control matrix with a vector of wavefront slope measurements), the computation of partial DM commands is possible, since the column of the matrix corresponding to a given sub-aperture is multiplied

by the slope measured from this sub-aperture (a scalar–vector multiplication) to give the influence that this sub-aperture has on the final DM command. This computation can commence as soon as the slope for this sub-aperture has been computed, which, in turn, is computed as soon as enough pixels for this sub-aperture have arrived from the WFS. Camera readout is relatively slow: a fast camera may take $0.1\,\mu$s to read out each pixel, so the time to read out a sub-aperture can be large compared with the time to compute the wavefront slope of this sub-aperture. The final DM command is then computed by summing together all the partial DM commands.

The reconstructor interface of the RTCP is flexible enough to allow both algorithms that compute partial DM commands and algorithms that cannot compute partial DM commands to be used, allowing different reconstruction algorithms to be plugged in and out of the RTCP.

One drawback of the horizontal processing strategy is that it is not as easy to separate processing tasks out onto different hardware as it would be with a vertical processing strategy, for example, by using one PC to perform image calibration, another PC to perform wavefront reconstruction, and another to perform mirror control. However, feedback between reconstruction and control algorithms is often a requirement, making these unlikely to be separated. Additionally, image calibration is optionally performed by an FPGA front end, such as the WPU developed at Durham, making a separate calibration processing task unnecessary. Furthermore, as we demonstrate in this paper, by using a horizontal processing system like the DARC on current top-end processors, it is unnecessary to separate these tasks for likely AO systems on current 8 m class telescopes, since modern processors are powerful enough to perform the required task.

3. *Algorithms Overview*

Standard image processing algorithms, such as background and noise removal, flat fielding, thresholding, and pixel weighting, are carried out by the RTCP to calibrate the raw WFS images (when not using the WPU front end).

The wavefront slopes across each sub-aperture are computed using a center-of-gravity (standard or weighted) centroiding algorithm or a correlation centroiding algorithm [14].

The RTCP uses one of a number of wavefront reconstruction algorithms to compute the commands to be sent to the DMs. Least-squares or minimum variance reconstruction is implemented, with several control laws, including Kalman filtering [15], being available. The RTCP can be used in closed-loop (where the WFSs are sensitive to changes on the DM) or open-loop (where the WFSs are placed before the DM in the optical path, and so do not measure changes applied to it) operation.

Slope linearization, essential for getting the best performance with open-loop systems, is optional. Here, a lookup table between computed wavefront slope (nonlinear due to the pixelated nature of the detectors) and actual wavefront slope is provided, and so computed slopes are linearized using this lookup table. A separate lookup table is provided for each sub-aperture and WFS, allowing for different nonlinearities to be compensated, for example, with elongated spots due to a LGS. The linearization calibration should be performed using a tip–tilt mirror, gradually tilting the wavefront across the WFS, and recording slope measurements as this is carried out. The nonlinearity is usually small enough that closed-loop systems are not adversely affected since slope measurements are minimized by the DM, and arises from the discrete pixelated nature of the detectors and is also caused by optical effects (imperfect lenslets, for example).

An adaptive windowing algorithm is also available with two operating modes. With a global mode, the mean spot motion is tracked and the pixels assigned to each sub-aperture follow this mean spot motion. With a per-sub-aperture adaptive windowing mode, the pixels assigned to each sub-aperture track the motion of the spot in this sub-aperture rather than being fixed. Adaptive windowing is a useful feature for open-loop control, where spot motions may be large, especially for LGSs.

4. *Optional Figure Sensing Operation*

The RTCP may optionally accept an asynchronous DM command input in addition to the WFS input, enabling it to be used as a figure sensor alongside an open-loop AO system, allowing a nonlinear DM to be observed and controlled so that the actual shape can be tweaked until it matches the requested shape. This allows for open-loop operation with a nonlinear DM (WFSs do not measure the shapes applied to the DM, so do not know its true shape). In this case, one instance of the RTCP would compute desired DM commands (assuming a linear DM) using the open-loop (on-sky) WFS measurements. This RTCP can assume that it has a perfect, linear DM attached. These commands are then sent to a second instance of the RTCP (on the same, or different hardware), which would have image input from a figure sensing WFS that is in closed-loop control with the nonlinear physical DM, operating at a higher frame rate than the on-sky WFSs. This second RTCP rapidly measures the physical DM surface and adjusts the commands sent to the DM until the DM surface has reached the shape that the first RTCP has requested, as shown in Fig. 3. It would do this by reading the figure sensor WFS at a rate faster than the main open-loop RTCP. The wavefront reconstructed from the figure sensor WFS data then provides the current actual DM shape (typically using a least-squares integrator control law), which is compared with the desired DM shape, and the difference

