2 Interactive Imaging with 'Skyview'

2.1 Introduction

This chapter describes the interactive use of **skyview**. The next chapter will describe the server mode, in which **skyview** is driven by another program. The server mode description assumes that you are familiar with the interactive use of **skyview**, so if you are planning to use the server mode, you should still start with this, the interactive chapter.

The remainder of this chapter will be broken down into the following pieces:

- We will begin at the beginning: getting started.
- We will then get involved with the basics of image display.
- The third section will give directions on image analysis and data inspection tools.
- Next will come a section on displaying catalog or table information in conjunction with images.
- Following that will be some discussion on the **skyview** environment.
- Finally, the complete list of **skyview** functions, their abbreviations, and a brief description of each, is at the end of the chapter.

2.2 Getting Started

To begin working with the program, type the name: `skyview`
To get out of the program, type the command: `unix, quit,end` **or** `exit`

One of the most important things you should be aware of is that you should **never** move the mouse between windows while in the middle of typing in a **skyview** command. The program will not recognize the command, and you will have to re-type it.

2.2.1 Listing files

Listing the files within a directory can be done two different ways.

```
> dir
```

will list the files in the current working directory.

```
> ls
```

will do the same thing. The **ls** command was added to make life easier for those who like more information than just the names of the files in the directory. All the arguments available to **ls** in UNIX work within **skyview**.

2.2.2 Changing Directories

Moving around between directories within **skyview** is done in the same manner as with UNIX.

```
> cd images
```

will move you to a subdirectory (of the current directory) named **images**.

```
> cd ..
```

will move you up one level in the directory tree.

```
> cd
```

will move you back to your home directory.

```
> cd /home/indus/jane
```

will move you to the given directory: in this case, /home/indus/jane.

```
>       cd ~jane
```

will move you to **jane's** home directory.

```
> pwd
```

displays the pathname of the current working directory.

2.2.3 That Darn Rodent

Many functions in **skyview** use the mouse attached to the workstation to interactively change either the images being displayed or the graphics. For each function, the three buttons on the mouse are defined to assist you with the job. In every case, you can end the function simply by clicking either the center button on the mouse or the 'Return' key on the keyboard.The following diagram describes the definition of the buttons in each case.

command:	left button:	center button:	right button:
>	st	pi	zp
st	inverse	end	0-255
pi	pick	end	—
zp	zoom in	end	zoom out
da bo	center	end	size
da ci	center	end	size
da po	corner	end	—
da el	center	end	size/angle
sl	center	end	end points
an	—	end	—
ba	pick	end	—
fl	pick	end	—
gpi	pick	end	—

The **stretch**, **pick**, and **zoom/pan** functions (which are shown on the first line of the diagram[2]) can be invoked either by clicking the appropriate button on the mouse when **skyview** has issued its prompt (and the cursor is in an image window), **or** by typing in the commands at the prompt, as is shown elsewhere in this document. The rest of the functions in this table **must** be entered at the command prompt.

2.2.4 Keeping Track of Your Work

The **session history** function is a good way to keep a record of what you did inside **skyview** without having to write everything down.

```
> sh
```

or

```
> sh on
```

turns the session history on. By default, the **sh** command writes to a file named *session* in your home directory, but you can specify another file name if you'd like.

```
> sh session.1
```

[2]The three main mouse buttons can be changed to invoke any functions you desire, by specifying them as **X-resources**. See **Section 2.6.5** for more information on **X-resources**.

will create a file named **session.1** and begin keeping the history of your work within **skyview**. If you're working in a special directory, and want the session history to be logged to a file in that directory, you must specify the full path name, like this:

```
> sh /project/work/images/tues.session
```

By default, **sh** always *appends* to the file. If the keyword **create** is used:

```
> sh session.1 create
```

sh will erase the old **session.1** file and begin anew. If you want to make notes or put comments into the session history, the hash mark, or pound sign(#) as the first character on the command line defines the entire line as a comment which is then inserted into the history file.

```
> #Image fftm31b4.fits is the output from my FFT on band 4 of m31.
```

The entire line is written to the history file.

```
> ph
```

Prints the **session history** file to default printer. If **session history** is turned on, the file you are currently logging to will be printed. If session history has been turned off, the last file used during the current session will be printed. If you haven't used the **session history** function at all during the session, and you decide to print the history file, if you don't specify the file name, it will automatically print the file named *session* in your home directory.

