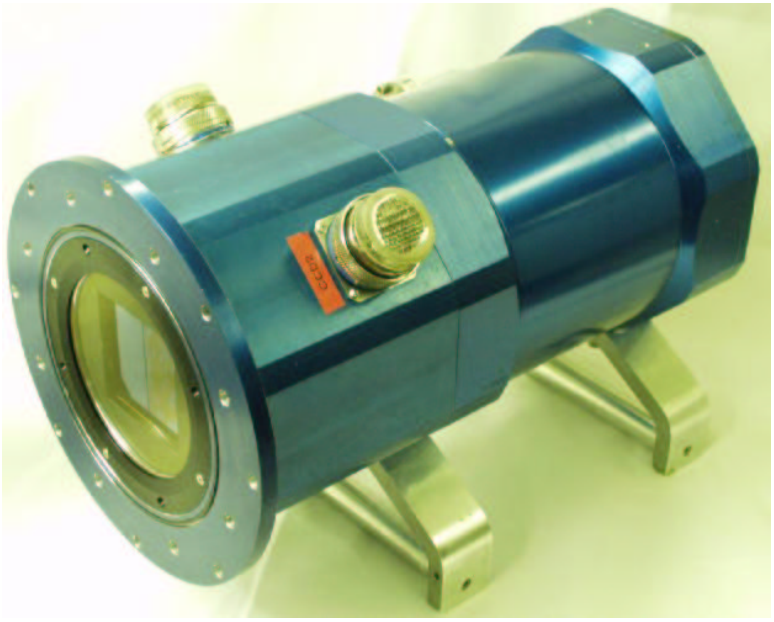
Contents

OPTIC

Orthogonal Parallel Transfer Imaging Camera



John Tonry

15 October 2003

1 Introduction

The Orthogonal Parallel Transfer Imaging Camera (OPTIC) consists of two Lincoln Lab CCID28 2K×4K chips mounted side by side in a dewar, so the imager is roughly 4K×4K altogether. The pixels are 15 μm, and at the UH2.2-m or WIYN they subtend about 0.14″. There are amplifiers at each end of each chip, so OPTIC conceptually appears as four 2K×2K CCDs in a square array.

OPTIC can function as a conventional imager, featuring a fairly rapid readout time ($\sim$25 sec full frame), fairly low read noise (4–5 e^-), extended red response, and low fringing. Its real advantage is its ability to do orthogonal transfer shifting which can be used to remove image motion, track moving objects, psf shape, etc. It also uses a "video" mode which can track up to four guide stars at fairly high rates (10-20 Hz) for measuring the image motion, doing the OT shifting, and possibly guiding the telescope.

OPTIC comprises a cryostat carrying the focal plane, SDSU-2 electronics to run the CCDs, an SDSU PCI interface in a Dell host computer running Linux, and a control program called "otcom". At the UH2.2-m otcom controls a shutter and filterwheel directly, and it communicates with the TCS for telescope functions. At WIYN otcom also uses TCS communication to access the shutter and filterwheel.

A general user will mostly be concerned with interactions with otcom. This program runs in a window and communicates with the user via text input and output. There is a modest set of commands to run the CCDs which are described below. Otcom can also accept socket connections from other programs which use the same text-based command interface. There is a GUI called "otgui" which displays status and commands in a variety of graphical ways, and translates actions such as button clicks or entry values into the text strings which otcom understands. The general user may very well only interact with otcom through otgui, but it is valuable to understand

the text commands because they can be bundled into scripts which can be executed by the interpreter.

The remainder of this introduction is fairly detailed information about the hardware, the CCDs, and the software. It is certainly worth skimming to get a general sense of how things work.

The next section is a brief tutorial on how an OPTIC observing session might run without the GUI. Following this is a section which provides a detailed description of all the otcom commands. This is the basic reference for all capabilities. The GUI is described next, but the descriptions rely fairly heavily on the commands already presented.

Subsequent sections provide information on data backup, utilities for making finding charts and logs, view data, and reduce data. There is a section which details how the guide star tracking takes place and the various strategies and algorithms which are available. Finally, an appendix gives information on a wide range of topics.

1.1 General hardware description

OPTIC has two Lincoln Lab CCID28 2K×4K chips mounted side by side in a dewar. The physical gap between the packages is 1 mm, and the dead area on the side of each chip contributes an extra 300 μm pixels, so the net gap between the two halves of the imaging areas is about 1.5 mm or 104 pixels. The dewar connects to SDSU-2 electronics which contains a 12-board backplane. At present there are seven boards present:

Timing board	–	the DSP which runs the whole thing
Utility board	–	another DSP which does auxiliary functions
Clock board (serial)	–	controls serial clocking on both chips
Clock board (chip1 parallel)	–	parallel clocking for chip1

Clock board (chip2 parallel) – parallel clocking for chip2

Video board (chip1) – biases and video processing for chip1

Video board (chip2) – biases and video processing for chip2

The connection between the dewar and electronics is by a pair of 18" 128-pin cables, and the connectors on the dewar go directly to the CCDs so **PLEASE NOTE:** do not touch these cables unless you know what you are doing. See the start up section of the appendix for details on handling these cables.

The SDSU-2 controller box is in turn connected to a power supply box via a 6' cable, a host computer via a pair of fibers, and the shutter and TFW filterwheel via a 15-pin, opto-isolated D-sub cable to the TFW controller box. There is also a BNC connector which can be used as an alternate TTL shutter signal, and there are two TFD's on the focal plane which are brought out to BNC connectors.

At the UH2.2-m approximately 30-50VAC exists between the metal of the back of the telescope and the AC ground found at the outlets on the dome floor, so it is essential that the dewar and SDSU boxes be insulated from the telescope and power brought up from the dome floor.

The SDSU controller talks to a Dell Dimension 4300 host computer with a 2 GHz processor, 0.5 Gb of memory, and 280 Gb of IDE disk, of which 250 Gb is available for data storage. It has a DVDR drive which can be used with to burn CDs or DVDs. This seems to be reasonably reliable, and a night's data will usually compress (using "gzip -1" or "compress") to less than the 4.7 Gb capacity of a DVD-R. See the data backup section for details on how to write CDs and DVDs.

The computer permits ftp and telnet as well as scp, ssh, etc, but please be careful about security.

Communications is through an SDSU-2 PCI interface board. The host is

running a nominal RedHat 7.3 Linux, although the kernel was rebuilt in order to enable IDE disk DMA as well as to provide a source tree that the device driver could be compiled with. The default kernel is called "2.4.18-3new", and "2.4.18-3" is the basic RH7.3 kernel. The device driver is provided by Sidik Isani at CFHT, currently lotuspci-2.11.

OPTIC includes a filter wheel/shutter assembly at the UH2.2-m. The filter wheel has four slots for 100mm round filters of thickness up to about 1cm. The filter wheel position is sensed with a magnet and Hall effect sensors, and has four detents and a fairly strong spring loaded roller to lock down the wheel at repeatable positions. The filter wheel has a hatch for changing filters. Note that the filter wheel has a moderately fragile drive mechanism, and will **not** turn freely without power. The shutter is a 100mm Prontor electronic unit. There is no provision for power reduction when the shutter is open, so it tends to generate a bit of heat. The filter wheel and shutter are run by a custom controller which connects to the SDSU utility board via opto-isolated signals through a 15-pin Dsub connector.

The Prontor shutter is not extremely fast, and the center of the field of view gets approximately 100msec more exposure than the nominal exposure time. For precision photometry, therefore, it is necessary to construct an exposure map which can be derived from the ratio of two flatfields of different (known) exposure times illuminated by a constant light source (a dome flat lamp which has been on for a few minutes is fine). I believe that the shutter exposure map is very reproducable, so I've put an exposure map called `optic_shutter.fits` in `/usr/local/inst/Data`. If you add this file to a constant nominal exposure time (in sec) you will get the actual exposure (as a function of position).

1.2 General CCD description

These CCDs have a number of unique features.

1. They are capable of "orthogonal transfer", i.e. the charge can be noiselessly clocked horizontally as well as vertically, which permits tracking of image motion and fast readout of guide stars.

2. They are each split into four regions with different clock lines so that each region can be clocked independently. These regions have no interruption of the metrology of the pixels however.

3. They have serial registers at both ends, and OPTIC is normally run using one amplifier at each end, hence four amplifiers altogether. In many ways it looks like four 2k CCDs butted together. This is the nomenclature normally used instead of the "chip1 and chip2" above.

The CCD are illustrated in figure 1-1. Various useful parameters are listed in table 1.2, and table 1.2 give the quantum efficiency of these devices as a function of wavelength. The thick, high-resistivity chips have enhanced red response and fringe very little in the I band. However, the U reponse is poor and quite non-uniform.

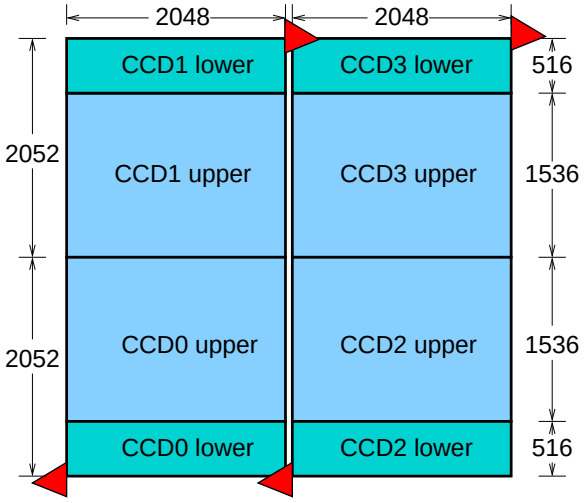


Figure 1-1: Dimensions of the CCDs.

The results of a readout through four amplifiers are always rotated and flipped in the output file into a physically consistent picture. The bias strips for all four regions are found on the right hand side in two stripes, with the

Table 1-1: CCD parameters

Parameter	Value
pixel size	15 μm
readout time (unbinned)	25 sec
readout time (binned 2x2)	8 sec
CCD thickness	45 μm
read noise	4 e^-
gain	1.4 e^-/ADU
full well	$>$ 80k e^-
non-linearity	$<$ 1% to $\sim$30k ADU
gap between CCDs	104 pixels
dewar hold time	$>$ 24 hour

Table 1-2: CCID28 Quantum Efficiency

λ	QE	λ	QE	λ	QE
0.36	42	0.60	77	0.95	44
0.37	44	0.65	83	1.00	23
0.38	46	0.70	88	1.01	18
0.40	52	0.75	90	1.02	14
0.45	57	0.80	88	1.03	11
0.50	63	0.85	83	1.04	8
0.55	70	0.90	68	1.05	6

left stripe from the left chip and the right stripe from the right chip. The top of the output image corresponds to the side of the dewar opposite the connectors. With a normal view of the sky (e.g. a lens or a Cassegrain focus) the parity of the output image is conventional, i.e. east is found counterclockwise of north. With another reflection off a flat mirror the parity is flipped. The gap between the CCDs is not represented in the output image in any way.

Note that readout parameters refer to each quadrant and each amplifier, so that reading out 2048×2052 gets the entire mosaic. Also offsets are amplifier-centric, i.e. a read with an offset skips pixels upstream of the amplifiers in the serial and parallel directions, which may cause a surprising piece of the CCDs to be read. The amplifiers of the four quadrants are

normally found in the lower left and upper right corners, but if the "geom" specification in the "status" command or the "CCD?GEOM" FITS header is 2 instead of 0 the amplifier is found on the opposite side of serial register.

Although the four regions in each chip are fundamentally the same, the "lower" regions enjoy the ability to reach the amplifiers without disturbing the upper regions. Therefore the "lower" regions are potential locations for guide stars, and the control software will permit you to specify zero or one stars in each lower region to be used as guide stars.

When exposing, the software quickly clocks a small patch which you specify around each selected guide star first horizontally and then vertically over to amplifier. This does not disturb the image in the upper regions or any unselected lower regions. The software then does a readout of the small patch, analyzes the guide stars, possibly performs shifts on the non-guide regions, and then pauses for another guide star image to collect. The overhead involved in reading out these small guide patches varies from 10 msec to 30 msec, depending on whether they are all close to their amplifiers or on the other side of the chip.

1.3 General software description

The software which normally runs OPTIC is called "otcom". The location of all software is /usr/local/inst, where subdirectories include

bin	–	binaries, should be first in path
config	–	setup files for telescope and filter wheel
doc	–	documentation
dsp	–	DSP code for the SDSU controller
etc	–	sample save files and sound files
pro	–	Vista procedures

script – scripts for otcom

src – source code for otcom and other programs

The normal places where data are saved are /b0/optic or /b1/optic. This should be writable for all users. General observers normally should log in as "obs" and there is a root account known as "otroot". See somebody knowledgeable about passwords.

When otcom starts up it first checks to see whether it can communicate with the interface, initializes it, then checks to see whether it can communicate over the fiber to the SDSU controller. If so, it reads a telescope configuration file and filter definition file. Finally, it begins accepting socket connections from client program(s), which might typically include the TCL GUI called "otgui", as well as accepting commands which are typed at the window where otcom is run.

An important thing to do at this point is to "download the DSP code". The DSPs in the SDSU controller boot from ROM, but need specialized code which is adapted to the particular CCDs and hardware connections and which can run the needed routines. For OPTIC, the normal DSP code is called "optic4.lod" for the timing board and "opticlub.lod" for the utility board. These are automatically downloaded when you type "df" at otcom. In principle this ought to be done as part of the initialization, but it isn't (yet).

Otcom has a simple interpreter which accepts commands (any unique abbreviation is acceptable and case does not matter), and carries out actions such as `clear 4` (clear the CCDs four times), or `go` (perform a readout), or `status` which lists the important variables and state of the camera. These will be detailed later.

The GUI allows the user most of the functionality of the command line interpreter in a more intuitive graphical format, but it is important to realize

that the standard input of otcom is normally accessible for typed commands. Each command which the GUI passes on to otcom is echoed in the GUI's log window, and these commands can be typed directly in the "Command:" entry window of the GUI or at otcom's standard input. The reason that it's worth going beyond the GUI is that otcom has a script facility (described below) which can be extremely useful, but it only uses the command set, of course.

Most of the communications from otcom are shown in the log window of the GUI but not all. In particular, if otcom requires interaction with you, perhaps because you entered a command with unacceptable arguments or because you are trying to overwrite an existing file, it does so via the otcom's standard input, *not* the GUI. Therefore otcom should never be run purely as a server in the background, but should be run as a normal program in a window of its own.

If OPTIC is run as a conventional CCD there are no other windows which the user needs to know about. However, if OPTIC is run with guide stars or in tracking mode there are two other windows which might appear. The command `fgs` (Find Guide Star) takes a quick exposure of the lower CCD regions (i.e. bottom 516 pixels), binned 2×2, and pops up an image from which the user can select guide stars. Its use is described below in detail, but this is how the user chooses stars (or not) for guiding and enters the coordinates.

When the user has selected "OT" mode and starts an exposure, another window, called "XCCDVideo", pops up which shows video of the guide stars and various statistics about them (also described in detail below).

Finally, OPTIC has a few utilities which are good to know about. One is called "othead" which will dump out one line of header information from an exposure (or a title line with no argument). This is helpful for making an automatic log of a night's observing. See the section on utilities for more details.

Another useful utility is "mkfinder" which will download a 15 arcminute image from the Digital Sky Survey, overlay an OPTIC footprint, and write a postscript file which you can print.

The third important utility is called "conflat". The result of OT shifting an image is that each pixel in the final image has actually been integrated on a variety of physical pixels. An exposure during which OT shifting has taken place normally writes several auxiliary files in addition to the final image:

`myprefix.???`	–	science data image
`myprefix_fgs.???`	–	guide star locating image used by fgs
`myprefix_gsc.???`	–	guide star locations which were chosen during fgs
`myprefix_gs.???`	–	table of guide star locations and properties
`myprefix_ot.???`	–	table of where each pixel was integrated
`myprefix_tg.???`	–	telescope guide commands (if enabled)
`myprefix_vid.???`	–	3D FITS file of all video images

Conflat takes an unshifted, unbinned flatfield and convolves it according to the data in the `_ot` file so that it agrees with the OT convolution which took place in the image file.

A note about file names. In some people's opinion, a mandatory suffix of ".fits" for FITS files is foolish and contributes to the general problem of RSI, and the refusal of the FITS standard to recognize unsigned 16-bit integers also foolish, given the properties of A/D converters, and is curiously in conflict with the 2880 byte block size and mandatory byte order on which FITS is based. Otcom by default writes files which consist of a prefix and a 3-digit suffix, e.g. "/MI5/bond.007", and writes *unsigned* 16-bit integers. If you don't like the file names, create a script which will rename them after the fact. If you insist on *signed* 16-bit integers (with an appropriate BZERO), you can use the "set" command described below to enable it. *(030120: I've knuckled under to the GUUC and made signed FITS files the default.)*

2 Tutorials

This section contains a variety of tutorials.

2.1 Quick start at WIYN

Startup procedures

In this section, * is used to denote the prompt for commands which the user supplies to otcom.

Log into the OPTIC computer mkbill as "obs", the password is written on the wall somewhere.

```
% startx   (This command starts the X-window system)
```

Open up at least two windows. It is recommended that these be placed in the directories `/home/bill/obs` and `/b1/optic/datadir`, where `datadir` is your chosen data directory for each night. For example, your data directory might be `/b1/optic/031017`.

In the /home/bill/obs window...

```
% otcom_WIYN
```

In the `/b1/optic/datadir` window ...

```
% otgui &
```

This command starts the graphical user interface (GUI). It must be loaded after `otcom_WIYN`. Section 4 of the OPTIC manual details the GUI but note that most of the buttons are *not* needed by the average user.

In the `/home/bill/obs` home directory window which is now running `otcom`:

```
* df
* restore vanilla_wiyn_17oct03
* temp -95.0
* pon
```

This command restores a save file of WIYN specific values for OPTIC as well as sets the instrument configuration in its simplest mode - that of a standard CCD camera.

Startup parameters

File prefix: otcom writes fits files of the form `prefix.001`, `prefix.002`, etc. where `prefix` is a user determined prefix. The user sets this as follows:

```
* fp /b1/optic/031017/s
```

In this example, it is assumed that the user has set up data directory `/b1/optic/031017` and otcom will write files `s.001`, `s.002`, etc.