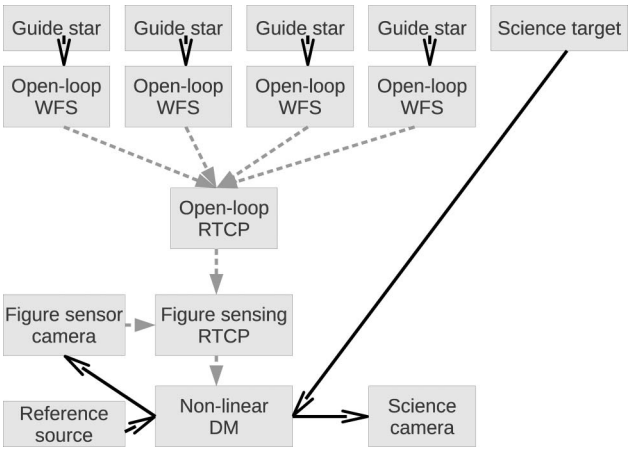

Fig. 3. Schematic diagram of a typical configuration for an open-loop AO system used with a figure sensor. Solid black arrows with unfilled arrow heads, optical paths; dashed gray arrows with filled arrow heads, electrical paths. The open-loop RTCP receives wavefront data from open-loop WFSs. The mirror demands (an ideal mirror is assumed) are passed to the figure sensing RTCP, which combines these with the measured shape of the mirror (obtained using the figure sensor WFS), to set the mirror to the desired shape.

applied to the DM. The figure sensing control loop typically runs at between 2 and 5 times faster than the main open-loop RTCP. This allows open-loop operation with DMs that have poor characteristics, such as large hysteresis and a nonlinear response. Figure 3 shows a typical configuration for an open-loop AO system with a figure sensor. It should be noted that the on-sky, open-loop RTCP does not receive any feedback from the DM or from the figure sensing RTCP.

B. Control Interface

The control interface runs on the same hardware as the RTCP and accesses shared memory to update the RTCP parameters. The parameters are double buffered and, so, the control interface does not delay the RTCP while parameters are updated. The control interface exposes a common object request broker architecture (CORBA) object [16], which is then used by clients to perform any required operations. CORBA is a standard architecture and infrastructure that allow programs from almost any computer, operating system, and programming language to interoperate.

The control interface is controlled remotely using scripts or a graphical interface, to perform operations such as computing system interaction (response) matrices, calculating and setting background maps, and fine-tuning algorithms. It can also be used to obtain diagnostic data.

C. Diagnostic System

The DARC implements a philosophy of separate diagnostic streams for all diagnostic data (images, slope measurements, etc.). These streams are turned on or off as required, depending on demand and the ability of the system to cope (considering network bandwidth, processing power, etc.). Additionally, streams can be independently decimated with data for every Nth frame being sent. The DARC offers two different sources for diagnostic streams. For most systems, a diagnostic server will be implemented, typically running on different computational hardware. This server receives diagnostic streams from the RTCP via a process that takes data from the RTCP shared memory circular buffers and then distributes the data to clients as requested, removing the processing load from the RTCP machine. For small scale AO systems and laboratory tests, diagnostic data can be sent using the control interface and avoiding the complexity of the diagnostic server, allowing the control system to be set up quickly with minimal hardware in a laboratory environment.

Clients subscribe to diagnostic streams using a CORBA interface and are then sent the data when it is ready. Diagnostic streams can be switched on or off, and decimated in three separate places: At the RTCP, in the diagnostic server, and on a client-by-client basis. This allows fine control of how much data should be sent, for example, every 10 frames of raw pixels for logging (by the diagnostic server), but every 20 frames of raw pixels sent to client A, and every 25 frames of raw pixels send to client B. This ability allows the performance of the DARC to be fine-tuned to available hardware for a given system size.

D. Graphical Interface and Scripting

A graphical interface is used to control the real-time system remotely, display current status and parameters, and view diagnostic streams. This tool is generic and is used with systems with any number of WFSs and DMs. More than one instance of the graphical interface may be run simultaneously. This tool uses the CORBA objects of the control interface and the diagnostic system to realize the necessary control.

Scripting is also carried out in a similar way, and a user script may be written in any language for which CORBA is available. There are also some command line tools for performing simple operations with the DARC (for example, setting parameters and obtaining diagnostic data).