To send the default session history file to the screen, use the command:

```
> ph terminal
```

2.3 Image Display

There are many aspects to image display that will be addressed in this section. The functions which *assist* you in the display process will also be discussed. None of the commands in **skyview** is more important however, than the **paint** command itself.

2.3.1 Paint

```
> pa image1.fits
```

A new window pops up with **image1.fits** appearing in it.

2.3.2 Range mapping

Images are painted using a default range for transformation: the pixels are histogrammed, and the pixel values in the file corresponding to the first and ninety-ninth percentiles are used for the minimum and maximum levels to transform to screen values 0 and 255.[3]

The range can be explicitly set to any values (or percentiles on the histogram) you choose. For the previous paint, the image looks saturated (i.e., a greater range is needed), so I will try using the full **range** of file pixel values for the transformation by specifying to the range command to use the minimum and maximum values in the image:

```
> ra min max
```

Or, I could have shortened it to: `ra m m`. Once you have changed the **range**, you must repaint the image in order to make the new values take effect.

```
> ra m m
> pa
```

The program remembers the filename of the last image you painted[4] so you don't have to specify the filename if you want to repaint the same image.

2.3.3 Histograms

To show a histogram (of screen values) on the screen of the last image you painted:

```
> hi
```

does the trick. The name of the image is given in the title bar of the window containing the histogram.

To display a histogram on the screen of the original image data from the file (in units specified in the image header), use the **histogram_original** command:

```
> ho
```

[3]See **Appendix A** for more discussion of the range mapping process.

[4]i.e., the **filename** parameter of the **pa** command is 'sticky'. See **Section 2.6.2** for information on 'sticky' features.

is all it takes. In both cases, a small window appears on the screen, showing the bins and the distribution of the pixels among them. Hereafter, every time a new image is painted, it's histogram will be shown in the histogram window, as long as the histogram window has not been erased.

If you want to examine only the distribution of pixel values between a certain range, you can give **ho** lower and upper limits. These limits can be in absolute pixel values or in percentages (by using a '%' sign), or in any other format allowed by the **range** command.

```
> ho 20% 90%
```

Will show the distribution of the pixel values between the twentieth and ninetieth percentiles.

```
> ho 100 1e4
```

Will show the distribution from one hundred (in whatever units the image is in) to ten thousand.

If you change the limits and want to get back to the minimum and maximum image pixel values for the histogram, you can also specify the **min** and **max** keywords.

```
> ho min max
```

2.3.4 The Mode Command

The default mode of transformation from file pixel values to screen values is linear. For some images a logarithmic transformation will show more detail...

```
> mo log
> pa
```

Note that, just as with the **range** function, once you have changed the **mode**, you must repaint the image in order to make the new mode to take effect. All the images that get painted after this will be using a logarithmic transfer function. To change back, use the

```
> mo lin
```

command.

If the image still doesn't look the way you think it should, try some 'picks' to find out what typical high and low values are. The command is:

```
> pi
```

Now, place the cursor on the current image and click the left mouse button to select each point. Here's an example of some typical **pick** output:

```
-8.73691e+06 JY/SR at RA = 21h31m36.0s DEC = 58d36m25s
-1.35743e+07 JY/SR at RA = 21h34m24.3s DEC = 58d42m49s
-1.88855e+08 JY/SR at RA = 21h39m08.1s DEC = 58d05m59s
```

Clicking the middle mouse button or the return key on the keyboard ends the **pick** function. Now, set the range to two of those values and re-paint the image.

```
> ra -1e7 -1e8
> pa
```

With most images, the default range works well, but by now you can tell that iterating with different range values will help bring out features that may be difficult to see.