File number: otcom starts with number 1, and increments by 1 each time an exposure in taken. If previous files exist in the datadir (you took 10 images but then exited otcom and returned) you might type

```
* fn 11
```

to tell otcom to call the next OPTIC frame `s.011`

Data collection

If you prefer to use OPTIC as a standard CCD imager, you are all set. You will need to set the filter and exposure time,

```
* filter 5
* et 100
```

OR in the GUI you can click on the filter button you desire and change the `ETime` value.

```
* go        (or clicking GO in the GUI)
```

starts the exposure or snap. You will receive a full frame normal CCD image

If you prefer to use OPTIC 's ability to provide electronic tip-tilt (OT mode), you have two additional steps. After slewing to your chosen part of the sky, take a test exposure to locate possible guide stars which are imaged in the four guide regions (see page 5 of the OPTIC user manual). Hopefully, at least one fairly bright star will be in the guide regions and better yet if four are. If less than the desired number are in the guide regions, you will have to tweak the telescope position to place at least one star in a guide region.

Once the guide stars are in the regions, you will need to mark their positions for OPTIC.

```
* fgs        (or click on FGS in the GUI)
```

This command does a 2 sec exposure and pops up a display of the four guide regions only. Move the cursor into each guide region onto the desired star's location. An `'x'` marks the spot and an `'e'` ignores the region setting no guide star and returning this guide region to the science frame.

You must now tell OPTIC you wish to use the OT mode during your science exposure. This is done by clicking the OT button in the GUI. So a `'snap'` makes a normal CCD image, a `'OT'` uses the selected guide stars

and OT shifts each science region during the exposure, providing electronic tip-tilt.

```
* go          (or clicking GO in the GUI)
```

Starts the OT guided exposure and pops up a video display for the users amusement and enlightenment. Section 4.3 in the OPTIC manual describes the workings of this window.

Some good commands to know

To stop an exposure and kill it without writing anything to disk, click on `PAUSE` in the GUI and once `RESUME` appears, click `Clear`.

ALWAYS remember to set the `PA` GUI box (position angle) to the value of the winr (WIYN rotator).

The OPTIC user has a choice of three image display utilities. On mkbill itself, Vista (OPTIC manual sections 6.7, 8.3see) or ds9 are available. The third choice is to ftp images to a WIYN computer and use IRAF.

Some specific detailed information on the mkbill connection at WIYN and known bugs and crash recovery are given in OPTIC manual sections 8.5, and 8.6.

A brief OPTIC command summary is given in Section 8.8 of the OPTIC manual.

2.2 Command tutorial

Here's a simple session you might carry out which will illustrate most of the important features of OPTIC. What you type is indicated by a >> prompt. This supposes that you are running otcom from the command line instead of the GUI. The GUI is actually doing no more than providing these text

commands to otcom, so this is probably the best way to get a sense of how all this works. Refer to the commands section for specifics on what all these

Start up `otcom` (or use `otcom_WIYN` at WIYN):

```
% otcom

OTCOM v4.8

logonly: Found SDSU card at /dev/lotuspci0, fd=3
Interface is alive,
logonly: Fiber reset to SDSU GEN2 mode
Interface is reset

 Controller alive, please download DSP code!

Reading tel_config info from /usr/local/inst/config/tel_config
Tel: 3.5m at f/6.3, Scale: 0.141"/p, Inst: OPTIC: 4-5-2 & 4-3-2

Atlas/XY: , Guider: viper, Port: 0
Reading from filter file /usr/local/inst/config/filters.def
Filter #0 is 'z', ZP = 25.10
Filter #1 is 'OIII', ZP = 25.00
Filter #2 is 'H-b', ZP = 25.00
Filter #3 is 'I', ZP = 26.40
Filter #4 is 'R', ZP = 26.50
Filter #5 is 'H-a', ZP = 25.00
Filter #6 is 'V', ZP = 26.50
Filter #7 is 'B', ZP = 26.40
cmdqueue: Communications to clients started on port 9511
error: Can't read symbol ST_SH...  Have you downloaded yet?
001>
```

Download the DSP code as requested:

```
>> df
High voltage off...
info: Downloading /usr/local/inst/dsp/optic4.lod...
logonly: 491 symbols defined
info: Verifying downloaded code...
info: Downloading /usr/local/inst/dsp/opticlub.lod...
logonly: 744 symbols defined
info: Verifying downloaded code...
warn: P:0x000091 -- 0x000004 sent, 0x053779 received
logonly: P:0x000091 -- 0x053779 OK now
Boot code updated to allow >7 arguments...
info: Filter is positioned at 3
Frame: 001   File prefix: '/tmp/ccd'   Filter 3 'I'   Shutter 0
Science:   bin 2x2, origin (0,0), size (1024,1026), 32 bias, 4 amp, texp 1.00
Guide:     bin 2x2, origin (1,1), size (24,24), geom 0,0 0,0, texp 0.100
GS at:     0.0 0.0 1  0.0 0.0 1  0.0 0.0 1  0.0 0.0 1
Params:    nwipe 1, shut 1, read 1, wrdat 1, track 0, wrot 0, wrvid 0
OT track:  alg 10, wait 5, gain 1.00, max 8, regions 11111111, wgt -1.00, bias 1
Tel guide: on? 0, rmin 2.0, tmin 1.0, gain 0.50, PA top 0.0, EccwN? 1
Move:      pat 0, dx 0.00, dy 0.00, #rep 0, max 1000
001>
```

Turn on the high voltage:

```
>> pon
 High voltage on...
 status: 0 p5v 5.05  p15v 16.4  m15v -16.4  p35v 36.0
```

Check that it happened:

```
>> volt
 status: 0 p5v 5.05  p15v 16.4  m15v -16.4  p35v 36.0
```

Set the temperature regulation to −105C (necessary every new download):

```
>> temp -105
 status: 0 tccd -104.9  tset -105.0  telec 22.1  hperc 0
```

Set the file prefix and file number:

```
>> fp /b0/optic/021217/f
 status: 0 fp /b0/optic/021217/f

>> fn 24
 status: 0 fn 24
```

Check out the status of everything

```
>> status
 Frame: 024   File prefix: '/b0/optic/021217/f'   Filter 3 'z'   Shutter 0
 Science:   bin 1x1, origin (0,0), size (2048,2052), 32 bias, 4 amp, texp 20.00
 Guide:     bin 2x2, origin (1,1), size (32,32), geom 0,0 0,0, texp 0.050
 GS at:     0.0 0.0   0.0 0.0   0.0 0.0   0.0 0.0
 Params:    nwipe 1, shut 1, read 1, wrdat 1, track 0, wrot 0, wrvid 0
 OT track:  alg 10, wait 5, gain 1.00, max 8, regions 13331111, wgt -1.00, bias 1
 Tel guide: on? 0, rmin 2.0, tmin 2.5, gain 0.50, PA top 180.0, EccwN? 0
```

Set up for a dome flat, full chip, unbinned, 32 bias, 6 sec exposure:

```
>> auto 1 1 1 1 0 0 0
>> sf 1 1 0 0 2048 2052 32
>> et 6
```

Take an image and write to /b0/optic/021217/f.024:

```
>> go
```

Take a 2×2 binned snapshot of a field where you want to guide. (If all you want to do is use OPTIC as a conventional CCD with telescope guiding, this is all you need to ever do.)

```
>> auto 1 1 1 1 0 0 0
>> sf 2 2 0 0 1024 1026 32
>> et 2
>> go
```

After examining it, move the telescope to place a suitable guide star in position. It was found at 412 611 and you want it at 100 100, these coordinates being of the previous image which is binned by 2:

```
>> telescope xyoff 412 611 100 100 2
```

Find the guide star(s) and mark them with a 4 second snapshot:

```
>> fgs 4
```

This pops up an image of just the "lower" regions, binned 2×2, so you get an image which is 2048 wide by 512 tall, and it is displayed with a $2\times$ compression so you see 1024×256 for an effective binning factor of 4. The four "lower" regions are delineated with a green cross. You then make a selection in each of the four lower regions, either hitting `e` in the displayed image to make no star selection or `x` to choose a star. (If your star has saturated you can force the selection of a particular location by using capital C, `C`, instead of `x`.)

When you hit the fourth key (either `e` or `x`) the window disappears and a new status display should show you non-zero numbers for the guide stars you have selected. If for some reason a selected star is not given reasonable coordinates, try fgs again.

You are now ready to take an OT image. Set the auto parameters, choose a 200 second exposure time and a full frame, unbinned readout, a 50 msec guide star exposure time (plus overhead), and go!

```
>> auto 1 1 1 1 1 1 1
>> sf 1 1 0 0 2048 2052 32
>> et 200
>> gt 50
>> go
```

This will pop up a new window called XCCDVideo which has lots of interesting features described in the GUI section of the manual. It is for your amusement and is not important for the exposure and star tracking. In previous versions it was possible to change things such as bin factor and guide star exposure time on the fly; they might be restored someday.

When the exposure finishes the auto parameters will cause the shutter to close, the CCD to read out, the image to write to f.026 (f.024 was the dome flat, f.025 the snapshot; the fgs image is called f_fgs.026), and various other data files pertaining to the tracking to f_*.026 (described below).

,

3 Detailed listing of otcom commands

3.1 Startup

Download DSP code

`df` – this tries to turn off the high volts and then downloads the code to both timing and utility boards. The default is optic4.lod and opticlub.lod found in `/usr/local/inst/dsp`. The environment variable LODPATH (default `/usr/local/inst/dsp:.`) governs where DSP files are sought.

Test communications, Power on, Reset

`pon` – turns on the high volts (i.e. $+/-15$ and $+35$. The idea is that we want the $+5$V on and stable so that the DACs which will supply voltages to the CCD are stable before their power is turned on.)

`pof` – turns off the high volts.

`tdl` – test data link, pings the CCD controller board

`system-reset` – full reset timing board via controller

Verify temperatures and voltages

`temp` – view temperatures of CCD and electronics

`temp -105` – set CCD regulation temp to -105

`volt` – view voltages

Status

`status` – shows what's doing, crowded but pretty complete

```
Frame: 001   File prefix: '/tmp/ccd'   Filter 1 'R'
Science:   bin 1x1, origin (0,516), size (2048,1536), 32 bias, 4 amp, texp 1.00
Guide:     bin 2x2, origin (1,1), size (16,16), geom 0,2 0,2, texp 0.020
GS at:     0.0 0.0 1  0.0 0.0 1  0.0 0.0 1  0.0 0.0 1
Params:    nwipe 1, shut 1, read 1, wrdat 1, track 0, wrot 0, wrvid 0
OT track:  alg 0, wait 5, gain 1.00, max 8, regions 11111111, wgt 2.00
Tel guide: on? 0, min 2.0, gain 0.35, PA top 0.0, EccwN? 1
Move:      pat 0, dx 0.00, dy 0.00, #rep 0, max 1000
```

The first three lines are relatively self explanatory, although it's important to realize that the guide box specification has two parts. There is a common readout "bin, origin, size" for all stars, which is used in reading out the little regions around them. The "bin" and "size" define the box parameters, and the "origin" defines where the boxes are shifted prior to readout. An origin of (0,0) or (1,1) is good.

The fourth line defines the center location for each of the stars (or zero if that region is not going to be used). It is set by fgs or guide. The integers following the coordinates are the dwell times requested by `gdwell`. The fifth line describes the OT tracking parameters which are explained in detail below. The sixth line does the same for the telescope guide parameters. The last line lists the `move` parameters and the "otmax" variable.

3.2 Normal CCD control commands

`fnumber N` – set the running file number to `N`

`fprefix xx` – set the file prefix to `xx`, i.e. output file will be `xx.NNN`

`etime T` – set exposure time to `T` seconds

`sformat` – set the CCD readout format. This can take 7 arguments or otcom will prompt you for them if you do not supply them. The arguments are (in order)

xbin ybin xoff yoff xsize ysize nbias

where the bin factors can be 1–N, the offsets are measured in physical (not binned) pixels, and the sizes and number of bias overclocks are in binned pixels.
e.g. sf 2 2 0 0 1024 1026 32

`filter` – request current filter number

`filter (N)` – set the filter to `N`

`go (N)` – Do an exposure. If `N` is present you get `N` identical exposures. This can perform a large variety of actions depending on the "auto" parameters. There are also piecemeal versions of go:

`clear (N)` – clear the CCD (`N` times)

`shutter A` – open/close the shutter if `A` is "open" or 1 / "close" or 0

`expose` – open shutter (if auto permits), wait, and close shutter

`rc` – read out CCD (and write files if auto permits)

`wf` – write files (according to auto)

`auto` – set up the automatic actions to be undertaken with "go" This can take 7 arguments or otcom will prompt you for them if you do not supply them. The arguments are:

#wipe do_shut? readout? wr? do_OT? wr_OT? wr_vid?

#wipe	=	number of clears before an exposure
do_shut?	=	0/1 to open the shutter for an exposure
readout?	=	0/1 to readout after the exposure completes
wr?	=	0/1 to write the image to a file
do_OT?	=	0/1 to follow guide star(s) and do OT shifting
wr_OT?	=	0/1 to write the (all important) OT data file
wr_vid?	=	0/1 to write the 3D FITS file of the GS video

some examples will make this clearer:

`auto 1 1 1 1 0 0 0` normal snapshot

`auto 1 0 1 1 0 0 0` bias or dark frame

(depending on whether `etime`=0 or `etime`>0)

`auto 0 1 0 0 0 0 0` focus frame

(would be preceded with clear, and followed with an `auto 1 1 1 1 0 0 0` and an `rc` or an `auto 0 0 0 1 0 0 0` and a `go`)

`auto 1 1 1 1 1 1 1` OT tracking image

`auto 1 1 1 1 1 1 0` *LONG* OT image

`auto 1 1 1 0 0 0 0` exercise CCD

(maybe while looking at signals with an oscilloscope)

`video` – run video for etime seconds. This does not attempt to find or track guide stars, it simply provides a view of what is in the selected guide regions. `video` is interrupted by any command, and it does not attempt to save any data upon completion.

`tel cmd` – do a telescope command. There are a variety of commands available for a given telescope. You can probably count on the following ones:

`tel roff E N`	do an offset from the current position by `E` and `N` arcsec
`tel xyoff dx dy`	offset by `dx, dy` unbinned pixels
`tel xyoff x0 y0 x1 y1`	offset from posn 0 to 1, unbinned pixels
`tel xyoff x0 y0 x1 y1 bin`	offset from posn 0 to 1, `bin` pixels
`tel afocus F`	set the telescope focus to `F`
`tel dfocus D`	change the telescope focus by `D` units

Note that the `tel xyoff` command depends on otcom knowing the orientation of the CCD; see the `gparams` command below.

At the UH2.2-m there is a large variety of commands which are probably already familiar to users of tcdcom, the most important being:

`tel goff E N`	offset both telescope and atlas guider

At WIYN there are three other commands:

`tel info`	dump out the telescope's idea of things
`tel filter N`	set the filter to F (note that this currently uses `N`=0-7 for WIYN filters 1-8, sorry)
`tel exptime T`	set the WIYN shutter exposure to `T` this is overridden by `etime` if `T`>4 sec, and `etime`>4 sec, but defines the actual exposure for 1<`T`<4 sec.

3.3 Guiding commands

`sgformat` – set the guide box parameters. This can take 6 arguments or otcom will prompt you for them if you do not supply them. The arguments are (in order)

xbin ybin xoff yoff xsize ysize

where the bin factors can be 1–N, the offsets are measured in physical (not binned) pixels, and the sizes and number of bias overclocks are in binned pixels.

e.g. sg 2 2 1 1 16 16

The important part of this is the bin factor and box size, the offset simply is where the guide patches are temporarily lodged before readout. The `fgs` or `guide` commands are how the actual locations of the guide stars are specified.

`gtime M` – set guide exposure time to `M` msec. The actual exposure time will be a bit longer because the time the host spends analyzing the guide images adds to the integration time, and the overhead time of readout contributes to the net frame time.

`fgs (T)` – takes a quick exposure of `T` (default 2) sec of the four lower regions, pops up a display window, and expects you to make a selection in each region of `e` for no guide star or `x` to select a guide star.

You can also use `fgs` non-interactively with 10 arguments, which are exposure time, tolerance (unbinned pixels), and eight coordinates in physical coordinates (or two zeros for any region with no star). In this mode, `otcom` takes a snapshot of the requested time, finds the brightest star within the tolerance distance of the named coordinate, calculates the median offset, does a telescope offset to bring the stars to the named coordinates, and then enters them as the current guide coordinates. This is particularly suited for scripts which try to move between fields where the guide stars are known but the telescope offset may not be accurate enough to go directly to a `guide` command.

`guide par` – set the guide locations by hand. In this case there are 8 parameters, the x y locations of guide stars for each lower region, in pixel coordinates of the current binning. The coordinates are what you would report if you had just taken a snapshot and not changed the readout parameters yet. Use 0 0 to indicate a region where there is no guide star present. This is seldom used.

If the eight arguments are followed by `physical` the coordinates are taken to be in the physical coordinate system (as reported by `status` and saved in the `gx` and `gy` variables), not in the current readout system. If you want to force the telescope to return to the specified location, use the `telfix` variable in conjunction with this command.

A more common usage is if guide gets two arguments, which are interpreted as x and y offsets in pixels to be applied to existing guide star locations. This is useful for dithering, e.g.

```
tel xyoff 50 50
guide 50 50
```

Note that although both are in pixels, the sense is deliberately opposite: dx and dy move the telescope in the $+dx$ and $+dy$ direction, whereas the guide star positions are offset by $-dx$ and $-dy$.

A final usage is `guide prev` which simply restores guide positions which have been previously set with an `fgs` or `guide` command. A runaway guide star could be returned this way.

`tparams` – set up the OT tracking parameters with 6 arguments or otcom will prompt you for each. The arguments are:

alg wait gain max region_use exponent bias

e.g. tpar 10 5 1.0 8 13331111 -1.0 1

alg = decimal coded number of various prediction parameters – see the section on tracking for definitions and details about how the predictions are done.

wait = number of guide iterations to wait before applying OT shifts to the science regions. This can be set to a large number (50000) to use only telescope guiding and no OT tracking, or it can be set to zero if you are trying to track a moving object.

gain = fraction of indicated shift to apply

max = maximum offset permitted a box to follow a guide star (pixels)

region_use = 8 digits for the regions in order *0l 0u 1l 1u 2l 2u 3l 3u*

1: (T) do OT tracking

2: (G) use for a guide star

(Note that uses of 1 and 2 get set and reset by fgs.)