E. Background Tasks

Background tasks, such as control matrix optimization using turbulence profiling data and optimization of system performance, are to be performed remotely, using data from the diagnostic system and updating the RTCP using the control interface. Such operations are dependent on AO system type and configuration and so should be developed for each system on which the DARC is deployed, rather than being part of the DARC, which is designed to be generic.

3. System Performance

When evaluating the system performance of an AO control system, there are a number of performance

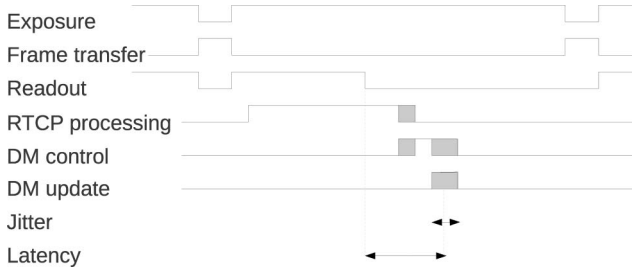

Fig. 4. Schematic timing diagram for an AO system with a frame transfer CCD. Jitter in signals is shown by the gray areas (where time of transition may vary), which is introduced by the nondeterministic processes of RTCP processing and DM control.

parameters that should be measured. Figure 4 shows a schematic timing diagram for an AO system, assuming a frame transfer CCD. Here, RTCP processing commences once enough pixels have been received and is concurrent with CCD readout. Latency is defined as the time from end of readout to average DM update time, while jitter is the variation in DM update time, the stability of the system. Low jitter is crucial for an AO system, as there is little point having a very low average latency (e.g., $1\,\mu$s) if it sometimes takes a long time to process a frame (e.g., 1 s) because the scientific image quality would drop during this delay. The latency is an important parameter because it affects the mean age of the WFS image data at the time when the DM is updated, equal to $t_a = t_e/2 + t_r + t_l$, where t_e is exposure time, t_r is readout time, t_l is the latency, and t_a is the mean data age. A larger mean data age means a less accurate DM correction since the atmosphere will have had greater time to evolve, which will result in reduced performance, as the DM shape will no longer be matched to the evolved atmospheric disturbance. Minimal latency is essential for best performance. The maximum RTCP frame rate is the maximum rate at which the system can process camera images (although usually this will be limited by the camera). It should be noted that, in a RTCP with a vertical processing strategy (see Fig. 2), the latency can be long even when the maximum frame rate is high, since multiple frames may be at different stages of processing simultaneously. For example, frame one could be at the processing stage where actuator commands are being produced, while frame two is at the slope measurement stage, frame three is at the image calibration stage, and frame four is being read from the WFS. Conversely, with a horizontal processing strategy, such as that we have implemented with the DARC, the latency will be approximately the inverse of the maximum frame rate (assuming the case where this is not limited by cameras), since the processing of one frame has to complete before processing of the next frame commences (post-processing may add some latency). Hence, by reducing the latency of the RTCP, the maximum possible frame rate is increased. This then equates into hardware savings because less hardware can be used to run a given AO system at the required frame rate. Less hardware also provides a system that is simpler to manage, and more reliable (fewer components to break).

We have tested the performance of the DARC using a dual quad-core Intel Xeon based computer and a quad quad-core AMD-Opteron-based computer (both 2008 specification with 5500 series Xeon processors at 2.26 GHz and 8378 series Opteron processors at 2.4 GHz respectively) and both give similar performance for small and medium-sized systems. It should be noted that recent Intel chip sets perform a housekeeping task approximately once per second, which needs to be switched off (using a Linux kernel patch) to avoid occasional jitter when these tasks perform. The RTCP results presented here use a standard Linux (non-real-time) kernel on the quad quad-core Opteron-based system. The RTCP will also run with a real-time kernel without modification, further reducing jitter, although we do not present such results here because the camera interface drivers for which we are making these measurements are not compatible with a real-time kernel.

A. Oscilloscope Latency and Jitter Measurements

We have measured the latency and jitter of the DARC using a digital oscilloscope triggered on the camera exposure trigger (start of readout), looking at the variation of when changes in the voltages sent to the DM drive electronics occur. Similar measurements were also recorded by the RTCP (the time taken to compute a frame) and compared. The oscilloscope is set with infinite persistence, meaning that newly triggered signals are overlaid on existing signals, allowing a record of signal transition times to build up, giving a measure of the variation over which these transitions occur. We transition the DM actuators from low to high or vice versa every camera frame and so the DM signal is at half of the frequency of the camera signal (high on one camera frame, low the next, and so on *ad infinitum*). Combined with the infinite persistence, this means that the DM transitions from low to high and from high to low are shown overlain. The mean distance of the set of actuator transitions from the camera trigger pulse gives the mean latency of the system once the camera readout time has been subtracted. The width of the set of actuator transitions gives a measure of the jitter. The latency is assumed to have a Gaussian distribution so the oscilloscope trace built up over a long period of time (infinite signal persistence) will approximately display the mean latency $\pm 3\sigma$, where σ is the root-mean-square (RMS) jitter. Therefore, we record the RMS jitter as 1/6 of the measured long time average actuator transition width.

