

Advanced IRAF Commands

Now that you've gotten the hang of the basics like *imexamine* and using *ds9*, it's time to move on to some new commands.

Phelp

The task, *phelp*, is one of the most useful you'll find in IRAF. It will bring up a help page on any IRAF task. The syntax for using *phelp* is as follows:

```
cl>phelp imexamine
```

This will bring up the help page for *imexamine*. At the top of the page, you can find where within the IRAF packages *imexamine* exists. In this case, it is *tv.imexamine*. First it lists the possible input parameters. Required parameters are always listed first, and then other possibilities come next. After that you get a brief description, details on the interactive options, and a couple of examples on usage. To move around in the help page, you can press the *spacebar*, *return*, *d* or *N* key to move down different amounts and the *u* or *P* key to move up.

Looking Through the IRAF Packages

IRAF has a set of premade routines that are set up in a kind of directory system (just like the directories and files in your cosmos account). In order to open some packages, you have to be in the correct directory. You can find the directory path you want using *phelp* described above. From now on, I'll call directories within IRAF *packages*, and the commands *tools*. In order to list the packages & tools available to you where you are, use the *?* key. To list all packages and tools, use *?*.

Now if I want to move into a specific package from where IRAF starts, for example *images*, I type:

```
cl>images
```

and get the response:

```
imcoords.  imfit.  immatch.  tv.  
imfilter. imgeom. imutil.
```

This is now a list of packages available to move into. If I want to leave the *images* package, I type *bye*.

Command History

So by now you've probably realized that IRAF won't let you use the up arrow to retrieve commands you've previously used. Instead, to access this command history, first type *e* and return. You should see something like this:

```
cl>e
images
```

if going into the images package is the last thing you've done. You can now access the command history by using the up and down arrow keys.

Epar

The command, *epar*, stands for edit parameters, i.e. it opens a screen and displays all of the possible parameters that you can supply. For example, if I type:

```
cl>epar implot
```

I see the following screen:

```
PACKAGE = plot
      TASK = implot
```

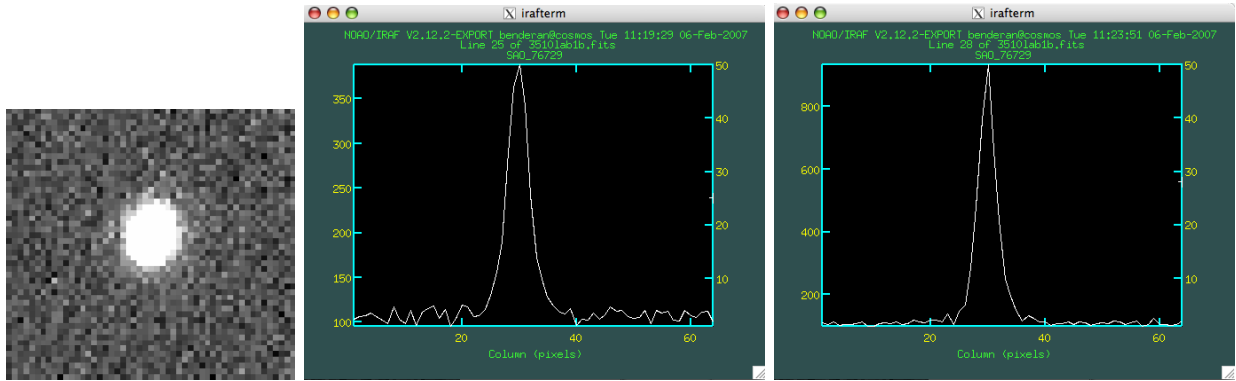
```
image  =  image to be plotted
line   =  line to be plotted
(wcs   =  logical) world coordinate system
(step  =  0) Step in pixels for j/k key
(coords =  ) graphics cursor input
(device =  stdgraph) graphics device for plots
(mode  =  ql)
```

I can now move through these options using the arrow keys, and enter the appropriate values. For example, if I wanted to implot the image *m56.fits* the line would look like:

```
image  = m56.fits  image to be plotted
```

Note that parameters that require input are the first listed, with optional parameters listed afterwards with parenthesis around them. Once you have filled in the values to your parameters, you can quit and save by typing *:q*, or you can run the task with *:go*. If you ever want to look at the parameters already filled in for a task (IRAF will remember what you last did), you can use the list parameters command *lpar*, e.g.

```
cl>lpar implot.
```



Implot

After *imexamine*, the command *implot* (image plot) is one of the most useful. *Implot* takes a 3-dimensional image, chooses a single line or column from that image and plots it 2-dimensionally. This is very useful for looking at the intensities in each pixel along a single line or column. For example, imagine you have an image of a star called *star.fits*. To implot I type:

```
cl>implot star.fits
```

A window like the two displayed above will pop up (with red crosshairs that you can move around). There are many different options that you can utilize in this interactive mode. To list this, make sure the pop-up window is active, and type *?*. After you're done in this screen, type *q* twice to quit. A quick command to find the value at any given pixel position is to move the vertical crosshair to the desired location and type *spacebar*. Two of the most useful are the commands that tell which line (y-position from ds9) or column (x-position from ds9) you would like to plot. To plot column *n*, I type *:c n* within the interactive window. A yellow line showing your command will popup at the bottom of the window when you begin typing. The same thing works for choosing a line, *:l n*.

Now that you can specify which line to plot, you may want to blow up a portion of the plot for closer examination. This is done through the *e* (expand) command. Use the crosshairs to mark out the lower left and upper right corners of the box that will become your new plot range. First put the crosshairs where you would like the lower left corner to be, type *e*, move the crosshairs to the upper right and type *e* again (Note that it prompts you at the bottom of the screen). If you don't like the result, type *r* for redraw the original line and try again.