3: (Q) leave quiescent, no OT tracking, telescope guiding only

4: (P) push-away, steadily clock charge into the scupper to overcome a horrible bright defect or star.

5: (N) region is napping during a dwell

exponent = exponent for how multiple guide stars contribute to the shifts applied to different regions. The default of -1.0 means "use the closest guide star". A positive exponent means weight the different GS by inverse distance to the exponent power.

bias = 0/1 to do bias subtraction on the GS video frames

`gparams` – set up the telescope guide parameters with 6 arguments or otcom will prompt you for each. The arguments are:

on? rmin tmin gain PA EccwN?

e.g. gparams 1 2.0 2.5 0.5 180 0

on?	=	0/1 to turn on telescope guiding, i.e. a low-pass filtered error signal from the guide stars making offset requests of the telescope
rmin	=	minimum distance (pix) to trigger a guide request
tmin	=	minimum time (sec) between guide requests
gain	=	fraction of indicated offset to request
PA	=	position angle on the sky of the top of the CCD
EccwN?	=	0/1 parity, 0 for E clockwise of N, 1 for E CCW of N

Note that the last two parameters must be set properly if you are going to use the `tel xyoff` command, regardless of whether you are doing telescope guiding.

`gdwell` – ask for extra dwell time for four guide regions. The default is for each designated guide star to be read out every iteration. If one star is much fainter than another it may have very poor signal to noise. You can ask for that a guide region be read out only every N times the base rate in that case. For example:

gdwell 1 2 3 5

would read out the first region at the rate requested by gtime, the second region would be read out every 2nd iteration, the third region every 3rd time, and the fourth region every 5th time. The resulting video file will have bias data during the times when a region is not read out. Note that you must have one dwell time set to 1.

Dwell times are deliberately "fragile" and an `fgs` or `guide` command will silently reset them to 1.

`move` – ask for charge to be shifted during an exposure in the regions designated as "science". Move takes a variety of possible arguments:

move none	–	no extra OT shifting during an exposure
move offset dx dy	–	immediately move the charge in all the science regions by `dx,dy` unbinned pixels. This can be useful for focus frames or for reading out a contiguous subarray.
move rate dE dN	–	during an OT exposure, move the charge by `dE,dN` milliarcsec per second. This depends on having the PA and parity correct, of course. The guide regions work normally to keep the guide stars fixed on the detector, but the charge created by a moving object can be made to follow the object.
move box S move box dx dy move box dx dy N	–	during an OT exposure, move the charge in a rectangular raster scan of dimensions `S`×`S` or `dx`×`dy` unbinned pixels. If `N` is present, the raster is performed `N` times during the exposure. Note, however, that the charge is shifted only upon guide star readouts, is not continuous, and therefore can cause beat patterns if `N` is too large or discrete psf lumps if `dx*dy` is larger than psf^2 * `etime/gtime`.

Be careful to turn off moving (`move none`) when you are finished with your set of exposures; otcom does not do it for you.

3.4 Generally helpful commands

`help` – list all commands with a very brief description

`quit` – quit otcom, ask whether the current state should be saved

`Quit` – quit otcom, without saving the state

`ls (arg)` – list files

`cd D` – change directory to `D`

`set` – set all sorts of internal variables. The basic syntax is

`set var val` - to set variable `var` to value `val`, or

`set var` - to examine the value of variable `var`

In many cases `set` can be used in place of the normal otcom verb to set variables, but when in doubt use the normal verb to set things, since it may carry out other functions besides simply setting the variable. Typing

```
set help
```

will list all the variables that set knows about, their values, and a description of what they mean. There are a few variables which can only be accessed via the set command:

clobber = 0	:	ask before overwriting a file
1	:	overwrite files of the same name silently
shortfits = 1	:	write signed short FITS files with BZERO=32768
0	:	write unsigned short FITS files with BZERO=0
otmax = 1000	:	maximum OT shift permitted per request.

`! cmd` – execute (and wait for) a shell command

`# anything` – comment

3.5 Set header quantities

`object xxx` – Set the object name for header to `xxx`. If `xxx` is `read_base` otcom will attempt to fill in the object field from the TCS basename

`imtype` $\langle N \rangle$ – Set image type: N = 1-7 for `Obj Bias Dark Dome Sky Foc Std`

`comment X` – Set the header comment to `X`

3.6 Save and restore the state of otcom

`save (F)` – save the current setup (to file `F`). If `F` is not present, the setup will be saved to `/.otcom.save`.

`restore (F)` – restore saved setup parameters (to file `F`)

3.7 Interrupt an exposure

`interrupt` – interrupt an ongoing exposure (just type `interrupt` and a newline). The shutter will be closed and the exposure paused. A `clear` will reset the exposure.

`^C` – Type ^C at the otcom window to immediately interrupt whatever is going on and reset the controller link

`resume` – resume an interrupted exposure where it left off.

3.8 Macro (script) facility

`call F (args)` – execute a script of commands in file `F` with the specified arguments

Otcom has a simple but fairly powerful macro facility. It is invoked as `call macro_file` where the path of `macro_file` is determined by the environment variable `SRCPATH` (`.:/usr/local/inst/script` by default) or the current directory. Script files can call other script files to a very deep level. Arguments can be provided to script files as white space separated tokens. Within the script tokens of the form `$n` are replaced with these argument strings. Only one pass is made of substitution. The argument `$0` is the name of the script file itself, argument `$1` is the first argument, etc. For example:

```
call hoser this is a test
```

within the script hoser, the following replacements would be made:

```
$0 = hoser
$1 = this
$2 = is
$3 = a
$4 = test
```

and if hoser had a line like

```
call bingo \$4 \$0
```

the script bingo would have the following substitutions made:

```
$0 = bingo
$1 = test
$2 = hoser
```

It is important for many commands that arguments be present (for example if `sf` had too few arguments in a script it would go into a mode of prompting standard input for answers and hang). The script facility therefore permits default settings for arguments so that a user cannot get into trouble by calling a script without the right number of arguments. If the first line of a script starts `DEFAULT:`, the following tokens are taken to be defaults for the ensuing arguments. For example, this is a good start to an script to take a focus frame which steps the focus and telescope through a range:

```
DEFAULT: 5000 20
# Args are $1 = focus start
```

```
#          $2 = focus increment
#
tel afoc $1
et 2
auto 0 1 0 0 0 0 0
sf 2 2 0 0 1024 1026 32
clear
go
tel dfoc $2
tel xyoff 0 50
```

There are some scripts in `/usr/local/inst/script` which will offer some examples. If an error is encountered during the execution of a script, the entire input sequence is aborted and otcom returns to the command line prompt. Two useful scripts which are worth examining are:

```
call autofoc <start_value> <step_size>
call dither <xoff> <yoff>
```

4 The Graphical User Interface

4.1 Starting the GUI

The otcom GUI is invoked as `otgui`. It cannot be started until otcom is running. There is no real limit to the number of clients which otcom will serve, so many copies of otgui can be run. Otgui can accept a number of command line arguments.

```
-countdown    - Count down exposure time to zero instead of
                counting up
-countup      - Count up exposure time to completion instead
                of counting down
-id "my name" - Set a name associated with this client; useful
                if you are eavesdropping and want to be friendly.
-temp T       - Request a temperature T (default -105) for
                regulation when the GUI executes a 'pon' command.
-host XYZ     - Connect to otcom running on host XYZ instead
                of localhost
-eavesdrop    - View what's happening with otcom, but send no commands
-cmd          - Tell otcom to accept commands from this client (this is not
                really implemented yet; the eavesdrop flag works on the
                TCL end just as well as this might on the server side.)
```

Normally when otcom runs xoptic in video mode it will put up a display on the machine running otcom. If you want to alter this you can use the environment variables `VIDEOHOST` to specify which machine will run xoptic; `localhost` is the default. The environment variable `VIDEODISPLAY` is a normal X specification for where xoptic should put up its window. (It is also possible to specify a display program other than xoptic using the `VIDEOPROG` environment variable.)

It is possible that you want to have multiple copies of the xoptic windows for eavesdropping purposes. If so, a client can ask for its own video channel. (Again, you don't normally do this if you are the only one using otcom – the server itself always puts up the video window governed by `VIDEOHOST`.)

If a client wants to request an extra copy of the video windows, it can use the following flags.

```
-video        - Ask otcom to run xoptic video for this client (using
                the default DISPLAY if it exists)
-vhost host   - Host where the xoptic video should run
-vdpy display - Tell otcom which X display to use for xoptic (:0.0)
```

The `-video` flag by itself requests video run on the machine which originated the otgui on the `:0.0` X display. In order to get xoptic to run on a different machine than the one running otgui, use the `-vhost host` syntax. Finally, if you want X to display at other than `:0.0`, use the `-vdpy display` argument. If you have asked for xoptic to run on a different machine than the one running otcom (i.e. `VIDEOHOST` or `-vhost host` different from `localhost`), the process on mkbill running otcom (normally belonging to `obs`) must be able to rsh to the destination machine, and xoptic must be in the path of this rsh'ed process.

Some examples might be helpful. Suppose a person from a distant machine called `morass.anu.edu.au` wants to eavesdrop on otcom which is running on `mkbill.ifa.hawaii.edu`.

The simplest way to do this is to have `mkbill` run an otgui and xoptic for the remote person and use X to make the display connection. You would enable X to display from `mkbill` to `morass`, log into `mkbill`, and run otgui:

```
morass% xhost mkbill.ifa.hawaii.edu
morass% ssh mkbill.ifa.hawaii.edu -l obs
mkbill% otgui -video -eavesdrop -vhost localhost -id Dorothy \
```

```
        -vdpy morass.anu.edu.au:0.0
```

You could use "rlogin" and a "setenv DISPLAY" instead of ssh, of course. The disadvantages of this are two: you have to be able to ssh to `mkbill` and it loads down mkbill to run two copies of otgui and xoptic and use X for communication. Note that the `-vhost localhost` is explicitly telling `mkbill` to run xoptic locally.

The next most simple way to do this is put a local copy of otgui on `morass` and let it make the connection directly to `mkbill`:

```
morass% otgui -video -eavesdrop -vhost localhost -id Dorothy \
        -host mkbill.ifa.hawaii.edu -vdpy morass.anu.edu.au:0.0
```

This still loads `mkbill` with two copies of xoptic and incurs the X overhead discussed above, which can cause serious delays in `mkbill`'s ability to guide at a reasonable rate.

The best way to eavesdrop is to compile a local copy of xoptic as well as otgui. In order to permit the socket connection between otcom on `mkbill` and xoptic on `morass` you need to tunnel through the firewall via an ssh. Do this by

```
ssh -l obs -L9511:127.0.0.1:9511 mkbill.ifa.hawaii.edu
```

This ssh running locally is listening on port 9511 locally and talking to port 9511 remotely (otcom). Once this is done, you can connect to otcom (with no video) via this ssh using:

```
otgui -host 127.0.0.1 -id "Dorothy in Oz" -eavesdrop
```

If you want to have video appear on the remote machine as well you would invoke otgui as:

```
otgui -host 127.0.0.1 -id "Dorothy in Oz" -vhost morass.anu.edu.au
```

It must be possible for "obs" to do an rsh from `mkbill` to `morass`, i.e. obs has to have an account on `morass` and must have `mkbill.ifa.hawaii.edu obs` in its `.rhosts` file. (Remember that the `.rhosts` file must be readable only by the owner, and you might need an xhost request for the xoptic process to have permission to put up an image, for example `xhost localhost` might be necessary.) You also need to have otgui and xoptic set up properly on `morass` to run. You should then find that video which appears to the observer on `mkbill` also appears on the the remove machine, although a large fraction of the frames may be dropped (this is harmless).

See the section on remote observing for more detailed instructions how to set all this up.

4.2 Main GUI window

Most of the GUI should be familiar from the descriptions of the otcom commands in Manual_otcom_cmd, and detailed descriptions of what the commands do are found there.

Starting at the top of the GUI, the upper left has a menu bar with four main items and a variety of subitems:

```
Init –          View –              Tel –                 Help –
  Download        OT params           On/Off –              otcom commands
  Power           Shift Chg (P)         on / off            Manual –
  Exit            Shift Chg (W)       Windscreen –            many entries...
                  Object (P)            obs / park            JT Phone
                  Object (W)          Mirror Cover –
                  Recent Macro          open / close
                                      Dome Fluorescents –
                                        on / off
                                      Dome Incandescents –
                                        on / off
```

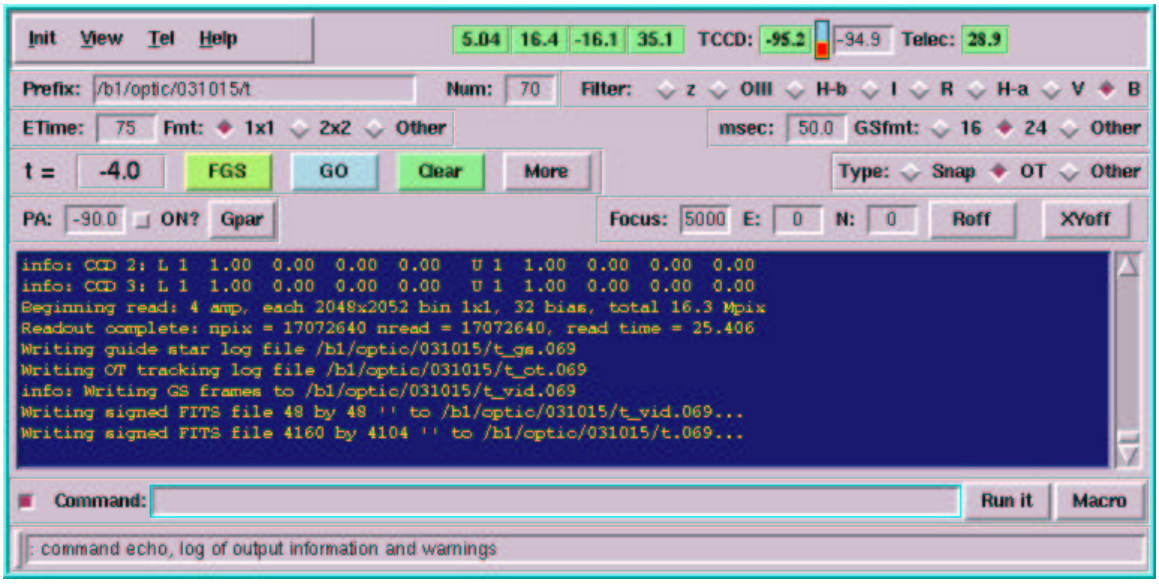


Figure 4-2: Normal appearance of the GUI.

The "Init" commands are for downloading (`df`) and powering on (`pon`), and the final menu item terminates the GUI (but not otcom).

The "View - OT params" command pops up a window to control the OT parameters (`tparam`) and will be discussed in detail below.

The "View - Shift Chg" pops up a window (P option) or adds a window (W option) which allows control of the `move` command.

The "View - Object" pops up a window (P option) or adds a window (W option) which allows setting the image type and object name. Using the `Base` button sets the object name to the current basename held by the TCS.

The "View - Recent Macro" pops up a window which keeps track of your macro calls and allows you to reexecute any one by clicking on it.

The "Tel" menu items, for the UH2.2-m only, implement some telescope commands which are helpful for taking dome flats.

The "Help" menu items will pop up various sections of the manual. Please note that the `JT Phone` numbers will automatically switch you to a

900 number and you will be charged $20/minute.

The upper middle of the top bar shows the voltages which are green if they are in range and red if not (the `volts` command). The upper right shows the CCD temperature (`temp`) and an entry window where the temperature can be set, followed by the electronics temperature. Again a green color indicates good values and blue or red indicates too cold or hot. Between the CCD temperature and the CCD set temperature is a little thermometer graph which shows the percentage of heater which is on to maintain the CCD temperature.

The next line in the GUI is an entry window for setting the file prefix (`fprefix`) and file number (`fnumber`). Clicking on these allows you to enter a value and hitting return within the window executes the command.

There is a subtlety to these (and other entry forms), namely that they are intended both to display the current state of otcom as well as accept input to send to otcom. Otcom updates the GUI every 30 seconds if otcom is otherwise idle (the otcom heartbeat) and this could interfere with user input. Therefore every time you click on an entry form to start input, the background turns pink and the entry is disconnected from being updated. When you hit ¡enter¿ or use whatever other means are required to send your value to otcom the form changes to the normal background and raises up, indicating that it is again reflecting the state of otcom. Therefore beware of pink entry forms. The color indicates that you may not have sent a value that you intended to, and at the very least the form is not keeping up to date with otcom.

Right of the file entry forms is the filter selection buttons. They indicate the current filters and if you click on a button it will execute the command (`filter`) to change filter.

The next line is the exposure time and format control. The left half is the science observations, "ETime" sets the exposure time in seconds (`etime`), and you can choose a 1×1 or 2×2 full-frame readout with the two buttons.

Clicking the "Other" button will pop up a new row which offers complete control of readout format (`sformat`). There is a button to execute the command and another to dismiss the popup.

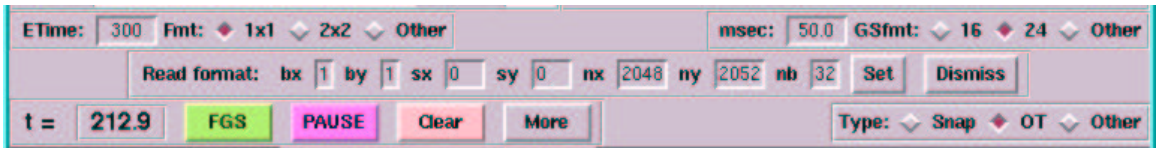


Figure 4-3: GUI with the detailed set format window open.

To the right is the guide time and format control. You can select the guide exposure time in milliseconds (`gtime`), and guide formats which are binned 2×2 and have size 16×16 and 24×24. Other formats can be obtained with the "Other" button which pops up a new row which offers complete control of guide readout format (`sgformat`).