1. *Single Camera System*

Our bench system used for the tests presented here is comprised of a single 128×128 pixel EMCCD camera with 7×7 sub-apertures and a 52 actuator

DM with a weighted center-of-gravity slope calculation algorithm, a least-squares matrix–vector-based wavefront reconstruction algorithm, and an integrator control law. It should be noted that, in this system, the RTCP sends actuator demands to a separate computer running a figure sensor, which then places these demands onto the mirror (the DM control time in Fig. 4). In these tests, actuator values are just passed unaltered to the DM; however, latency and jitter associated with an additional sFPDP transfer and additional PC are included in these results. A figure sensor should be an integral part of any open-loop AO system (such as MOAO) design when using DMs with a nonlinear response, being the only way we have of knowing what shape the DM is actually taking, and so it was included in these performance tests.

The mean time between start of camera readout (trigger pulse) and setting of DM actuators is measured using the oscilloscope to be about 620 μs when the total camera readout time is 400 μs (the maximum pixel rate limited by the data transfer rate to the RTCP). A RMS jitter of 8 μs is measured (a peak-to-valley jitter of 50 μs), giving a latency measurement of $220 \pm 8\,\mu$s. After recording actuator transitions for about 1 min, we typically see that there are three outlying DM transitions during this time, the largest of which adds an additional 65 μs latency (which may be due to the figure sensor). A 15 min recording over a larger oscilloscope time base to check for large occasional latency variations shows no actuator transitions away from the main set. Therefore, large infrequent changes in latency are not occurring. It should be noted that this is not an oscilloscope selection effect, and that, if we manually perform a high priority intensive task on the computer (running at a higher priority than the RTCP), then single lines become visible at larger latencies.

When the camera is running as typically used on-sky (i.e., with good image quality), it has a readout time of about 2000 μs, so the last pixel arrives at the RTCP about 2000 μs after the trigger pulse. By using the oscilloscope to measure the actuator transition times relative to the trigger pulse, we find that the DM commands are sent 2100 μs after the trigger pulse, meaning there are about 100 μs between the last camera pixel being received by the RTCP and the DM actuators being set. The latency in this case is, therefore, $100 \pm 6\,\mu$s since the oscilloscope measured latency has a peak-to-valley difference of 36 μs. This is reduced compared with the previous measurement because the RTCP has more time for calculation while waiting for camera pixels to arrive.

We have measured the actuator response time of the RTCP by operating the system in closed-loop mode, and then applying an offset voltage to one actuator, while recording the commands sent to the DM. Figure 5(a) shows the AO system response time while operating at different integrator gains. In this figure, an offset voltage is applied at iteration 50, and the RTCP subsequently corrects this offset. With a unity gain (with which the system can close the loop

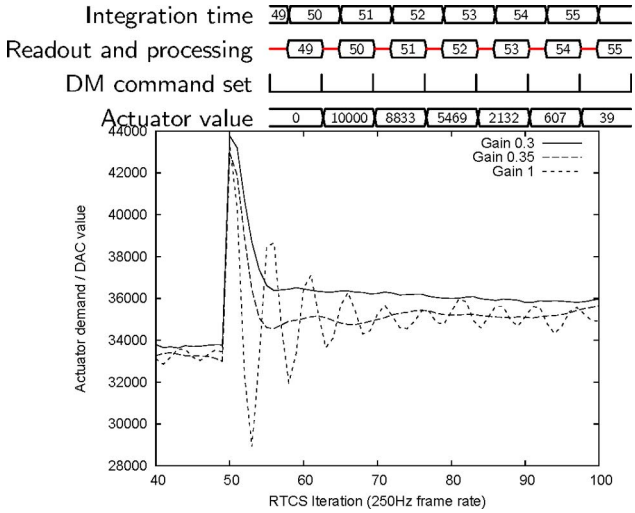

Fig. 5. (Color online) (a) Actuator response to an applied perturbation as a function of time. The plot shows the command applied to the DM actuator, after a step change of 10,000 DAC values was inserted at iteration 50. (b) Schematic timing diagram demonstrating the expected number of iterations before the perturbation has been corrected, for a gain of 0.35, with readout and processing taking two-thirds of the exposure time.