After zooming in on the profile of a star, you might want to measure it (just like in *imexam*). The way to do is using the *p* command. Place the horizontal cross-hair in the middle of the background noise and the vertical cross-hair on the left side of the stellar profile and type *p*. You want this be significantly far from the wings of the star so as to get a good measure of the background. Now move the cross-hairs to the righthand side of the star, keeping the

horizontal one in the background noise and type *p* again. At the bottom of the screen the measurements describing the profile will pop-up. There are four bars of information that you can move through using the / key. The first bar, gives the line or pixel describing the 2-D centroid, the full width at half max (denoted by a dashed line overplotted on the screen), the peak value, and the background level. The other bars give moments of the background subtracted data.

A last interactive command is the *s* command. It works essentially the same way as the profile command, by bounding a region with the crosshairs and typing *s* twice. The output is then basic statistics of the region, the median, mean, standard deviation (rms), sum, and number of pixels in the designated region.

To leave the implot interactive environment, simply type *q* and you're IRAF command line will become active again.

Imcopy

At some point you may want to copy an image within IRAF to a new file, for example as a backup. While the UNIX command *cp* will accomplish transferring the entire image to a new file, you may want to only transfer a portion of the new image. This is done through the *imcopy* command. The basic idea is that you can specify the region in pixels to transfer. So if I want to copy from pixel 400 to 500 in the x-direction, and pixel 600 to 700 in the y-direction I would type:

```
cl>imcopy f56.fits[400:500,600:700] f56new.fits
```

Now I have a new .fits file that is only 100 X 100 pixels as opposed to 1024 X 1024 pixels.

Imarith

Once we begin using calibration frames in data analysis, you'll need to perform basic arithmetic functions on images. The command, *imarith* is the quickest way to accomplish this within IRAF. In order to use imarith I suggest:

```
cl>epar imarith
```

You need to fill in the name of image or a number, the operand (+, -, *, /, min, max), a second image or number, and what the result will go into. If I wanted to subtract the image *dark.fits* from the image *science.fits* and put the result in *final.fits* my screen would look like:

```
PACKAGE = imutil TASK = imarith
operand1= science.fits Operand image or numerical constant
op = - Operator
operand2= dark.fits Operand image or numerical constant
result = final.fits Resultant image
```

After filling in these values I would type `:go` to activate the command sequence. I can also do the same thing from the IRAF command line:

```
cl>imarith science.fits - dark.fits final.fits
```

Now there should be another image called *final.fits* in the same directory as the other images. You can access this new image through `ds9`, `imexam`, `implot`, and any other IRAF command. When you start doing more advanced image combinations, there is another task *imcombine* that you'll need to use, but more info on that will come later.

Imstatistics

Sometimes you'll want to compute statistics over a region of several columns or rows of an image. The correct task for this is *imstat*. To calculate statistics on the region [400:500,600:700] of image *final.fits* using *imstat*:

```
cl>imstat final.fits[400:500,600:700]
```

The output looks something like this

```
#          IMAGE  NPIX  MEAN  STDDEV  MIN  MAX
test.fits[400:500,600:700]  10201  135.4  15.59  48.  794.
```

You'll find this task extremely useful later on in the course when you need to do noise calculations.

Imheader

Sometimes you might find yourself needing some piece of information about your image that you forgot to record (or something else). While you should always record your observations carefully, there is sometimes a way to recover this information. Every fits file has two parts, the actual image data, and a header. In this header is a list of information and keywords describing your observations. To access the short version of this header for the image *f56.fits*:

```
cl>imhead f56.fits
```

A line stating, `f56.fits[1024,1024][ushort]:`, pops up. You get the image name and size, but nothing else. There is also a long version of this header, accessed by:

```
cl>imhead f56.fits l+
```

The resulting output looks like:

```
f56.fits[1024,1024][ushort]:  
No bad pixels, min=0., max=0. (old)  
Line storage mode, physdim [1024,1024], length of user area 1053 s.u.  
Created Wed 21:07:45 13-Oct-2004, Last modified Wed 21:07:45 13-Oct-2004  
Pixel file "f56.fits" [ok]  
BSCALE = +1.000000000000e+000  
BZERO = +3.276800000000e+004  
BIAS = 100  
ORIGIN = 'SBO '  
TELESCOP= '192 24 '  
FOCALLEN= +1.920000000000e+002  
APERTURE= +2.400000000000e+001  
OBSERVER= 'Observer '  
DATE-OBS= '2004-09-18 '  
TIME-OBS= '03:40:53 '  
CCDSFPT = 1  
CCDSUBFL= 0  
CCDSUBFT= 0  
CCDXBIN = 1  
CCDYBIN = 1  
EXPSTATE= 293  
TEMPERAT= -4.817803680962e+000  
INSTRUME= 'SBIG ST-1001E/1001XE'  
E-GAIN = +2.720000000000e+000  
XPIXSZ = +2.400000000000e-002  
YPIXSZ = +2.400000000000e-002  
SBIGIMG = 9  
FILTER = '3 '  
EXPOSURE= +2.000000000000e+001  
CBLACK = 105  
CWHITE = 175
```

Most of this information will become invaluable as you do more advanced observations. For now, the keywords of importance to you are *Filter* (the filter number), *Exposure* (exposure time in seconds), *Date*, and *Time*.

This is obviously a lot to absorb in one sitting. Instead I suggest you use this as a reference document, and try out these commands as you go. Have fun, good luck, and don't forget to ask questions if you get stuck!