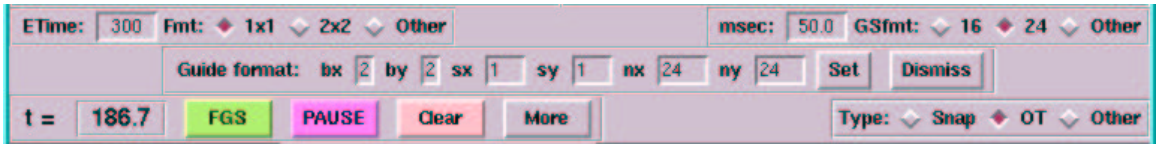


Figure 4-4: GUI with the detailed set guide format window open.

The next row down is the exposure control. The left half of this line has an elapsed time counter, buttons to do an "FGS" (with default exposure of 2 seconds), a "GO" button (the button changes to "PAUSE" and "RESUME" during an exposure), a button to "Clear" the CCD (its color changes between green and pink depending on whether the shutter is closed or open). The individual exposure commands can be accessed by using the "More" button to pop up a new row with "Shutter", "Expose", "Read", "Write", and "Video".

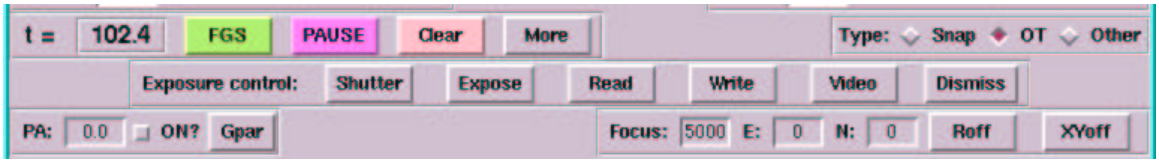


Figure 4-5: GUI with the exposure buttons revealed.

To the right is the control of the auto parameters. Three buttons are available: "Snap" to take an unshifted snapshot (`auto 1 1 1 1 0 0 0`), "OT" to take a tracked image (`auto 1 1 1 1 1 1 1`), and "Other" which pops up a new row where each of the auto options can be checked off and then set.

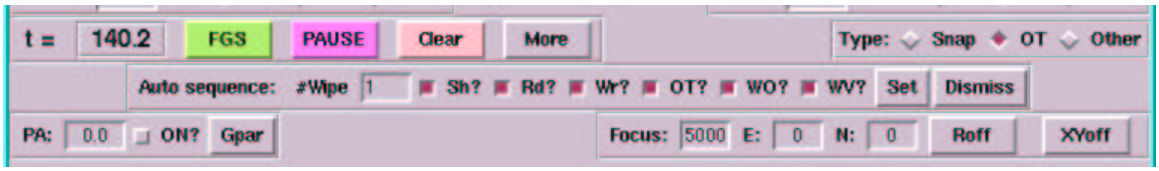


Figure 4-6: GUI with the automatic function widgets revealed.

The two essential and variable telescope guide parameters are visible at the left of the next line. "PA" displays (and sets) the PA of the top of the detector. The checkbutton tells whether telescope guiding is enabled or not. The "Gpar" button pops up a row which offers control of the rest of the guide parameters (`gparams`).

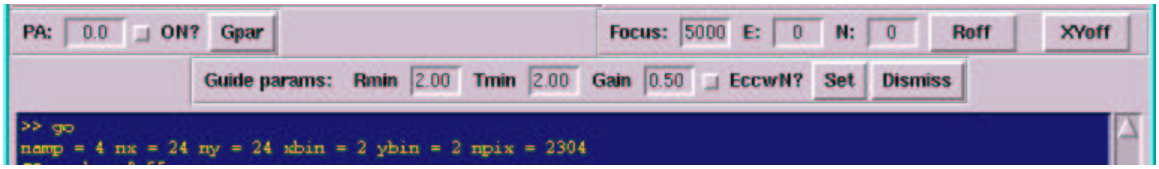


Figure 4-7: GUI with the telescope guide controls revealed.

The right half of this line is used to issue telescope offset commands. The "Focus:" entry can accept an absolute focus value and hitting Enter will issue the `telescope afocus` command. If the value entered is less than 500, it is assumed to be a relative focus offset and a `telescope dfocus` command is sent from the GUI instead. Note that this entry form is not updated automatically according to the current focus. The next two entries are used to enter offsets in arcseconds and issue them to the telescope. The "XYoff" button pops up a new window which offers the two `tel xyoff` formats: two arguments for an offset in unbinned pixels and five for a from,to x,y pair with a specified bin factor.

Figure 4-8: The telescope xyoffset popup.

The "View - OT params" menu item pops up a window which shows the current OT parameters (`tparam`), as well as offering a menu to set the prediction and tracking algorithm. To the right is an attractive schematic of the two CCDs whose color reflects each region's use. Guide stars are drawn in this window, and because of what appears to be a bug in Tk they are never deleted until the window is dismissed and reinvoked. There are little entry windows on the radio buttons for each guide window which contain integers (normally 1). These are used to set the dwell time for the video regions along with the "Set Dwell" button on the top line.

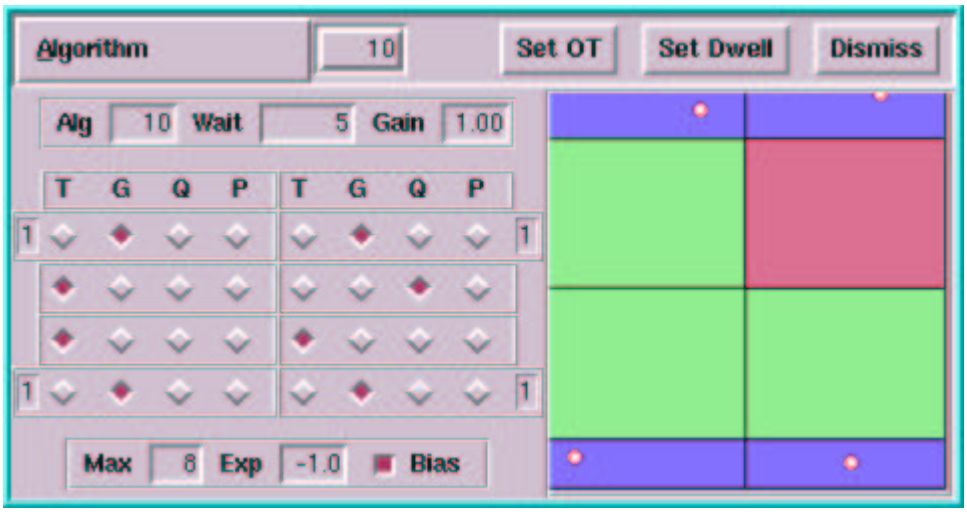


Figure 4-9: The OT tracking popup.

The "Algorithm" menu is discussed at greater length in the tracking section of the manual.

Finally, there is a log window where the commands which are being issued to otcom are echoed after a >> prompt, and output information from otcom is printed. Below this is an entry window in which you can type any command and send it to otcom. This will be sent if you hit enter or if you

click the "Run it" button. To the left of the command window is a little button which changes state with each heartbeat, i.e. with each command or every 30 seconds. The lower right has a "Macro" button which can be popped up to select a script. Selection of a script and use of the "Open" button will do a `call script`, but it's far easier to just type it. Recall that there is a menu entry under "View" which will bring up a new window in which macro calls are maintained for reexecution with a single mouse click.

The bottom-most line of the GUI provides a line of description for whatever widget the cursor is visiting. You can get a sense of what the various widgets do by looking at this line, and you can begin to learn what otcom commands are invoked for various functions. Referring back to the command section of this manual will provide details on what otcom does and what parameters mean.

4.3 Xoptic

Xoptic is the program which accepts a socket connection from otcom or videoplay and displays the video images and parameters which are sent to it. Originally intended to run on a Sparc2 at the same time as the CCD control program, it tries to be as efficient as possible and is written at the Xlib level. The socket communications is set up to be non-blocking (so that otcom can never hang waiting for the display) and uses a handshake to tell otcom when xoptic is ready to accept a frame.

Xoptic has a lot of controls which are non-obvious because I didn't want to use up a lot of display real estate with more intuitive graphical cues, and in keeping with its lightweight nature, will only accept a limited number of X events per iteration. Nevertheless it can be quite adaptable to the user.

There are three windows across the top. The left shows the incoming video frames. There are always four, corresponding to the four CCD guide regions. Even a guide star is not being read out in a region, the amplifier is

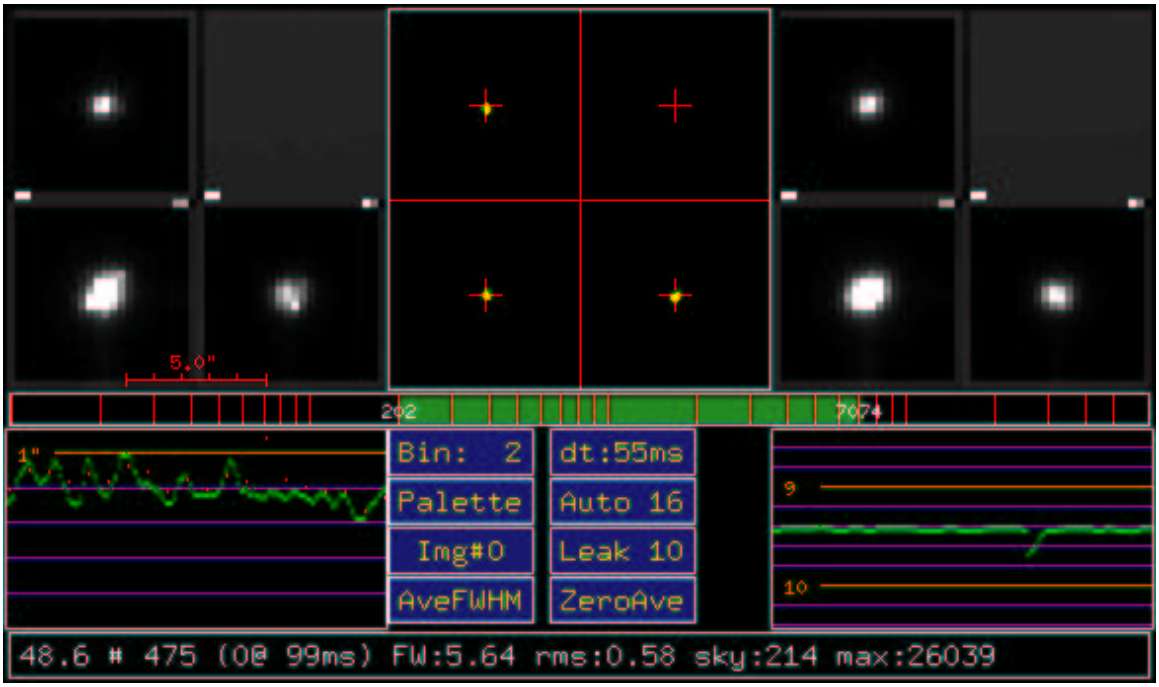


Figure 4-10: The XCCDVideo window from xoptic showing guide stars.

still providing a readout and the noise is displayed by xoptic. The normal zoom factor for xoptic is 4, i.e. four screen pixels correspond to one (binned) CCD pixel. There is a scale of arcseconds shown in the lower part of this image, which should be correct if the `tel_config` file is correct (`otcom`) or `SECPIX?` parameters in the FITS header (videoplay). Pixels greater than 255*255 are displayed as bright green. The stars should normally be more or less centered in the boxes, since otcom normally moves the guide boxes to follow the stars.

The middle image is the "radar" display. The left images show the guide stars in their guide boxes, but the guide boxes can move all over the CCD if the telescope guiding is not good. This radar display shows the center of each star on the CCD relative to the starting point, and can move quite far compared to the star images in the guide boxes. Clicking the left button on this window will recenter the dots relative to their position at that moment.

The right image is a leaky average of the incoming video. The leak

time can be adjusted using the "Leak" button below (the left mouse button decreases the number, the right button increases, and the middle button sets it to 0 which means don't update the display). The leaky average can be rezeroed using the "ZeroAve" button below.

Below the image windows is the "stretch bar". This is a logarithmic graph from 20 to 65535. When the display is autoscaling you will see a green bar fluctuating back and forth with numbers at either end labeled. This is the threshold and saturation for the display. If you click the left button on the stretch bar you will fix the threshold value at that value and the saturation where ever it happened to be. If you click the right button you will set the saturation value. Putting the saturation below the threshold is OK even though the green bar goes wrong – it just inverts the color map.

The bottom half of the display has some buttons in the middle and two strip charts on the left and right. The "Bin" button displays the current bin factor and the "dt:" button the frame time (they used to permit you to control it, and may do it again someday). The "Palette" button cycles through a B/W, inverse B/W, and rainbow palette. The "Auto" button indicates whether autoscaling is on. Left button decreases the contrast, right button increases.

The left strip chart shows the seeing FWHM as a function of time. The image for which this is being displayed is controlled by the "Img" button, left button decreases, right button increases, orientation of 0-3 is the usual. A FWHM of 0 is at the bottom, the left button decreases the number of ledger lines (decreases the FWHM at the top of the scale), and the right button increases the number of ledger lines (increases the FWHM at the top of the scale). The middle button cycles through a display in units of arcsec, arcsec and pixels, and pixels.

The FWHM samples are shown in green and by default are smoothed with other samples. If you want to see the raw samples, the "AveFWHM" button toggles between raw and smoothed. There are also occasional little

red ticks which are the FWHM being calculated for the leaky average, every leak time. They generally track the green points but are slightly higher, as you might expect.

The strip chart on the right shows the magnitude of the selected star as a function of time. As always, left button decreases the number of ledger lines (full scale is fewer magnitudes), and right button increases the number (full scale is more magnitudes). Clicking the middle button near the top of the window pushes the scale downward (moves the scale to lower magnitudes) and clicking the middle button near the bottom of the window pushes the scale upwards (moves the scale to higher magnitudes).

When run from otcom, the magnitudes are based on a zero point provided in the configuration file `filters.def`, and should be reasonably accurate, although no account is taken of airmass. Clouds are dramatically apparent. When run from videoplay, the magnitude zeropoint is simply taken to be 25, although in principle it could be obtained from the FITS header entry `PHOTOZP` written by otcom.

The bottom line lists several numbers, in a slightly different format depending on the size of the window. In the small size the labels displayed across the bottom of the xoptic window are:

```
Integration time
# Frame number
( frames_dropped frame_time )
FW: FWHM of the selected star (unbinned pixels)
rms: rms motion of recent radar positions (unbinned pixels)
sky: sky level
max: maximum pixel
```

5 Data backup

You can back up your data onto CD or DVD. A single night's data will almost certainly compress to less than the 4.7Gb capacity of a DVD, probably somewhere around 2–3 Gb. This will fit on one DVDR or on ~ 4 CDRs. Be sure to use DVD−R media (not DVD+R), which is available in $1\times$, $2\times$, and $4\times$ write speeds. You can also write DVD−RW, but it's more expensive. Burning a night's data onto CDR will take 30-60 minutes, and a similar amount of time onto a $1\times$ DVDR.

The easiest way to burn your data is to use `xcdroast`, a rather nice GUI driven interface to mkisofs and cdrecord.

From the command line execute:

```
xcdroast
```

From the X-CD-Roast main menu,

```
click the "Create CD/DVD" button
```

From the "Create CD" window:

```
click the "Master Tracks" button on the left
```

You should see under the "Devices-Setup" box at the top

```
Image Directory: Automatic
Write Device: SONY RW DW-U10A [0,0,0]
```

(If you don't see this something is really badly wrong, so don't expect things to work properly.)

There are five tabs just under the Devices-Setup box at the top. They are:

```
Master Source        - this is where you select the files to write
ISO9660 options      - options, don't mess with them
Boot options         - you can make a bootable CD, don't mess with it
ISO9660 header       - you can add a header to your disk, not necessary
Create session/image - this is where you create the CD/DVD
```

Go to the "Master source" tab. The left hand box "Session view" shows the files you have selected. The right hand box "File/Directory view" is how you select them. In the "File/Directory view" box:

1. navigate the directory tree to a directory you want to include and click on it
2. click 'Add' and a pop-up appears...
3. select a path option ('Add with last path component' is recommended)
4. click 'Ok'
5. repeat as desired

Most of the rest of the buttons on this window are pretty much self explanatory.

Now click on the "Create session/image" tab. You should see the size of your selection appear in the "Session information" box in the upper left. It may take a few seconds, but should happen automatically if the "Calculate automatically" box is checked.

At this point you can either

Make the image and write it to the CD/DVD on the fly.

Write a disk image of your directories and then write the disk image to CD/DVD.

Generally speaking I prefer to "Write session on-the-fly", but you must be careful that the computer is not loaded down when you do it. In particular, it is **strongly** recommended that you not take data while trying to burn a

DVD. Both otcom and xcdroast will compete at times for the full attention of the CPU, xcdroast will lose, and your DVD will make an attractive (but expensive) coaster.

Assuming that you want to do write on the fly, you need to tell xcdroast how large your disk is. Go to the "Disc Type" menu and select the size of your disk. For a CD it will either be 74 or 80 min, for a DVD it will be 4.7 Gb.

There are some options for writing on the fly, most of which you do not need to change. You can, however, click the "Simulation write" and xcdroast will go through all the motions of writing the disk, but will never turn on the laser to actually burn the disk. If you want to see what will happen but you don't want to waste media, this is a good thing to try.

At the bottom of the "Write session on-the-fly" window there are buttons to blank RW media if desired, and finally a button labeled "Master and write on-the-fly". Click it and you are on your way. To reiterate: don't load the computer while it is burning a CD or DVD. The burners must have a continuous, real-time stream of data and if they are interrupted the disk is ruined.

If you chose to create a disk image (perhaps because you want to burn multiple copies or you are worried about CPU loading or because you want to use the "Verify Tracks" option) and you are ready to write it to CD/DVD, you can now visit the "Write Tracks" button on the left hand side to do this. This window will permit you to specify the image you want to write and offers similar options to the "Write session on-the-fly" for write parameters, etc.