stably without blowing up, demonstrating the low latency performance), a fast actuator response is seen with correction being applied in only two iterations. Overshoot, or ringing, occurs due to the necessary delay between an image being integrated on the detector and readout (taking finite time), and the shape being applied on the DM. With a gain of 0.3, the system is under-damped, while a gain of 0.35 shows critical damping, taking about four iterations for the loop compensation to be fully applied, as expected from analysis of the timing diagram in Fig. 5(b). The actuator values given in this timing diagram are calculated by assuming that the RTCP will compute the actuator value to be a time weighted mean of the two states that the actuator is in during the integration time (taking two-thirds of the time in one state, and one-third of the time in the next state). The integrator then multiplies this by the gain, and adds to the previous integrator state. The actuator value is seen to have returned to near zero after only a short number of iterations.

2. *Figure Sensor Latency*

The latency and jitter of the figure sensor PC are included in these measurements, as mentioned previously. We have measured separately the latency and jitter for the figure sensor PC alone and obtain $70 \pm 5\,\mu$s latency with some outliers that double this value (recorded over approximately 30,000 transitions). To make these measurements, a trigger pulse is generated in an FPGA, which then also implements a fake camera interface with 52 pixels (since there are 52 DM actuators), which is passed straight to the figure sensor and interpreted as a DM command vector.

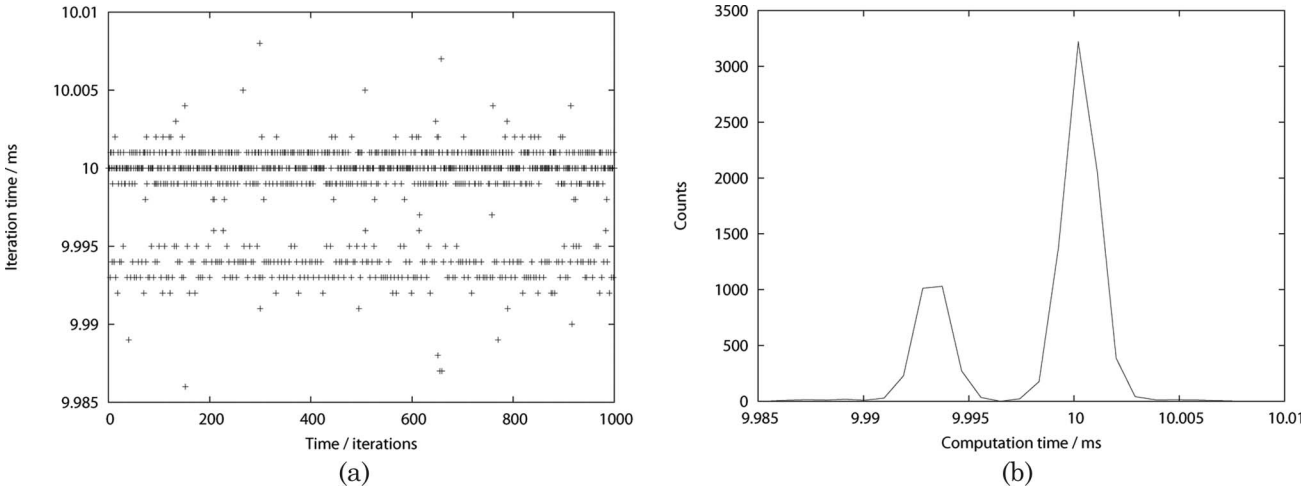

Fig. 6. (a) Frame iteration time generated by the real-time control pipeline over 1000 iterations. (b) Histogram of the frame computation time (10,000 iterations). Camera frequency is 100 Hz (10 ms frame time).

3. *Software Measured Jitter*

The RTCP is able to record the time taken to process frames using the PC hardware clock. Figure 6 shows timings for 10,000 frames, with a mean value of just under 10 ms (the camera frame rate), a worst case measurement of 10 μs above this, and a RMS of 3 μs. This is in good agreement with the oscilloscope estimated value of 8 μs, which also includes the figure sensor jitter of about 5 μs (note, these should be added in quadrature) and is only estimated from the width of the oscilloscope trace. The histogram shown in Fig. 6(b) shows that the latency distribution is non-Gaussian, having a double peak, which would cause an oscilloscope estimated jitter value to be a slight overestimation. Our oscilloscope measurements also give some evidence for this double peak, although this is not conclusive due to the crude nature of this measurement ("transition" or "not transition," but no record of number of transitions at a given time). We are unsure of the reason for this double peak, which may be a result of the software timer resolution. Alternatively, it could be due to scheduling in the Linux kernel or could be a true feature of the RTCP.

B. Other AO System Configurations

We have measured the maximum frame rate that the DARC running on our available hardware is capable of handling for a number of different AO system configurations. Software recorded jitter measurements are given, although, since we do not have suitable cameras for these cases, the oscilloscope measurements are not available.

The cases we consider are shown in Table 1 for a simple scaling of a single-guide-star, single-DM system. In these cases, the frame time is equal to the time taken to process an image and compute the actuator demands, but not to receive the image (since we do not have appropriate cameras) or to send the actuator commands (since we do not have appropriate mirrors).

Measurements are given for cases that include the Shack–Hartmann WFS image calibration and slope calculation time, and also for cases where the calibration and slope calculation is carried out by the WPU front end (which is FPGA based and has about 0.5 μs latency, so is negligible), allowing the RTCP to be used for wavefront reconstruction, DM command vector calculation, and housekeeping tasks.

It should be noted that these measurements are real, made with the RTCP, not extrapolations, and show that the DARC has the ability to operate a classical 32 × 32 sub-aperture AO system at over 500 Hz using a single PC, or to operate a 64 × 64 sub-aperture AO system at over 2 kHz using a single PC and a WPU front end. This would give the performance of a planet-finding instrument on a 8 m class telescope and demonstrates the ability of the DARC when compared with the computational hardware traditionally required by high-order AO systems [12], which involves large numbers of hardware-based

Table 1. Measured Frame Computation Time for Different AO System Configurations

Sub-Apertures	Image Size	Actuators	Frame Time (μs)	With WPU Front End [a] (μs)
8 × 8	128 × 128	52	149 ± 12	20 ± 2
16 × 16	256 × 156	208	519 ± 22	35 ± 3
32 × 32	512 × 512	832	1517 ± 37	83 ± 3
64 × 64	1024 × 1024	3328	9160 ± 85	355 ± 3

[a]The results with WPU front end assume that slope measurements are received from an FPGA and so only wavefront reconstruction is performed using the RTCP.

processing boards, subsystems, and complex communication protocols.

C. On-Sky Tests

The technology demonstrator instrument on the WHT with which the real-time controller is used has three separate phases. At phase A, from September 2010 onward, there are four on-sky WFSs, three of which are used for open-loop tomography (wavefront reconstruction) and do not see the DM, while the fourth is used as a truth sensor behind the DM to measure the corrected wavefront, although it is not used in the reconstruction process. There is a tip–tilt mirror, and a 52 channel DM, neither of which is visible to the on-sky WFSs. Phase B (beginning in 2011) will introduce four LGS WFSs, and phase C introduces an additional, high order (1024 actuator) DM, all of which are to be controlled by the DARC.

We have investigated the RTCP performance for phase A when computing the open-loop DM demands for the three-open-loop WFS configuration, each with 128×128 pixels and 7×7 sub-apertures. A mean RTCP computed latency of $245 \pm 11\,\mu$s for the total processing time without the WPU front end is achieved, well within the required specification of $1000 \pm 100\,\mu$s.

Figure 7 shows the optical bench used for the phase A configuration, with WFSs, truth sensor, and DM visible. A transfer function for this system gives a latency of $800\,\mu$s between the last pixel received and the DM command applied when operating in closed loop, processing wavefront data from all four WFSs, and using the truth sensor measurements to close the loop. This measurement includes the DM response time of about $500\,\mu$s.

Sample (optical) WFS data are shown in Fig. 8(a) from one of the open-loop WFSs imaging a 8.7 Mag_v star. Corresponding truth sensor measurements (for an 11 Mag_v star) when the loop was engaged, taken by the WFS behind the DM (which is tomographically controlled by the open-loop WFSs), are shown in Fig. 8(b). Uncorrected and corrected H-band science images for this setup are shown in Figs. 8(c) and 8(d), with a corresponding Strehl ratio about 26% for the corrected image.