If you want to burn a CD instead of a DVD, you can use xcdroast or you can execute directly from the command line:

```
mkisofs -R -J MY\_DIR | cdrecord -v fs=6m speed=8 dev=0,0 -
```

As above, don't try do this if you are loading the CPU, especially taking data! This will write the entire directory `MY_DIR` onto the CDR. Note that these will make `MY_DIR` the root of the CDR, i.e. a file `MY_DIR/myfile` will appear as `/mnt/cdrom/myfile` when you have mounted the burned CD.

6 Utilities

The source code for these utilities is found in `src/*`, `src/otcom/Utils`, and `src/otcom/xvideo`).

6.1 mkfinder

mkfinder is a script which makes a finding chart from the DSS which has an OPTIC footprint drawn on it. You invoke mkfinder with three arguments: an identifier name, RA, and Dec in a colon-separated format, J2000, e.g.

```
mkfinder n2419 07:38:08.5 +38:52:55
```

It will write several files starting with `Obj_name`, and the final postscript is called `Obj_name.ps`. If you are intending to use OPTIC for guiding/tracking, it is a very good idea to know where stars which are bright enough for guiding are located, it can be very frustrating to do a blind search.

If by any chance you are using an IP address which does not agree with your hostname (i.e. a nameserver will return an IP address for `‘hostname‘` which is not the address you are using), you can force mkfinder to use a difference hostname with the `-hostname` option:

```
mkfinder n2419 07:38:08.5 +38:52:55 -hostname myreal.ip.address.edu
```

If you are observing on a telescope with three reflections so that OPTIC has an inverted parity with respect to normal (i.e. W ccw of N), you can make a parity-flipped finding chart with `-parity -1`:

```
mkfinder n2419 07:38:08.5 +38:52:55 -parity -1
```

6.2 othead

The utility othead will look up header information from an otcom image (not video) file and write it as a single line. Without an argument it just dumps out a title line. Othead can also take a flag `psf` which causes it to run `videostack` if the image was taken in OT mode and has a video file. Vista is then run on the resulting stack of psf images and othead reports the FWHM of each. This is a good way to get a log of which regions were used for guiding and the seeing in each. Typical usage in a directory where you have written data might be:

```
othead > mylog
foreach i (myprefix.???)
  othead $i psf >> mylog
end
```

6.3 fitstile

Fitstile will read a 3D FITS file and write a 2D FITS file which has the individual frames laid out in a tile, starting at the lower left and working right and up. The syntax is

```
fitstile  input_3d_file  output_2d_tile  [options]
```

and options include

```
-first  N  first frame to use
-last   N  last frame to use
-ntile  N  number of frames to use
-sky    S  subtract this from final image
-book      arrange tiles as characters in a book: upper left to lower right
-grid      draw grid lines between tiles
```

6.4 videoplay / xoptic

Videoplay reads a 3D FITS video file and replays it on the screen. By default it reads the frame time from the header and tries to display the images at the same rate they were taken.

```
videoplay prefix_vid.NNN
```

There are optional arguments which can be used. The frame time can be set to a different value using `-t`. The xoptic display can show fewer than the normal 2×2 windows by using `-nx` and `-ny` to specify the number of panes. The data from the video file can be assigned to different windows using the `-w?` arguments.

```
-t T    Set the frame time to T
-nx N   Display N panes in x
-ny N   Display N panes in y
-w0 N   Put region N in window 0
-w1 N   Put region N in window 1
-w2 N   Put region N in window 2
-w3 N   Put region N in window 3
```

For example, if you want to display only the lower right image from a video frame on a display which has only one window (number 0) you could do it as follows:

```
videoplay prefix_vid.NNN -nx 1 -ny 1 -w0 2
```

Videoplay, like otcom, forks off a child process which runs the program xoptic. This in principle can be on a different machine, but at present is hardwired to run locally. Because X is so wonderfully inefficient at passing data back and forth, xoptic does all the X display (coded in Xlib for historical reasons and efficiency), and it receives a fairly compressed data stream from videoplay (or from otcom).

6.5 conflat

Conflat is an extremely important utility which convolves an unshifted (stare mode) flatfield in accordance with the shifts which were performed for an OT tracked image, and writes a new flatfield which should flatten the OT tracked image. The result of OT shifting an image is that each pixel in the final image has actually been integrated on a variety of physical pixels. An exposure during which OT shifting has taken place normally writes an auxiliary file called

```
myprefix_ot.??? - table of where each pixel was integrated
```

Conflat takes an unshifted, unbinned flatfield and convolves it according to the data in the `prefix_ot` file so that it agrees with the OT convolution which took place in the image file. The basic syntax is:

```
conflat flat_field ot_file out_file
```

For example, suppose that `f.115` was a stare mode flatfield in the same filter as an OT tracked observation `f.087`. You could make a flatfield for observation 087 by:

```
conflat f.115 f_ot.087 flat_115.087
```

There are a few things you should beware of with the current conflat.

(a) It assumes that it has short data (not floating point!)

(b) It assumes that the flatfield format is a standard, unbinned readout – the bias strip must be present.

(c) It wants data which is less than 32k (which you want as well for flatfields)

These are easy restrictions to lift, but if you get garbage results from conflat stop and think whether you have written short integer files or whether you have done some processing on the flatfields which has subtracted bias or rewritten in floating format.

6.6 conflat2

Conflat2 is an update of conflat, invoked the same way. It offers a few advantages and no disadvantages, so there is really no reason to use conflat:

1. The data can be any format, floating point is OK
2. The default is to *not* subtract bias. If you want it to subtract bias, you need to use a `-b` flag at the end.

Conflat2 is intended for use when you want to combine a number of images into a master flatfield. You do need to keep the bias strip and the original, full, unbinned size, but conflat2 allows you to have done the bias subtraction and to have changed to floating format.

There is a subtlety with conflat and conflat2 which is important to understand. A hot pixel in an image causes all pixels above it to be ruined when they are read out past it, causing what's known as a "hot column". There is nothing wrong with the pixels above, they just are always messed up by the readout procedure. If you do a readout of a flatfield you will get a hot column above the pixel. If you do an OT convolved observation and read it out, you will get an OT-smeared blob around the hot pixel and the same hot column. The problem then arises if you want to convolve the flatfield to match the image. The hot column in the flatfield then blurs out to a wide bar which does not match the data any more.

The most expeditious thing to do in this case is to replace the hot column in the flatfield with an interpolation before the convolution, and then mask out the remaining hot column in the data after flattening by the convolved flatfield. The most correct thing is more complicated:

1. Take a normal flatfield
2. Take another flatfield exposure Before readout shift it right by 10 pixels (`move offset 10 0`) Read out the image, yielding an "L" shaped hot column
3. Extract the line of pixels which are hidden by the hot column from this second image – it's the line above the tip of the "L", 10 pixels to the right of where the bad column appears.
4. Replace the pixels in the original flatfield with this new line

6.7 Vista

The source is found in `src/vista` and `src/mongo2k`:

Vista is a handy means of displaying images if nothing else, and it is used as part of the fgs routine. There is a section in the appendix about simple usage.

6.8 videostack

Videostack is a utility to read a 3D FITS file, stack all the frames, and write a 2D FITS file of the resulting average. It assumes unsigned short images. The syntax is

```
videostack input_3d_file  output_2d_average  [silent]
```

6.9 videosum

Videosum reads a 3D FITS video file (`prefix_vid.nnn`) and writes an output file which is supposed to be identical to the guide star file (`prefix_gs.nnn`) already written by otcom. Good for checking for bugs in the psf routine and otcom. Syntax:

```
videosum input_3d_file
```

7 Star tracking, OT shifting, and prediction

The basic idea behind OPTIC's operation is that it is tracking 1–4 stars found the "lower" CCD regions via "shutterless video" while the "upper" parts of the CCDs are accumulating a science image. Shift corrections are applied to the science regions during the exposure to compensate for image motion. There are therefore a number of related but different functions which must be carried out for this to work:

1. Guide stars have to be tracked at high speed
2. Guide star positions have to be analyzed
3. Temporal predictions need to be made for where the guide stars will be at the next iteration.
4. Spatial predictions need to be made for how to translate guide star position in the lower regions to image motion in the CCD regions.

These will be discussed in turn, with emphasis on how the user can influence and optimize the performance.

7.1 Tracking guide stars

Once a guide star has been identified by its location, otcom can read out a small box of CCD containing the star. It rapidly clocks the lower, guide region of the CCD horizontally to bring the patch to the xoffset requested by the `sgformat`, then rapidly clocks it vertically to the yoffset, then does a normal readout with the requested binning and size. All of this is done while leaving the rest of the array quiescent.

The parallel clocking presently occurs at a rate of about 16 μsec per pixel, so there will be a small horizontal smear on one side of the star image left behind by the guide star (or any other source in the field, but generally the guide star is the brightest thing present) as the pixels are clocked by.

After the patch is read out, a new clean patch is clocked down under the star for the next integration, and this will leave a small vertical smear on one side of the star. As an example, if the box size is 16×16, binned 2×2, the half width of 16 pixels will take approximately 250 μsec to move out from under the star and so the integrated light in the smear should correspond to approximately 250 μsec of exposure.

The DSP code in otcom is quite general, but does optimize the shutterless video clocking strategy when more than one star is being read out. Since the read out is done in parallel, all four amplifiers are read out, although any lower region which is being used for science will not have any charge shifted to the amplifier so the image which otcom receives is just a bias.

This clocking time of 16 μsec per pixel can get quite long if the guide star is found on the opposite side of the chip from an amplifier – as much as 40 msec. This clocking time and the readout time of about 6 μsec per pixel are overheads which are lost to guide star integration. The overheads can be as small as 10 msec for small patches found close to the amplifiers or as large as 40 msec for large patches which are a long way from the amplifiers.

Note that although there are no restrictions placed on guide star box size or binning, if the box encounters the edge of the lower CCD region you will get a truncated image. Xoptic is presently compiled with a limit of 64×64 on guide images, so that is a practical limit to the guide image size, although since the binning is arbitrary it is possible to read out fairly large regions rapidly.

Because the guide regions are being read out so rapidly and are so small, the "bias level" is not very flat – it has a component with an exponential decay away from the amplifier with an e-folding of perhaps 20–50 pixels. Also because of the desire for speed it is not possible to get a bias level in the usual way by emptying the serial register. Therefore if otcom is requested to do bias subtraction in the `tparams`, it starts each exposure with a few iterations, analyzes them in terms of this constant+exponential model for bias, and then

uses these parameters later for bias subtraction. You may note that the edge pixels of the guide box are not bias subtracted – they are generally corrupted for various other reasons and are not used. Also it is possible to get a bad fit because of an incompletely erased CCD or a very bright background seen when the video bias is taken. If this happens, consider turning off the bias subtraction.

7.2 Analyzing guide stars and temporal predictions

Otcom does quite a careful analysis of the guide star images, although it could be improved quite a bit. If bias subtraction is requested, the bias is first subtracted. A sky level is determined in a robust way from the four corners (ignoring edge pixels) as the second largest value. The highest pixels are then located and a half width (HWHM) estimated in x and y by running down from the highest pixel. Two marginal cuts through the image are extracted in x and y by summing around the highest pixel to ± 1 HWHM. A Gaussian is fitted in each direction which provides a center and FWHM in each direction. Finally the signal is added up within ± 2.5 FWHM of the center to provide a total flux for the star.

Once the star's positions are known it is necessary to make some sort of prediction where they will be at the next iteration. This has two implications – the guide boxes are small enough that we need to follow the guide stars fairly accurately if we are not to lose them, and the accuracy of our prediction influences how well we will do in removing image motion in the science regions. These have somewhat different requirements, in that we need to be quite conservative in the guide star location prediction because we can tolerate a many pixel error and not lose the star, but if we lose the star the exposure is ruined.

The standard temporal prediction method simply uses the most recent location as the prediction. Another possibility offered is a bit of code which makes a prediction based on the previous N observations. The idea is to

find linear coefficients which multiply the previous N to accurately give the present value. Clearly for uncorrelated motions the coefficients should be zero, but for constant, sinusoidal motions (telescope shake) the previous three points could give a very good prediction. If requested, otcom goes through the previous time history of positions and does a fit for such coefficients and then applies them to the present motion. The coefficients get updated every iteration, so the predictions for sinusoidal motion, for example, get better as data collect. Also this has the ability to adapt to changing conditions such as phase, frequency, or amplitude of motion.

The predictions must now be used to calculate how to move the guide boxes to follow the stars. The normal mode is to use the median shift of all active guide stars (plus a shift of 0 thrown to the median as a vote for conservatism), so as to have some immunity to losing a guide star because of a cosmic ray. This has the drawback that the guide boxes are all locked together so if a star is not centered after the initial tweak-up of guide box positions it will stay uncentered. Two other possibilities are to simply take the predictions at face value, but it is very easy to lose a guide star this way. Another possibility is to use the previous positions as well, and an option can be provided via `tparam` to use a median of all guide stars and the three previous iterations as well.

The maximum guide box shift permitted in an interaction is also limited by the "Max" parameter of `tparam`, and the ratio of the indicated shift to the actual shift is governed by "Gain". In cases (such as when the UH2.2-m dome rotates) where violent image motions occur, these may be used to maintain lock on the guide stars even though they may be lost for a few frames.

It would be good to use some sort of signal to noise information in the guide star analysis, but although the psf analysis returns a signal to noise for each star, this is presently not used for guide decisions.

7.3 Spatial predictions

The guide stars are not cospatial with the science regions, so it is an interesting question, given temporal predictions of where the guide stars will lie, to guess where the images will shift in the science regions. Otcom provides a single method at present to do this. Each science region gets a weight from each active guide star. This weight depends on the distance of the center of the science region from the guide star, and can be varied by use of an "exponent" in `tparam`, where the relative weights go as the inverse distance to the "exponent" power. So for example, and exponent of 0.01 is nearly a uniform weight of all guide stars. An exponent of 1 is weighting as inverse distance, and 2 weights as inverse distance squared.

If you simply want to use the nearest guide star you could use a huge exponent, or this special case is also obtained with a negative exponent, nominally -1.

We have done some experiments at the UH2.2-m to look for temporal correlations between different guide stars, which would indicate coherent turbulence blowing across the telescope pupil, and on some occasions we see such correlations at a reasonable timescale (of order 1 sec). However, they are at very low level compared to the overall image motion at the 2.2-m, so I've never had an opportunity to try to exploit temporal correlations to use the time history of guide star positions as well as their spatial locations to make science shift predictions. (The UH2.2-m mirror normally is at +2.5C, the closed telescope tube at −1.5C and ambient at −2C. The vast majority of seeing and image motion is usually generated within the telescope tube so our task of removing image motion is very simple: everything moves together.)

Although there is a lot of censorship on the use of guide star positions for guide box motions, there is presently no censorship on how the guide star positions get used for shifting the science images. There should be.

7.4 Summary of guide/shift iterations

An iteration consists of the following steps in host and DSP:

1. Issue OTF with 16 args which are the shifts applied to the 8 regions for OT tracking, and mask/shifts used on the 4 guide regions to align the guide stars for readout.
2. Host commences non-blocking DMA read on the guide regions, ships off the previous frame to the video display, occasionally issues telescope guide commands, then waits...
3. DSP immediately applies shifts to indicated subset of 8 science regions
4. DSP waits for gstime
5. DSP applies OT shifts to guide regions to bring them into alignment for a subsequent readout using ccd.G parameters.
6. DSP does normal readout of guide regions using guide parameters
7. DSP clocks down a clean bit of CCD to the guide regions
8. Host receives guide image, calculates actual GS exposure time (which is the full iteration time minus the time spent reading out)
9. Host inserts time and location stamps into 3 trailing dead pixels
10. Host unscrambles guide data
11. Host subtracts a bias from video frames
12. Host analyzes guide frames for guide star centroids, etc.
13. Host predicts where the stars will be next iteration
14. Host calculates shifts for guide boxes to follow guide stars
15. Host calculates OT shifts to apply to science regions

7.5 Definition of the algorithm coding

The algorithm field uses four decimal digits for four phases of OT tracking.

```
  alg = wxyz

**** 1's digit (z): Predict new GS positions from previous positions
  case 0: New = most recent
  case 2-9: Find coefficients among previous observations such that an
```

```
            observation is a weighted average of N=2-9 previous.  Apply
            these coefficients to the N most current observations.
            (N=1 is perforce identical to case 0.)

*** 10's digit (y): Determine guide box offsets from GS positions
  case 0: Take GS prediction at face value, shift box to match
  case 1: Use a median of current GS predictions (and a vote of zero
          shift as well), apply to each
  case 2: Use a median of current and previous 3 GS predictions (and
          a vote of zero shift as well), apply to each

** 100's digit (x): Choose offsets from science regions from GS positions
  case 0: Use (1/d)^gsexp from guide star as a weight (default gsexp = 2)
          If gxexp < 0, track closest guide star

* 1000's digit (w): Determine how a PSF centroid is determined
  case 0: Use quadratic fit to FWHM of brightest pixel
  case 1: Use brightest pixel
```

The default is `alg=10` which simply takes the previous iteration as the new location, conservatively uses the median of the guide stars to choose the new guide box locations (so a single bad star image won't lose the star entirely), and fits the PSF instead of centering up on the brightest pixel.

8 Appendix

8.1 Remote observing

Let's assume that you are sitting at a computer called "tinsley" and you want to observe with OPTIC on "mkbill". A few quick notes:

There are four communications channels between you and otcom

1. otcom itself puts out and accepts text
2. otcom displays images via fgs which are used for guide star selection
3. otcom displays video via the xoptic program
4. otgui provides a GUI which communicates with otcom

X is inefficient! You can have mkbill do the graphics computation locally and then have mkbill's X server communicate with tinsley's X server, and tinsley display the data, but this carries a huge penalty for the video communications. The better alternative is to have otcom ship a small amount of video data directly to tinsley and have tinsley process it into a graphical display by running xoptic locally on tinsley. This requires more setup on tinsely, of course.

Similar considerations suggest that you do not run imaging (Vista, ds9, ximtool, etc) on mkbill with display on tinsley. Copy the file first and run your imaging program locally on tinsley.

8.1.1 The right way

This runs xoptic and otgui on tinsley, so X is only used for the fgs display, which is not time critical. You need to have xoptic compiled on tinsley and otgui able to run on tinsley.