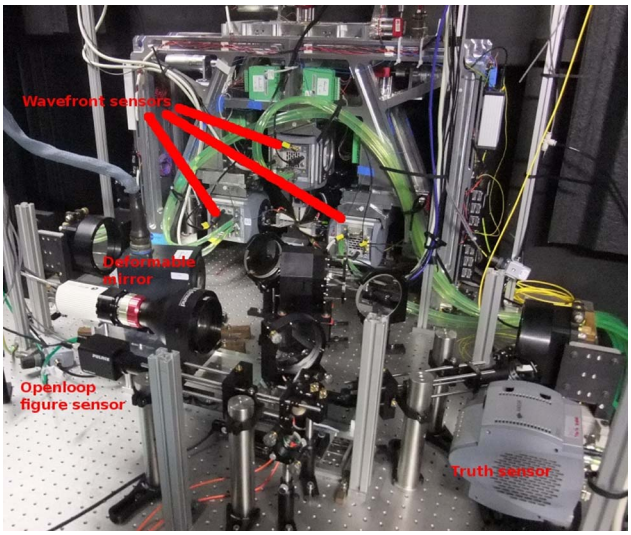

Fig. 7. (Color online) Picture of the technology demonstrator test bench at the WHT, with the AO system components clearly visible.

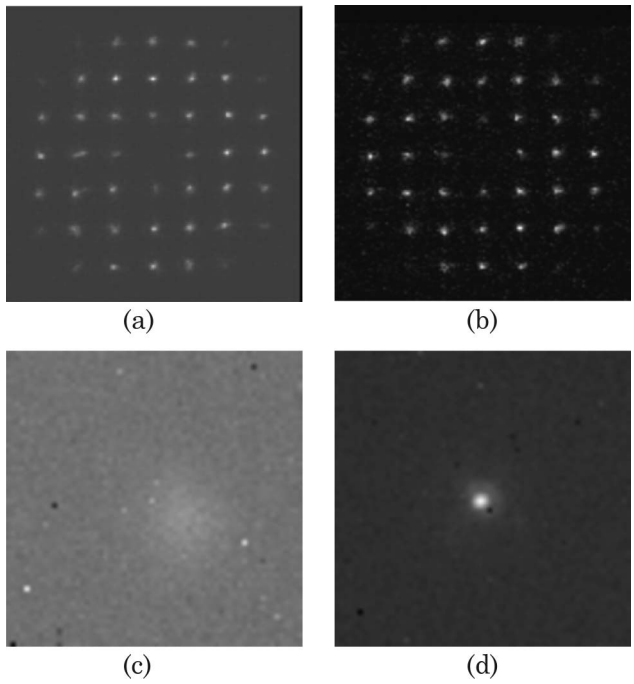

Fig. 8. (a) Sample open-loop WFS measurements. (b) Sample truth sensor measurements. (c) Uncorrected H-band image. (d) MOAO corrected H-band image.

D. Maximum Frame Rates

For the DARC, the maximum frame rates that can be achieved for a given system are the inverse of the latency due to the horizontal processing strategy (assuming a fast enough camera). For example, Table 1 shows that a frame rate greater than 2 kHz is achieved for a 64×64 sub-aperture system when using the WPU front end, which would be suitable for an 8 m class telescope with a planet-finder instrument. It should be noted that there is little point in operating an AO system with a frame time significantly less than that of the DM response time, since, in this case, the DM convergence curve will show errors increasing before the overall error is reduced. However, since some DMs can have an extremely fast response, and since the DARC is designed as a generic AO real-time control system, it is important that the performance of the DARC is maximized.

4. Conclusion

We have given details of the DARC and an overview of some of the features of this system and the algorithms it provides. This is a CPU-based system with an optional FPGA-based WPU front end for performing pixel calibration and slope calculation, and the ability to optionally perform wavefront reconstruction using a GPU. This real-time control system is

generic, not targeted at any particular AO system, and, yet, powerful. The performance of this system has been investigated for a number of different AO system configurations, and the latency and jitter measured. This system meets the requirements of the MOAO technology demonstrator instrument on the WHTelescope. The latency of the RTCP when used with a three-WFS system and a 52 actuator DM (demonstration phase A) is measured to be about 245 μs with a RMS jitter of 11 μs, well within the required specification.

When using an FPGA front end, the latency of the RTCP when used with an extreme AO system with a 64×64 sub-aperture WFS, driving 3328 mirror actuators, is 355 μs with a RMS jitter of 3 μs, allowing the RTCP to exceed a 2 kHz frame rate typically required for most planet-finding instruments on 8–10 m class telescopes.

The low latency and jitter provided by the DARC demonstrate that current CPU technology is suitable as a candidate for use with moderate-sized AO systems, and that expensive and complex custom hardware designs are no longer necessary for such systems. Such low latency is necessary to maximize the performance of AO systems, allowing changes to be applied to DMs before the atmospheric perturbations have had time to evolve significantly.

This work is funded by the Science and Technology Facilities Council (STFC). The authors also thank the team who used the DARC so well on-sky.