First of all, follow the instructions below for installing xoptic on tinsley, and make sure that you have an obs account with a `.rhosts` file which

permits access by mkbill.

Next check that you have Tk/TCL on tinsley by typing `which wish`. Note the path where it is found. If you do not have Tk/TCL on your machine, get it from a standard distribution.

Copy the otgui stuff from `mkbill:/usr/local/inst/gui` into a directory and edit `otgui.tcl` to reflect

a) top line must reflect the path of where wish is found on your machine
b) the `cd /usr/local/inst/gui` found around line 87 must be changed to wherever you have put the TCL files.

When you are ready to observe, tunnel through the firewall to mkbill using:

```
tinsley% xhost mkbill.ifa.hawaii.edu localhost
```
(permit mkbill and obs@tinsley to display on tinsley)
```
tinsley% ssh -l obs -L9511:127.0.0.1:9511 mkbill.ifa.hawaii.edu
```
(tunnel through the firewall using the otcom port number)
```
mkbill% setenv VIDEOHOST tinsley.ifa.hawaii.edu
```
(tell mkbill where to send the video)
```
mkbill% otcom
```
(start otcom; do **NOT** background this)
```
tinsley% otgui -host 127.0.0.1 -id ``My_name in Manoa''
```
(start up a local otgui which communicates with otcom)

8.1.2 The quick and dirty way (pretty much OK)

This runs xoptic on tinsley, so X is not used for the video communications between mkbill and tinsley. It's more efficient, but you need to have xoptic compiled on tinsley.

Once this has been done, you initiate an otcom session as follows:
```
tinsley% xhost mkbill.ifa.hawaii.edu
```

(permit mkbill to display on tinsley)

```
tinsley% ssh mkbill.ifa.hawaii.edu -l obs
```

(open a connection to mkbill)

```
mkbill% setenv DISPLAY tinsley.ifa.hawaii.edu:0.0
```

(tell mkbill where to display the fgs images)

```
mkbill% setenv VIDEOHOST tinsley.ifa.hawaii.edu
```

(tell mkbill where to run the xoptic video)

```
mkbill% otcom
```

(start otcom in this window, NOTE: it is a big error to background this!)

```
tinsley% ssh mkbill.ifa.hawaii.edu -l obs
```

(open a second connection to mkbill)

```
mkbill% setenv DISPLAY tinsley.ifa.hawaii.edu:0.0
```

(tell mkbill where to display the otgui windows)

```
mkbill% otgui &
```

(start up the GUI)

8.1.3 The extremely quick and dirty way (not advised)

This runs everything on mkbill and uses X for all communications back to tinsley. Use it **only** when you cannot compile xoptic on tinsley.

The problem with running otcom this way is that mkbill does the graphics locally by running otgui and especially xoptic, and then requests mkbill's X server to ship the graphics to tinsley's X server. The result is then displayed in front of you, but the amount of traffic is orders of magnitude bigger than if you let otcom ship just the data from mkbill to tinsley, and then let tinsley do the graphical computation as well as display. Also, otcom has a vulnerability which can make its video very slow if you are using a video box larger than about 20×20. Be prepared for dreadful performance with the xoptic video if you do this, and do use a small video box size!

```
tinsley% ssh mkbill.ifa.hawaii.edu -l obs
```

(open a connection to mkbill)

```
mkbill% setenv VIDEODISPLAY $DISPLAY
```

(tell otcom where to show video)

```
mkbill% otcom
```

(start otcom in this window, NOTE: it is a big error to background this!)

```
tinsley% ssh mkbill.ifa.hawaii.edu -l obs
```

(open another connection to mkbill)

```
tinsley% xhost mkbill.ifa.hawaii.edu
```

(permit X from mkbill to display on tinsley)

```
mkbill% otgui &
```

(start the GUI on mkbill, displaying on tinsley)

8.1.4 Testing it out

A useful thing to know is that you can try things out without running otcom. There is a program called `otsim` on mkbill which is otcom without any camera connection so it's safe to run. Start up otsim on mkbill in place of otcom and you can check out that all the communications are working properly. Here's the things to look for:

1. Can you start otsim (otcom)?
2. Can you get otgui started, communicating; can you download and power up?
3. Do an `fgs` and use `f` to mark 4 stars
4. Choose a 10 second exposure time and "OT" mode, and hit "GO". Do you get a video window with reasonable numbers and graphs? If not, there's a possibility that you did not compile xoptic with the right byte swap option.

8.1.5 Installing xoptic

If you want to compile xoptic on a different own machine, get the source from `mkbill:/usr/local/inst/src/otcom`, change the `Makefile` according to your machine architecture, and type "make xoptic".

You also need to make an entry for xoptic in your `.Xdefaults` file so that the xoptic display won't hang and wait for you to manually place it:

```
xoptic*geometry:                -3+3
```

You also need to permit obs to do an rsh from mkbill to tinsley, so you need to create a `.rhosts` file with the contents:

```
mkbill.ifa.hawaii.edu obs
```

Note that this file should be read protected against everybody but obs:

```
chmod 600 .rhosts
```

Be sure that you have tested that this works by trying

```
mkbill% rsh tinsley.ifa.hawaii.edu which xoptic
```

8.1.6 Getting rsh on a Linux box

1. Install rdist, rsh-server, rsync from up2date or http:rpmfind.net
2. Add rsh to the end of the file /etc/securetty
3. Change "disable = yes" to "disable = no" in the files rexec, rlogin, rsh, rsync found in /etc/xinetd.d

8.1.7 Copying the otcom distribution

```
> ssh obs@mkbill
$ cd /usr/local/
$ scp -pr inst/{bin,doc,gui,src} obs@tinsley:/tmp
$ exit
> su
# mv /tmp/inst /usr/local/
```

8.2 The OPTIC simulator

There is a server version called "otsim" which is identical to otcom except that it does not connect to the interface or camera at all, but simulates their effects. This is extremely handy for testing software and for gaining familiarity with otcom and otgui because otsim and otgui can be compiled and run on any Linux system.

You need the usual directory tree in `/usr/local/inst`, and don't forget to turn off the telescope by commenting out "TELESCOPE" from `/usr/local/inst/config/tel_conf`

By default otsim creates its own idealized sky. However, you can get it to use any FITS file (perhaps from mkfinder) using the syntax

```
otsim -interface fits_file
```

This can be helpful for determining where guide stars may be found. In this mode otsim uses the actual scale in the FITS image and assumes 0.138″ pixels for OPTIC. Rotations are not implemented (yet), so if you want a rotated image you need to rotate the FITS file (e.g. the Vista rotate command).

There are lots of other options, which are given to the otsim interface via the interface as a single argument e.g.

```
otsim -interface "lots of args"
```

These include

```
-file fits_name       /* Use a fits file instead of internal fake image */
-psf p                /* psf (arcsec) */
-sky sky              /* Extra sky level */
-bin N                /* Binning factor for internal image (speed up) */
```

```
-eccwn {0|1}          /* parity */
-pa x                 /* PA of top of image */
-dx x                 /* x offset of OPTIC wrt FITS image */
-dy x                 /* y offset of OPTIC wrt FITS image */
-scale s              /* Arcsec / pixel for OPTIC */
-telshake x           /* Telescope shake sigma (unbinned pix) */
-telleak N            /* Telescope leak time (iterations) */
-atmshake x           /* Atmosphere shake sigma (unbinned pix) */
-atmleak N            /* Atmosphere leak time (iterations) */
-accum                /* Accumulate (slow, accurate) instead of sample */
```

Typical usages include:

```
./otsim -interface "-bin 2 -accum"
```

This is fairly slow but allows us to see the effects of OT shifting.

```
./otsim -interface "-bin 2 -telshake 0 -atmshake 0 -accum"
```

This cuts the image motion to zero to which is helpful for testing deliberate motions such as the `drift` command.

8.3 Vista

8.3.1 Introduction to Vista

Vista is one of a number of "do it all" astronomy reduction programs such as AIPS or Figaro or IRAF. It differs somewhat from reduction suites such as imcat which use the Unix shell as glue between a toolkit of utilities, but the fundamental capabilities are similar. There are at least two versions of Vista in use, "LickVista" or the "One and True Vista", and a version of "Vista" which split off in 1986 and which I have evolved forward. Unfortunately the

schism has become quite deep and it is no longer straightforward to bridge the gap. Equally unfortunate is the capital "V" in Vista, but it has become a historical artifact which can not be changed.

There are two basic ways to use Vista: interactively or as a shell command. In the former mode you type things at Vista and it executes them. In the latter mode you type "Vista" followed by the name of a procedure and (perhaps) some other arguments and it does the work specified and terminates. Since the shell command mode is essentially identical to interactive mode, with the procedure providing the input commands, I'll describe the former mode first.

First some Vista basics. Vista maintains 32 image buffers into which data can be read, referred to by their numbers: 1–32. All images are taken to be two dimensional, even though the dimensions may be $N \times 1$. Vista maintains a set of numerical variables and string variables which can be used in place of literal arguments and which can be used for arithmetic calculations. Vista can execute lists of commands from disk files, called procedures. These procedures allow branching constructs and can be nested, but unfortunately do not offer any means of passing arguments. All interprocess communication must be via variables (which all have global scope).

The basic syntax of a Vista command is a command, followed by arguments, followed by modifiers. The arguments depend on the command and various commands may have different actions depending on the number of arguments present. The modifiers are all of the form "KEYWORD=VALUE" where the KEYWORD is an adverb specific to the command, the "=" marks the end of the adverb, and the VALUE can be a variety of possibilities. There is no whitespace allowed in a "KEYWORD=VALUE" combination unless it is quoted with single quotes.

When Vista is started it responds with a `>>` prompt indicating that it is ready to accept commands. A few general comments:

`q` on the command line is how Vista is terminated (or ^C if you must)

`e` in the display window is how you get out of a video-interactive routine such as "ITV" or "PSF"

`?` in the display will give you a menu during a video-interactive routine

8.3.2 Tutorial

Before delving into a lot of details about different commands and arguments, here is a quick tutorial on the three most important commands used for quick look image analysis: `rd`, `tv`, and `psf`.

The `rd` command reads a FITS file into a Vista buffer. This example uses buffer 1, but there is no reason to prefer one over another, and buffers remain intact until overwritten. The second argument is the filename. Vista uses the usual pathnames from UNIX, so a filename of `m0493.235` would have been adequate if we had started Vista in the directory holding the image file.

(NOTE: A filename without a decimal decimal point, "." will lead Vista to append ".`fits`": "`filename.fits`". If you have a filename without an extension, you need to rename it.)

The routine can distinguish different types of FITS files (short integer, floating point, long integer), but it can't always distinguish signed from unsigned short integers. Many CCDs provide 16 bits of data which normally shows up in an unsigned 16-bit FITS image. Thus `rd` needs to know what sort of FITS file you have if you are reading a 16-bit FITS image. The keywords are `short` or `unsigned` according to whether the 16-th bit is a sign or a data bit; use these keywords as a third argument to `rd`. Once you have made a specification Vista will continue to use that choice until you read a 16-bit FITS file of the other type, so you need only type `unsigned` once. Vista tries to make a reasonable choice, so chances are that you don't need this keyword at all.

(NOTE: If you see lots of negative numbers in a FITS image or lots of numbers near 65000, you probably used the wrong choice of `short` or

`unsigned`. Reread the image with the other choice.)

The `tv` command has many options, but you need only know a few in order to display images effectively. `tv` requires buffer number as its first argument, and can optionally take threshold value, saturation value, and other options in the "KEY=VALUE" format. Vista takes the smaller of the two stretch values for the threshold and the larger for the saturation.

Suppose we discover that the mean pixel value in buffer 5 is 2809.733. Then we might choose a `tv` threshold of 2700 and a saturation of 3000 as follows: `tv 5 2700 3000`. For a color map which ranges from white to black (the default), the resulting image would be white where the pixels have values of 2700 or less and black where they have values of 3000 or more; intermediate values would be displayed as linearly increasing depths of gray. Vista will automatically determine threshold and saturation values if none is given on the command line. If only one number is given following the buffer number, it will be taken as the saturation, and the threshold will be set to zero.

Some examples of the options to the `tv` command which you might want to explore include:

```
 tv 1                      --- Autoscale and display.
 tv 1 300 2700 noresize    --- Display the image and turn off the automatic
                               resizing of the image display window, so it
                               won't change from the size you like.  You
                               only need "noresize" once.
 tv 1 300 2700  cf=jt      --- Display the image with a rainbow color scheme.
 tv 1 300  2700 cf=bw      --- Display with a normal B/W color scheme.
 color cf=jt               --- Switches to the rainbow color scheme.
 color inv                 --- Invert the color scheme.
 tv 1 1e4 2500. sqrt       --- Display with a square root mapping of data to
u                                colors for pixel values between 2500 (low)
                               and 10000 (high).
 tv 1 10000 2e3 log        --- Display with a log mapping of data to color.
```

```
tv 1 old                  --- Display with the same threshold, saturation
                              and mapping as used last time.
tv 1 3e4 sqrt             --- Display with a square root mapping from a
                              saturation of 30000 to a threshold of 0.
tv 1 old flip             --- Display the image flipped left-right (using
                              the old threshold, saturation, mapping).
<left mouse in image>     --- Zoom out centered on the position of the mouse.
<middle mouse in image>   --- Keep zoom but center on position.
<right mouse in image>    --- Zoom in centered on position.
"@" struck in image       --- Restore usual zoom and centering.
"#" struck in image       --- Zoom way in and display data values.
"[" struck in image       --- Display previous image.  The current image
                              has a special status and old ones are
                              "zombies" which cannot be zoomed, etc, but
                              this can be used to blink up to N (4) images.
"]" struck in image       --- Display subsequent image
"$" struck in image       --- Choose sampling/averaging for zoom out
<deiconify Xzoom window> --- Continuous zoomed in display.
"4" struck in zoom window--- Display four zombie images in zoom window
"n" struck in zoom window--- Display N zombie images in zoom window
"1" struck in zoom window--- Display just current image in zoom window
<left mouse in zoom>      --- Shrink zoomed pixels
<right mouse in zoom>     --- Enlarge zoomed pixels
<left mouse in palette>   --- Change the color stretch
<middle mouse in palette>--- "Roll" the color palette
```

The `psf` command (a.k.a. `fondle` when non-PC) is used to find out about star-like images: their size, shape, and flux, and it also has some plotting options. Psf puts Vista in "input" mode, as indicated by the little status light by the `(x,y,z)` display of the image. In this mode Vista responds to mouse or keyboard events in the display window, but not to commands typed at the interpreter. In order to get back to the interpreter type the

letter `e` in the image window. As always, typing `?` at the image window will list the options available in this mode.

In the example, psf is told to use `scale=0.14`, i.e. the plate scale for OPTIC at the 2.2-m and WIYN, for reporting image widths. Psf will keep this scale until you explicitly change it either by restarting psf with a different `scale=` argument, or by using the `p` window option.

When you put the cursor on a star (reasonably near is OK) and type `f`, psf will report the position of the star (in pixels), the peak intensity (in ADU), the fwhm (in scale*pixels = arcsec), the total flux within the aperture radius (in units of 1e5 ADU), the sky level near the star (in ADU), and the dimensions of the psf. Psf fits a 2-dimensional Gaussian or Waussian (Paul Schechter's parametrization of a psf as the inverse of a truncated Taylor expansion of a Gaussian) to the core of the star image (after determining the sky value), and these dimensions and angle refer to the major and minor fwhm dimensions of the Gaussian fit, along with the azimuth of the major axis CCW from the x axis. If the Gaussian is so round that this azimuth is poorly determined, psf will report it as 0.

If you hit the `g` key you will enable plots of star's profiles when the psf calculation takes place. If you would like to see a cut through a piece of your image, place the cursor and hit the letter `l` at each of the two endpoints of the segment you desire.

Examples:

```
 >> psf scale=0.14
        "/pix=0.140    (") (1.0E+05)                (")    (")
    x      y     peak  fwhm   flux   +/-    sky      maj x  min   phi  Gpeak

(Move to a star and hit "f":)

  811.8 1088.2  15847  0.47  2.209  0.041  4875.9   0.47 x 0.47     0  11335
```

```
(Hit "g"; move to a star and hit "f":)

 Plot mode: radial profile
  961.5  896.0  15589  0.47  2.296  0.046  4905.4   0.47 x 0.47     0  11429

(Hit "g"; move to a star and hit "f":)

 Plot mode: mesh/contour
  884.2  870.0  22637  0.48  3.812  0.039  4908.0   0.48 x 0.48     0  18203

(Hit "g"; move to a star and hit "f":)

 Plot mode: aperture profile
  582.1 1245.8  14391  0.48  1.668  0.036  4870.8   0.48 x 0.48     0   9827

(Place the cursor and hit "l", move it and hit "l" again to plot a cut)

(When you want to get back to Vista, hit "e" when the cursor is in the
  image window.)
```

What's PSF up to?

Psf was designed to help in the difficult task of determining the total flux from a star. When you hit `f`, psf goes through the following chores:

1. Find the nearest peak.
2. Assemble average and median fluxes as a function of radius from the star.
3. Estimate a value for the local sky and the fwhm of the peak.
4. Determine a better value for the sky by fitting the median profile with a r^{-3} profile for the skirts of the star plus a constant sky.

5. Add up the flux of the star using this sky value. The pixels closer to the star than a radius of 10 are added directly; exterior to this radius the star's flux is estimated as the median profile times the number of pixels (which confers substantial immunity to neighboring stars).
6. Perform a 2-D G/Waussian fit to the image to derive a better fwhm.

Psf computes three estimates of the flux of a star. The first is the straight sum of the pixels with the fit sky subtracted. This is obviously subject to contamination by neighbors. The latter two are based on the sum/median profile, the first having the fit sky subtracted and the second having the median sky estimate subtracted. Obviously the first will suffer if the fit to the sky is poor and the second will have problems if the background has curvature (for example near a large galaxy).

In the course of this work psf has garnered a pretty good idea of what the star's profile looks like, which it reports to you during some of its modes. Since psf is designed to work in "input" mode, it seemed like a good idea to add a number of line graphics options to enable you to visualize your data. These are the keys you can use to perform different functions:

```
         "/pix=1.000    (") (1.0E+05)                (")    (")
    x      y     peak  fwhm   flux   +/-    sky      maj x  min   phi  Gpeak
Commands:  (? to repeat help message)
X   Center on star, report info            A   Same as X, remember profile
F   Same as X, sum (pixels-sky), best phot B   Same as X, scale to saved profile
C   Ctr on cursor, sum (pixels-sky) C=save H   Make hardcopy of current plot
S   Ctr on cursor, add ave to sky stats    Z   Reset stats
E   Quit                                   -   Remove last stat

V   Cycle through flux modes:
       psf info / brief flux stats / verbose info on radial profile
G   Cycle through plot style executed with "X", "C", etc:
       no plot / radial profile / 3D plot or contour / aperture flux
l-l Draw a cut between cursor positions (rectilinear if L-L)
```

```
R   Set flux aperture (def r=20)     U   Set flux scale factor (def 1E5)
P   Set plate scale (def 1.0)        W   Set width averaged for cut line (def 1)
T   Cycle Gaussian/ 7/4/2 param Waussian fits

M   Toggle mesh (3D) vs contour plots (def 3D)
I   Toggle interactive plotting (def N)
D   Toggle dumping to fort.13 (def N)
N   Toggle reset of stats when psf is invoked (def Y)
O   Toggle drawing of circular outline (def Y)
```

Statistics

Psf has three levels of detail in its report of a star's flux. The `v` key cycles through the three levels of verbosity, from least to greatest. The most terse mode gives you a fit position, net flux estimate (after sky subtraction) and sky value, and information on the psf dimensions from the 2-dimensional G/Waussian fit. The next mode accessed by `v` dispenses with the psf information and tells you about flux and sky statistics being accumulated. The last level of verbosity gives you a listing of the radial profile of the accumulating flux. In all cases, any peak brightness listed is the actual brightness, but any flux listing has had a sky value subtracted from the pixels summed up.

In the first two modes, the flux listed is that found within a circular aperture whose radius may be specified with the `r` key or as a `rad=aperture_radius` keyword when psf is invoked. The circle drawn on your image shows this aperture box. The flux is reported in a scaled form; you may change this scale factor with the `u` (units) key.

There are five ways to ask for flux information. The first, invoked with the `f` key, finds the nearest peak and does the full analysis of star profile, sky value, etc. The `x` key does nearly the same thing but does a poorer job of summing up total flux. Almost identical is the `a` key, which differs only in

that it "remembers" the profile of that star as a fiducial profile. If you use the `b` key psf will not try to follow the profile in detail, but will try to fit the star as a multiple of the current fiducial profile. This is intended to help dig information out of faint images.

If for some reason you do not want psf to center on a star or do a fit, the `C` key will simply add up the flux in the aperture radius centered on the cursor position and report it (minus the current average sky value). Psf maintains statistics on the fluxes it encounters and the sky values. If you want to add to the sky statistics, you may use the `s` key to add up the flux in the aperture centered on the cursor.

In some instances (such as a focus frame) there will be many images of the same star, and you might want to stomp around the image with `s` to establish a sky value, and then cookie-cut the star images with `C`. In other cases (such as stars near a large galaxy) the different flux and sky values will have nothing to do with another, so the accumulating statistics will be worthless. If you want to reset the statistics (done by default when psf is invoked, unless the `n` key has been used), you may use the `z` key. If you want to remove the last statistic, the `-` key is provided.

If you are familiar with Vista's photometry facility (`save phot`, `print phot`, etc), you will want to know that psf saves the star information, and that you can reset the accumulation with the keyword `new`. Another tidbit for Vista aficionados is that psf does not treat zero pixels as special in any way.

Line Graphics Plotting

Psf offers a number of line graphics options, and it's assumed that you have Mongo set up properly with a terminal window (and possibly a printer). For example, if you want a cut plotted from your image you can place the cursor at a spot in the image, hit the `l` key, place the cursor at another point and again hit the `l` key. Psf will plot a cut extracted from your image. The default is a cut of a single pixel width, but you may use the `w` key to specify

a width across the cut for averaging. If you want precisely the cut indicated by the cursor position use the lower case `l` key. If you prefer to have your cut straightened to the nearest vertical or horizontal, use the upper case `L` key.

You may accompany the psf analysis of star images with plots of the stellar profile. This will happen automatically when you hit `x` (for example) if you have turned on this graphing by hitting the `g` key which cycles through four plotting modes. There are three sorts of plots: radial profile, 3-D mesh plot or contour plot, and cumulative flux plot.

```
          "g"               "g"         "g"                "g"
No plotting => Radial profile => 3-D plot => Cumulative flux => No plotting
                                  \= "m" =\
                                 Contour plot
```

The radial profile plot shows the pixels near the central one, plotted as flux versus radius, and the Gaussian fit is drawn as well. If the Gaussian fitting routine has detected non-zero ellipticity, the major and minor axis profiles are both shown.

The 3-D plotting mode depicts a patch of the image as a mesh plot. If you prefer to see two-dimensional data presented as a contour plot, you may switch between 3-D and contour mode by hitting the `m` (mesh) key. Note that using the `c` key instead of the `f` or `x` key will center the plot where you placed the cursor as opposed to using the nearest peak as a center.

The last variety of plot has two panels. The left panel is intended to illustrate how accurately psf has succeeded in determining the local sky value, and the right panel shows how well the total flux from a star converges as a function of aperture size. The points in the left hand plot are the average star and median star brightness as a function of radius. The curve is a very crude fit to the profile, unless you have used the `b` key to obtain star information, in which case it is a scaled version of your fiducial star profile.

The right panel plots a variety of quantities as a function of radius; it's best to use `verbose` mode in trying to decipher which is which. You will see the cumulative flux determined as a straight sum of flux less the sky contribution, the cumulative flux determined as a median less a fitted sky value, multiplied by the number of pixels in an annulus, and the median cumulative flux less a median sky value from a large radius. Lastly the flux in each annulus is shown as a histogram. These plots are intended to be sobering reminders of how difficult it is to determine a good sky level and a total flux for a star.

If you want to mess around with a plot, you may enter interactive Mongo after one of these plots by using the `i` key to toggle interactive mode. You will be presented with the Mongo prompt `*` in the Vista window, and typing `end` at Mongo will return you to psf.

Should you desire a hardcopy of a plot which has appeared on the terminal, you may hit the `h` key which requests Mongo to redraw the last plot on the laser printer.

Odds and Ends

Hitting `o` toggles the drawing of circles around the star which psf is working on. You may use `d` to have psf dump results to a file called `fort.13`. Like `o`, `d` is a toggle, and their defaults are `on` and `off` respectively. The `n` key is used to toggle between a mode where the statistics are reset upon entering psf (the default) and a mode where you have to reset the statistics explicitly. This might be useful if you had several images of a single star and you wanted to average together statistics.

8.3.3 Examples

With very little explanation, here is a sampling of Vista commands. If these are all familiar to you, you are a true Vista master.

```
! Quit
```

```
q

Vista mypro.pro string2 string3 string4 string5
 => runs Vista through mypro.pro, with string variable argN set to stringN
    if mypro.pro finishes with "end" Vista silently terminates

! Examine various buffers
buf
! Examine various boxes
print box

! Set (numeric) variables
set am=1145 bm=1222
! Snag FITS keywords NAXIS1 from header of buffer dest (variable)
set size={dest:NAXIS1}
! variable arithmetic is forward polish, beware.
set slope=-ydiff/xdiff
typ bias1
! String variables (different name space => can use same names as numeric)
string fnum '178'
! Set a string to token from file myfile, line k, token 2
string FIELD  '@myfile.dat#k#2'
! Evaluate a string as a portion of a command
rd 1 {mystring}

! Write variable info to a file
printf {FIELD}_{PATCH}_{DATE}_{INST}_{FILT}.{DITH}.{CHIP}
printf '%a30 %f8.1 %f8.3 %f8.3' {arg2} sky sens frng
! Execute a shell command
! Note that there cannot be a space between the ">" and the filename!
% ls WARP/f{field}_{patch}*.fits | wc  >{patlist}
% ls WARP/f{field}_{patch}*.fits | sed 's/_/ /g' >>{patlist}
```

```
! Define a box and perform analysis on an image
box 1 sx=0 sy=0 nx=512 ny=512
box 1 sx={3:CNPIX1} ex=520 sy=0 ny=1
abx  1 1 silent mean=a1 rms=rms median=med medrms=medrms sx=sx
! Interactively do image stats analysis
itv

! Image arithmetic with constants
ac  i=scratch bias1
sc  i=scratch bias1
mc  i=scratch bias1
dc  i=scratch bias1

! Image arithmetic with two images
ai i=fixflat i=scratch
si i=fixflat i=scratch
mi i=fixflat i=scratch
di i=fixflat i=scratch

! Mathematical functions
ln i=dest inv
log i=dest
min 1 2
sqrt i=dest

! Clip function
clip 1 max=thresh vmax=10000 min=0 vmin=-100
clip  i=dest sn=snbad snlim=snpoor tv

! Fit a surface, smooth and image
surface i=dest params=(x0,y0,0,slope,1,0,0,0)
```

```
smooth i=dest fwc=5
fiterpolate 8 nx=5 ny=4

! Shift an image
shift   1  dx=45 dy=1

! Create a new image
open 1 nx=512 ny=512
open 1 nx=512 ny=512 header=6
! Window down an old image
wind i=image box=1
! Copy an image
cop   20 1
cop   i=dest i=image box=1
cop 10 i=dest xbin=-2 ybin=2

! Interactively clean an image or do it automatically
tidy
clean i=dest

! Write some text into an image
wtext 1 just=8 value=2000 loc=(i*mx+m,j*my) label="{dith}.{chip}"

! Interactively examine/photometer stars and objects
psf scale=0.1375 rad=10
print phot
print phot file=save.dat
print phot diff

! Display an image
tv i=dest thresh=-200 sat=200 cf=jt
tv 1 screen=1 old sqrt
```

```
tv 1 zoom=-4
color bw inv
! Line graphics
plot 10 y=200
plot i=mybuf r=(6,5,0,1,2,3,4) save=dudley_nt1.spec

! Read an image
rd  1 sky.179
rd 10 ./nt1/lris0037 short
! Read the 7th image in a multi-extension fits file
rd 9 ./nt1/lris0037 mef=7
! Read the 2nd image in a 3-d fits file
rd 9 ./nt1/lris0037 3d=1

! Write an image
wd 1 afsky.cln unsigned
wd 1 zfringe.fits short
wd 5 iflat.016 unsigned scale=3.0
wd i=image fringe.{fnum}

! Call a procedure
call ./abflatten

! Branching constructs (only usable in non-command line procedures)
do k=2,nimage+1
  string FIELD  '@myfile.dat#k#2'
end_do

if xmin<ex&xmax>sx&ymin<ey&ymax>sy
   mc i=tmp 0
else_if rot=1&flip=0
   cop i=image i=image flip
```

```
end_if
```

8.4 UH 2.2-m

Some description of the telescope commands, the environmental stuff, telescope guiding, etc.

At cass and normal mounting of OPTIC (connectors and electronics to the north)

```
Rotator = 360 - PA
ECCWN = 1
```

8.5 WIYN

At WIYN otcom does not directly control the filter wheel, and the communications with the TCS depends on having a server running which sometimes can be unresponsive. Since otcom is not completely integrated into the WIYN environment it is possible for it to hang up instead of ignoring an unresponsive system.

The communications with the WIYN system is via a server called `simplesock.tcl` on ivory. This is found in `wiyn/tonry` and is started using the script `startserver`. Otcom connects to this server and extracts information or makes requests for offsets, filter choices, or exposure time. Typing `tel info` at otcom is a good way to see whether this server is running and communicating.

The WIYN filter wheel has eight positions and can be set from otcom using the `tel filter N` command. Note, however, that out of perversity otcom uses `N=0:7`, whereas WIYN uses `M=1:8`. Don't get confused! The filter information displayed in the GUI is incorrect at WIYN, and the `FILTER` entry in the FITS header is wrong, but the `TELFILT` entry in the FITS header

should be correct.

If the `tel filter` command should bomb, you may need to reinitialize the filter wheel. Log into pearl as `wiyn_ccd`, execute a `start-fsa`, and then do a `filter init`. You can also do a `filter set M` command from this process.

The WIYN shutter offers some challenges. It is a rotating disk which is reputed to provide extremely accurate exposure times, although the disk sweeps across the focal plane at a modest rate. The basic function is to accelerate for 60 degrees, sweep across the focal plane at a constant speed for 60 degrees, and then decelerate for 60 degrees, either leaving the focal plane illuminated or dark depending on the phase of the shutter. The difficulty for OPTIC is that the latency between a `shutter` command and the actual exposure to light can be quite long, certainly at least 1 second and probably more like 1.5–2 seconds until the entire array is illuminated. Upon closing the focal plane is not dark for at least 1 second after the `shutter close`.

I have therefore implemented a "shutter delay" function for WIYN, which causes the `shutter close` routine to pause for a while before returning so that the array really is dark before the next step happens, such as CCD readout. This delay is determined by the `Tmin` parameter used by telescope guiding (`gparams`), since OPTIC does not do telescope guiding at WIYN. A setting of 2.0 sec appears to be satisfactory. If it is too short you will see what looks like vignetting at the very top and bottom of the array when you read out a short exposure – it's the pixels read out before the array is illuminated.

A similar problem is that the start of the OT tracking loop depends on a quick look at the guide stars before erasing the CCDs and starting the observation. I have another delay of length `Tmin` inserted there so that the array is really illuminated before the guide star quick look commences.

Because of its slowness the WIYN shutter also needs a special mode for exposures of less than 4 seconds. For exposure of 1 to 4 seconds the shutter

ignores any `close` command it might receive and simply goes through its own timing for an exposure which commences when it receives a `shutter open`. In order to get short exposure times you must issue a `tel exptime N` to the shutter before starting an exposure, where 1¡N¡4, (and be sure that otcom's exposure time is at least N!). Beware that you have to issue a `tel exptime 10` in order to get back to normal shutter mode. The exposure time `10` can be anything greater than 4 – in normal mode the shutter obeys both the open and shut commands from OPTIC.

If the shutter should hang up, you can also control it from the `start-fsa` on pearl. `shutter init`, `shutter mode normal`, and `shutter exptime N` are the usual commands.

Here is a list of the commands which the WIYN TCS will accept from otcom (remember that short, unique abbreviations are acceptable):

```
 * User Commands
 * -------------
 * telescope info         - returns TCS's idea of where we are
 * telescope afocus A     - sets focus to A
 * telescope rfocus D     - changes the focus by D
 * telescope filter N     - sets filter N (0-7) (NB: OPTIC F# = WIYN F# - 1)
 * telescope exposure E   - sets WIYN shutter exposure time (nec for E<4.0)
 * telescope roffset E N  - offsets the telescope by E and N arcsec
 * telescope xyoffset x y - offsets the telescope by x and y pixels
 * telescope xyoffset x0 y0 x1 y1
 *                        - moves star from x0,y0 to x1,y1 (pix)
 * telescope xyoffset x0 y0 x1 y1 b
 *                        - moves star from x0,y0 to x1,y1 (binned pix)


031012: startup issues
----------------------
* light leak through arcon bolt hole
```

```
* filter/shutter TCS client has to be started
* need tmin = 2.0 to avoid readout before shutter has closed
* box overheats badly with panel on: 41C and rising, 30C with it off.
? horizontal stripes in images???
! Be sure to set PA and parity
```

8.6 Bugs and problems

This section is a hodge-podge of bugs, notes, and specifics.

8.6.1 GUI overhead

You may find after a while of operation that video images periodically flare up in brightness, and that the Unix overhead time is periodically popping up by 20-30 msec. The text status line put out in the otcom window looks like

```
783: t=149.8 dt=191.1 E102 O36/ 2   0.0, 0.0  0.0, 0.0  0.0, 0.0  0.0, 0.01
```

and the `dt` entry gives the average iteration time, the net exposure that the guide star got (`E`), and readout overheads (`O readout_time / unix_time`). This behavior may correlate with the GUI heartbeat, which I think is caused by Tk/TCL becoming extremely inefficient due to some runaway garbage collection. The solution is simply to kill and restart the GUI.

8.6.2 Disk DMA

If you find the system really grinds to a halt just after a readout and write to disk, so much so that the mouse is only barely responsive, you probably do not have disk DMA turned on. As root execute `hdparm -d1 /dev/hda`. This is currently set by rc.local.

8.6.3 Petty annoyances

If otcom won't recognize "backspace" as backspace but reports ^H, you need a "stty erase ^H". Otherwise use a literal ^H for backspace.

8.6.4 Resetting

If otcom is completely unresponsive, or is acting weird, this is the sequence of things to check. Generally speaking you should not have to go beyond (4), and even that is very rare. If you do have to go beyond (4) be sure to let JT know along with as much diagnostic information as possible.

0. First of all look hard at the information in front of you. Often there is a perfectly accurate diagnostic warning or error message if you can perceive it. Are the engineering diagnostics looking reasonable? Are the voltages and temperatures in range? For example, if the electronics temperature gets above 40C you can expect flaky behavior.

1. Is otcom hanging because it's trying to communicate with the TCS? These communications can block for a long time if the TCS is not talking. Try to get the TCS going again.

2. Try a `tdl` to see whether you can talk through the interface to the controller. It may clear things.

3. Try a `pof` / `pon` cycle. Try a `pof` / `df` / `pon` cycle.

4. Try killing otcom and restarting it.

5. If you see a message like

   ```
   "read failed: Resource temporarily unavailable"
   ```

   it means that the driver is hung up. You may be able to diagnose things with a unix `dmesg`. You can remove and reload the driver (as root) with

```
rmmod lotuspci
cd /usr/local/inst/bin
insmod lotuspci.o
```

6. If necessary you may have to cycle the power on the controller at the telescope. This is the power switch on the power supply box. You then need to restart otcom, re-download, etc.