References

1. E. Gendron, A. Coustenis, P. Drossart, M. Combes, M. Hirtzig, F. Lacombe, D. Rouan, C. Collin, S. Pau, A.-M. Lagrange, D. Mouillet, P. Rabou, T. Fusco, and G. Zins, "VLT/NACO adaptive optics imaging of Titan," Astron. Astrophys. **417**, L21–L24 (2004).
2. E. Masciadri, R. Mundt, T. Henning, C. Alvarez, and D. Barrado y Navascués, "A search for hot massive extrasolar planets around nearby young stars with the adaptive optics system NACO," Astrophys. J. **625**, 1004–1018 (2005).
3. E. Marchetti, R. Brast, B. Delabre, R. Donaldson, E. Fedrigo, C. Frank, N. N. Hubin, J. Kolb, M. Le Louarn, J. Lizon, S. Oberti, R. Reiss, J. Santos, S. Tordo, R. Ragazzoni, C. Arcidiacono, A. Baruffolo, E. Diolaiti, J. Farinato, and E. Vernet-Viard, "MAD status report," Proc. SPIE **5490**, 236–247 (2004).
4. D. Mouillet, A. M. Lagrange, J.-L. Beuzit, C. Moutou, M. Saisse, M. Ferrari, T. Fusco, and A. Boccaletti, "High contrast imaging from the ground: VLT/planet-finder," Astron. Soc. Pac. Conf. Ser. **321**, 39–46 (2004).
5. V. I. Tatarski, *Wavefront Propagation in a Turbulent Medium* (Dover, 1961).
6. F. Roddier, *Adaptive Optics in Astronomy* (Cambridge U. Press, 1999).
7. R. G. Dekany, J. K. Wallace, G. Brack, B. R. Oppenheimer, and D. Palmer, "Initial test results from the Palomar 200 in. adaptive optics system," Proc. SPIE **3126**, 269–276 (1997).
8. P. L. Wizinowich, D. Le Mignant, A. H. Bouchez, R. D. Campbell, J. C. Y. Chin, A. R. Contos, M. A. van Dam, S. K. Hartman, E. M. Johansson, R. E. Lafon, H. Lewis, P. J. Stomski, D. M. Summers, C. G. Brown, P. M. Danforth, C. E. Max, and D. M. Pennington, "The W. M. Keck Observatory, laser guide star adaptive optics system: overview," Publ. Astron. Soc. Pac. **118**, 297–309 (2006).
9. T. N. Truong, A. H. Bouchez, R. G. Dekany, J. C. Shelton, M. Troy, J. R. Angione, R. S. Burruss, J. L. Cromer, S. R. Guiwits, and J. E. Roberts, "Real-time wavefront control for the PALM-3000 high order adaptive optics system," Proc. SPIE **7015**, 70153I (2008).
10. R. Stuik, R. Bacon, R. Conzelmann, B. Delabre, E. Fedrigo, N. Hubin, M. Le Louarn, and S. Ströbele, "GALACSI: the ground layer adaptive optics system for MUSE," New Astron. Rev. **49**, 618–624 (2006).
11. R. M. Myers, Z. Hubert, T. J. Morris, E. Gendron, N. A. Dipper, A. Kellerer, S. J. Goodsell, G. Rousset, E. Younger, M. Marteaud, A. G. Basden, F. Chemla, C. D. Guzman, T. Fusco, D. Geng, B. Le Roux, M. A. Harrison, A. J. Longmore, L. K. Young, F. Vidal, and A. H. Greenaway, "CANARY: the on-sky NGS/LGS MOAO demonstrator for EAGLE," Proc. SPIE **7015**, 70150E (2008).
12. E. Fedrigo, R. Donaldson, C. Soenke, R. Myers, S. Goodsell, D. Geng, C. Saunter, and N. Dipper, "SPARTA: the ESO standard platform for adaptive optics real time applications," Proc. SPIE **6272**, 627210 (2006).
13. ANSI VITA, "Serial front panel data port," Tech. Rep. ANSI-VITA 17.1-2003, VITA (2003).
14. S. J. Thomas, S. Adkins, D. Gavel, T. Fusco, and V. Michau, "Study of optimal wavefront sensing with elongated laser guide stars," Mon. Not. R. Astron. Soc. **387**, 173–187 (2008).
15. C. Petit, F. Quiros-Pacheco, J. Conan, C. Kulcsar, H. Raynaud, T. Fusco, and G. Rousset, "Kalman-filter-based control for adaptive optics," Proc. SPIE **5490**, 1414–1425 (2004).
16. http://www.omg.org/cgi-bin/doc?formal/00 10-33.pdf, "The common object request broker: architecture and specification" (2000).