8.7 Camera installation

8.7.1 Booting up the computer

1. Boot up the default kernel (2.4.18-3new)
2. log in, save old XF86Config files, run Xconfigurator (as root) for the monitor you are connected to. The video board will not probe properly so you need to tell it "32 Mb".
3. startx to begin X.
4. run neat (as root) to set up network (DHCP we hope)
5. insert the SDSU PCI driver module (as root)

```
cd /usr/local/inst/bin
insmod lotuspci.o
lsmod
```

6. Make sure that disk DMA is on (as root): `hdparm -d1 /dev/hda`

   NOTE: these two are now implemented in `/etc/rc.d/rc.local` so that they should not need to be done by hand; found in `rc.local` is:

```
insmod /usr/local/inst/src/otcom/Driver/lotuspci-2.11/lotuspci.o
hdparm -d1 /dev/hda
```

7. Edit the files in `/usr/local/inst/config` (create your own and change the two soft links `tel_config` and `filters.def`). If you don't want to communicate with a telescope TCS, comment out the `TELESCOPE` line in `tel_config`.

8.7.2 Hardware setup

1. If the instrument has been recently shipped, open the end of the SDSU electronics box, inspect for damage, and gently reseat all seven boards and connectors. Be careful of the fiber optic connectors at the other end of the box – they are exposed and fragile.

2. Install the dewar, controller, and power supply on the telescope. At the UH2.2-m approximately 30-50VAC exists between the metal of the back of the telescope and the AC ground found at the outlets on the dome floor, so it is essential that the dewar and SDSU boxes be insulated from the telescope and power brought up from the dome floor.

3. Connect up the dewar, controller, fiber between the controller and host. The connection between the dewar is by a pair of 18" 128-pin cables, and the connectors on the dewar go directly to the CCDs so **PLEASE NOTE:**

   These pins are fairly delicate and it takes a lot of force to put on these connectors, so be VERY certain that

   - no male pins are even slightly bent out of line
   - ease the connectors together before tightening
   - you, the dewar, and the controller box are grounded together

4. Power on the controller and three LEDs should go on in the controller box dark window (more lights will come on when the high volts are powered on via software). Look for the green LED on the host interface board which indicates fiber communications. If not on, try swapping the fibers.

8.7.3 Communications and Checkout

1. Start otcom and check to see that it reports communications with the driver, interface, and controller.

2. Download (`df`) and see if it executes without reporting an error.

3. Power on (`pon`) and see if it executes without reporting an error. Use `volt` to examine the voltages. Nominal is 5V, ±16.5V, and +36V. Verify that all the LEDs are on in the dark window on the controller box.

4. Check the CCD temperature (`temp`) and electronics temperature. OPTIC should bottom out around −110C below ambient, and the electronics cannot go above +38C or so before getting flaky (the A/D's get locally very hot and when they thermally shut down they draw a lot of current and generally trigger the power off). If the electronics get too hot, remove the perforated plate on the end of the electronics box. The CCDs normally want to run somewhere between −100 and −110C, so set the temperature regulation (`temp -104`) to somewhere above bottom.

5. Read out (`go`) and see if there appears to be sensible data in all four quadrants (bias should be around 1000 ADU, noise around 2.8 ADU if the CCDs are cold).

8.8 Command list

This is just a brief compendium of the commands, available by typing the `help` command:

```
/* Generally helpful */
??          <> Help
help        <> Help
```

```
quit        <[savefile]> quit; 'Q' for no save
ls          list files
cd          Change directory
set         <var [val]> Set variable
call        macro_file <args> execute a macro
interrupt   interrupt
!           <unix_command> execute unix command (output to otcom window)

/* What's doing? */
status      Show status
temperature <[tset]> request/set CCD temperature
volts       Request voltage status
heartbeat   check on status; update clients

/* Set parameters, dialog to otcom window */
auto        <> Set automatic action
save        <[filename]> Save control params
restore     <[filename]> Restore control params
sformat     <> Set format of CCD readout
sgformat    <> Set format of guide star readout
tparams     <> Set OT tracking parameters
gparams     <> Set telescope guide parameters
gdwell      <d0 d1 d2 d3> Set guide dwell factor

/* Data file specs */
fnumber     <fileno> Set running file number
fprefix     <prefix> Set file prefix

/* Exposure times */
etime       <seconds> Set exposure time
gtime       <msec> Set guide exposure time
```

```
/* Actually do things */
fgs          <[time [dt coords]] > quick look and find guide stars
go           <[#rep]> Start exposure
resume       Resume interrupted exposure
telescope    <[command]> Send telescope command
filter       <[filt]> Set filter (0-3)
move         <pattern params> offset charge from tracking

/* Set some header parameters */
object       <name> Set object name
imtype       <> Set image type: 1-7 = Obj Bias Dark Dome Sky Foc Std
comment      <comment> Set comment in FITS header

/* Do things piecemeal */
clear        <[#rep]> Clear (wipe) CCD
expose       <seconds> Do exposure; set exposure time
shutter      <(open|close)> Command shutter
rc           Read CCD
wfile        Write FITS file
video        Run video

/* Probably not useful to general observer */
tdl          <[pattern]> Test data link
guide_stars <x,y1a x,y1b x,y2a x,y2b | prev> [phys] guide star locs
led          <dt { |b|r}> Light LED for dt msec

/* Turn on and manage DSP, potentially very dangerous for general user */
df           <[file]> Read DSP file and download
pof          Turn off high voltage
pon          Turn on high voltage

/* Potentially extremely dangerous for general user */
```

```
rdm               { |t|u} B:hex[-hex]   read from SDSU memory
wrm               { |t|u} B:hex value   write to SDSU memory
cmd               cmd { |t|u} XXX args   send SDSU command
rewrite_clocks    rewrite horizontal clock tables in DSP memory
```

8.9 Variable list

This is just a brief compendium of the variables, available by typing the `set help` command:

```
>> set help
Set can be used to examine or set otcom variables but does *not* check for
reasonable values nor does it perform actions such as 'shutter'.  You must
use the appropriate otcom verb to cause actions to take place.
variable   / abbrev   value       description
clobber   / clb    = 0           Silently overwrite files?
shortfits / fits   = 1           Write signed FITS files?
sound     / snd    = 1           Make sounds?
verbose   / verb   = 2           Level of verbosity
fgs       / fgs    = 0           Options for fgs
imtype    / imtyp  = 0           Image type
object    / obj    = {}          Object name
fprefix   / fp     = {/tmp/ccd}  File prefix
frameno   / fn     = 1           Frame number
etime     / et     = 1           Requested exposure time (s)
gtime     / gt     = 0.1         Requested guide time (s)
acctime   / acct   = 0           Accumulated time (up to last interrupt)
inttime   / intt   = 0           Accumulated time (since last interrupt)
usedsptime / dspt   = 0           Use DSP 65002 clock?
nwipe     / nwipe  = 1           Auto param: number of times to clear
doshut    / doshut = 1           Auto param: open shutter?
doread    / doread = 1           Auto param: read CCD?
```

```
wrdat     / wrdat  = 1          Auto param: write frame?
doot      / doot   = 0          Auto param: write GS info?
wrot      / wrot   = 0          Auto param: write OT info?
wrvid     / wrvid  = 0          Auto param: write video?
gostat    / gostat = 0          Go status
sbx       / sbx    = 2          Science bin factor in x
sby       / sby    = 2          Science bin factor in y
ssx       / ssx    = 0          Science skip offset in x
ssy       / ssy    = 0          Science skip offset in y
snx       / snx    = 1024       Science number of pixels in x
sny       / sny    = 1026       Science number of pixels in y
snb       / snb    = 32         Science number of bias pixels
gbx       / gbx    = 2          Guide bin factor in x
gby       / gby    = 2          Guide bin factor in y
gsx       / gsx    = 1          Guide skip offset in x
gsy       / gsy    = 1          Guide skip offset in y
gnx       / gnx    = 24         Guide number of pixels in x
gny       / gny    = 24         Guide number of pixels in y
algorithm / alg    = 10         GS prediction algorithm
gswait    / gswait = 5          How many GS iterations before shifting?
gsgain    / gsgain = 1.00       Fraction of indicated shift to make
gsmax     / gsmax  = 8          Maximum distance allowed for shift (p)
gsexpo    / gsexpo = -1.0       What power to use for (1/d) GS weighting?
gsbias    / gsbias = 1          Subtract bias from video frames?
otmax     / otmax  = 1000       Maximum distance allowed for OT shift (p)
psfalgo   / psfalg = 1          Psf algorithm (u=0/1:p/Gauss, h=1: ignore R)
use0l     / use0l  = 1          Use for CCD region 0 lower (guide)
use0u     / use0u  = 1          Use for CCD region 0 upper (science)
use1l     / use1l  = 1          Use for CCD region 1 lower (guide)
use1u     / use1u  = 1          Use for CCD region 1 upper (science)
use2l     / use2l  = 1          Use for CCD region 2 lower (guide)
use2u     / use2u  = 1          Use for CCD region 2 upper (science)
```

```
use3l     / use3l  = 1          Use for CCD region 3 lower (guide)
use3u     / use3u  = 1          Use for CCD region 3 upper (science)
gx0       / gx0    = 0.0        Location in x of guide star in region 0
gy0       / gy0    = 0.0        Location in y of guide star in region 0
gx1       / gx1    = 0.0        Location in x of guide star in region 1
gy1       / gy1    = 0.0        Location in y of guide star in region 1
gx2       / gx2    = 0.0        Location in x of guide star in region 2
gy2       / gy2    = 0.0        Location in y of guide star in region 2
gx3       / gx3    = 0.0        Location in x of guide star in region 3
gy3       / gy3    = 0.0        Location in y of guide star in region 3
telguide  / telg   = 0          Use telescope for guiding?
telgtol   / telgtol= 2.00       Error before telescope correction (p)
telgain   / telgain= 0.50       Fraction of indicated shift to request
teltmin   / teltmin= 1.00       Minimum elapsed time between requests (s)
patop     / patop  = 0.0        PA of top of image
eccwn     / eccwn  = 1          Image parity: E CCW from N?
telwait   / telwait= 5          Iterations before telescope guide starts
telfix    / telfix = 0          Fix telescope origin at gsx,y (0/1)?
movepat   / mpat   = 0          Move pattern
moveeast  / mde    = 0.0000     Move amplitude east (arcsec)
movenorth / mdn    = 0.0000     Move amplitude north (arcsec)
movedx    / mdx    = 0.000      Move amplitude in x
movedy    / mdy    = 0.000      Move amplitude in y
movenrep  / mnrep  = 0          Move repetition
gnap0     / gnap0  = 1          Dwell factor for guide region 0
gnap1     / gnap1  = 1          Dwell factor for guide region 1
gnap2     / gnap2  = 1          Dwell factor for guide region 2
gnap3     / gnap3  = 1          Dwell factor for guide region 3
namp      / namp   = 0          Number of amplifiers
geom0     / geom0  = 0          Geometry of amp 0
geom1     / geom1  = 0          Geometry of amp 1
geom2     / geom2  = 0          Geometry of amp 2
```

```
geom3     / geom3  = 0          Geometry of amp 3
p5v       / p5v    = 0.00       Current +5 volts
p5target  / p5t    = 0.00       Target +5 volts
p5delta   / p5d    = 0.00       Tolerance for +5 volts
p15v      / p15v   = 0.0        Current +15 volts
p15target / p15t   = 0.0        Target +15 volts
p15delta  / p15d   = 0.0        Tolerance for +15 volts
m15v      / m15v   = 0.0        Current -15 volts
m15target / m15t   = 0.0        Target -15 volts
m15delta  / m15d   = 0.0        Tolerance for -15 volts
p35v      / p35v   = 0.0        Current +35 volts
p35target / p35t   = 0.0        Target +35 volts
p35delta  / p35d   = 0.0        Tolerance for +35 volts
pon       / pon    = 0          Power on?
shutstat  / shut   = 0          Shutter status
tempccd   / tccd   = 0.0        CCD temperature
tempset   / tset   = 0.0        CCD set temperature
tempelec  / telec  = 0.0        Electronics temperature
heatperc  / hperc  = 0          Heater percentage
filtno    / filt   = 0          Filter number
nfilt     / nfilt  = 8          Number of filters
filtname0 / filnam0= {V}        Filter 0 name
filtname1 / filnam1= {R}        Filter 1 name
filtname2 / filnam2= {U}        Filter 2 name
filtname3 / filnam3= {z}        Filter 3 name
filtname4 / filnam4= {W022}     Filter 4 name
filtname5 / filnam5= {B}        Filter 5 name
filtname6 / filnam6= {I}        Filter 6 name
filtname7 / filnam7= {W021}     Filter 7 name
```

8.10 Pinouts

This is how the wires are connected from CCD through the Nanonics connector on the package, to the micro-D on the flexprint board in the dewar, through the hermetic connector, to the 37-D connectors which attach to the SDSU boards, and finally to the SDSU defined signals.

```
 011220: fixed error in shutter wiring to 15-pin
 011218: ground FBC
 011201: swap B-side P1 and P2
 011128: wire utility connections
 011111: define SDSU connections (and wire SDSU-128 cables)
```

Vid/Clk	37D	Herm	u-D	Nano	CCD	Signal	NB: A=short, B=long cable
P06	7	77	21	1	CCD-A:	IA3=P3U	
P07	8	74	3	2	CCD-A:	IA4=P4U (GL:T2)	
P04	5	81	20	3	CCD-A:	IA1=P1U	
P05	6	79	1	4	CCD-A:	IA2=P2U	
V05		87	34	5	CCD-A:	SCPb	
V09		86	15	6	CCD-A:	OGb	
V03		85	33	7	CCD-A:	RDb	
S00	1/13	126	14	8	CCD-A:	RGb	
V01		128	13	10	CCD-A:	VDDb (Std)	
outA		118	32	11	CCD-A:	OSb (Std)	
retA		119	31	12	CCD-A:	Retb	
S04	5/17	121	12	13	CCD-A:	SWb	
PGnd	22	127	30	15	CCD-A:	SUB	
S02	3/15	110	9	16	CCD-A:	S2b	
S01	2/14	108	28	17	CCD-A:	S1b	
P08	9	117	36	18	CCD-A:	SDG (EC1)	
V07		125	18	19	CCD-A:	SDD (EC2)	

		110	29	20	CCD-A: S1a (=S2b)
		108	11	21	CCD-A: S2a (=S1b)
S03	4/16	112	10	22	CCD-A: S3
		127	27	23	CCD-A: SUB
		123	19	24	CCD-A: FBC (EC3) GROUNDED at flex
S05	6/18	114	8	25	CCD-A: SWa
NC		101	26	26	CCD-A: Reta
NC		100	25	27	CCD-A: VDDa (FB)
NC		102	7	28	CCD-A: OSa (FB)
		104	6	30	CCD-A: RGa (tied to RGb at flex)
		85	24	31	CCD-A: RDa (tied to RDb at flex)
		86	5	32	CCD-A: OGa (tied to OGb at flex)
		87	23	33	CCD-A: SCPa (shorted on chip)
P01	2	91	22	34	CCD-A: FS2=P2L
P00	1	93	4	35	CCD-A: FS1=P1L
P03	4	75	17	36	CCD-A: FS4=P4L (GL:T1)
P02	3	89	2	37	CCD-A: FS3=P3L
P18	19	54	36	1	CCD-B: IA3=P3U
P19	33	52	17	2	CCD-B: IA4=P4U (GL:T1)
P17	18	58	37	3	CCD-B: IA1=P1U (note P1-P2
P16	17	56	19	4	CCD-B: IA2=P2U swap here)
		41	34	5	CCD-B: SCPb (shorted on chip)
V10		40	15	6	CCD-B: OGb
V04		39	33	7	CCD-B: RDb
S06	7/19	35	14	8	CCD-B: RGb
V02		31	13	10	CCD-B: VDDb (Std)
outB		33	32	11	CCD-B: OSb (Std)
retB		32	31	12	CCD-B: Retb
S10	11/36	24	12	13	CCD-B: SWb
SGnd	23/32	1	30	15	CCD-B: SUB
S08	9/34	20	9	16	CCD-B: S2b
S07	8/33	18	28	17	CCD-B: S1b

```
   P20      34     10     21     18     CCD-B: SDG (EC1)
   V08             4      2     19     CCD-B: SDD (EC2)
                  20     29     20     CCD-B: S1a (=S2b)
                  18     11     21     CCD-B: S2a (=S1b)
   S09   10/35    22     10     22     CCD-B: S3
                   1     27     23     CCD-B: SUB
                   2      1     24     CCD-B: FBC (EC3) GROUNDED at flex
   S11   12/37    14      8     25     CCD-B: SWa
   NC             12     26     26     CCD-B: Reta
   NC              7     25     27     CCD-B: VDDa (FB)
   NC             11      7     28     CCD-B: OSa (FB)
                   5      6     30     CCD-B: RGa (tied to RGb at flex)
                  39     24     31     CCD-B: RDa (tied to RDb at flex)
                  40      5     32     CCD-B: OGa (tied to OGb at flex)
                  41     23     33     CCD-B: SCPa (shorted on chip)
   P12      13     45     35     34     CCD-B: FS2=P2L (note P1-P2
   P13      14     47     16     35     CCD-B: FS1=P1L  swap here)
   P15      16     51      3     36     CCD-B: FS4=P4L (GL:T2)
   P14      15     43     18     37     CCD-B: FS3=P3L

-----------------------------------------------------
Utility board connections to 128-pin and 15-pin Dsub
-----------------------------------------------------
DOUT11/12  A14/A13  72   violet       LED+ y:0x1000 = red y:0x0800 = blue
Gnd        B14/B13  73   blue         LED-
AIN5       A27+C32  71   red     A    Diode(+)
AOUT0      C25      70   yellow  B    H1
Gnd        B32      67   black   C    TFD2 & Diode(-) return
Gnd        B25      69   yellow  D    H2
      BNC           68   white   E    TFD1

DIN12      C13    D-03   blue    Filter posn 0   Y:0 0x0100
```

DIN13	C12	D-11	violet	Filter posn 1	Y:0 0x0200
DIN14	C11	D-02	brown	Filter moving	Y:0 0x0400
DOUT7	A02	D-04	yellow	Filter req 0	Y:1 0x0080
DOUT8	A17	D-12	green	Filter req 1	Y:1 0x0100
DOUT10	A15	D-10	gray	Shutter pulse	Y:1 0x0400 (For LFW only)
SHUT	C22	D-05	orange	Shutter	NOTE: 200 ohm pullup to +5V
(NC)	C21	D-01	red	+5V	NOTE: Jumpered on utility board
Gnd	B21	D-09	black	Gnd	