2.3.5 The Frame Command

So far in this section, there have been 3 examples of the **paint** command. If you have tried them out, you will have noticed that every **paint** placed the image in the same window. So you may ask, *'What if I want each image painted in a separate window?'* Well, that's what the **frame** command is for. It works 2 different ways. The easiest way is to issue the command:

```
> fr adv
```

which tells **skyview** to *advance* the frame number (i.e., open a new window) automatically every time a new **paint** command is given. This feature is 'sticky',[5] so in order to turn it off, use the command:

```
> fr noadv
```

The other way to tell **skyview** not to paint images into the same window is to specify the frame number literally before each paint command. For example,

```
> fr 1
```

[5]see **section 2.6.2** for more information on 'sticky' features

declares frame number **one** to be the next frame number to be used. To advance to the next frame for a subsequent paint, you would have to either specify the new frame number

```
> fr 2
```

or turn on the frame advance.

2.3.6 Stretch

There are two ways to change the color table stretch of an image: Interactive and command line input.

Interactive Color Table Stretching. Put the cursor into an image window, and click the left button on the mouse.

or

Type in the **stretch** command:

```
> st
```

With either way of invoking **stretch**, the buttons on your mouse are now redefined for use with the stretch command:

command:	left button:	center button:	right button:
st	inverse	end	0-255

Moving the mouse from left to right changes the intensity, and up and down changes the contrast. To get the stretch back to the default 0 to 255 data values, click the right button. To invert the stretch, click the left button. To stop the interactive stretch, click the middle button.

Command Line Input of Color Table Stretch Parameters. Put the cursor into any **skyview** window and use the stretch command to change the color table using exact screen values.

```
> st 50 200
```

2.3.7 Changing Color Tables

The black–and–white image is the default, but there are 7 other pre-defined color lookup tables available. The color tables are numbered 0 through 7, with color table '0' as the default. To change to a different color table use the **ct** command.

```
> ct 4
```

This has changed the black–and–white image to pseudocolor. Here is a brief description of the color tables available:

0	grey scale: black to white
1	reverse grey scale: white to black
2	a trip around a color cube
3	eyeballed attempt at matching a spectrum
4	standard for false colors: black, blue, green, red, white
5	reverse color table 4
6	same as 4, but with blue compressed
7	for difference images; white at center (128) with red and blue at the extremities

Users are also encouraged to create their own color tables[6] customized to their own needs.

2.3.8 Zoom and Pan

There are two ways to zoom in and out on an image: Interactively and with command line input.

Interactive Zoom/Pan. Put the cursor into an image window, and click the right button on the mouse.

or

Type the **zoom/pan** command:

```
> zp
```

By using either of these methods, the buttons on the mouse are now redefined for use with the **zoom/pan** command:

[6]See **Appendix B** for information on how to create and use your own color table.

command: zp	left button: zoom in	center button: end	right button: zoom out

A small box now appears in the upper left corner of your image showing the entire image. An outline of a box appears within this image, with a cursor in the middle. Move this cursor around to the area you wish to zoom in or out on. Clicking the left button will cause the white box to get smaller (zoom in). Clicking the right button will cause the white box to get larger (zoom out). The zoom factor will be reported in the text window. Once you have the zoom factor set on the area you want to see, click the middle mouse button, and the image will be redisplayed with the new zoom and pan. **Tip:** Having zoomed the image, you will probably want to re-size the window. Rather than going through the pain of trying to manually re-size the window, use the **border** command to do it. The command

```
> border 0
```

will resize the window to be the size of the image as it is currently zoomed and place a border containing zero blank pixels around it. Of course, you can use a number other than zero if you prefer a border of blank pixels around the image. (If you want to put a coordinate grid or contour plot on top of the image, for example.)

Command Line Input of Zoom/Pan Parameters. Put the cursor into the text window running **skyview** and use the **zoom/pan** command to change the zoom and pan.

```
> zp 4 5h32m56.8s 9d40m
```

This **zooms** the image to a factor of **4**, centered on the given coordinate.

2.3.9 Other Parameters of the Paint Command

If you have been changing the color table stretch interactively in between **paint**s, you will notice the **paint** command automatically snaps the color table stretch back to 0-255 before it paints the image. If you want the image you are about to paint to appear with the same color table stretch, use the `nost` option.

```
> pa image nost
```

or

```
> pa nost
```

if you want the same image painted again. The nost option is 'sticky'. It will stay on until you turn it off with

```
> pa image st
```

Now, suppose I want to paint the image in a window placed in a certain area of the screen. I can use the X-Windows geometry as an argument for placement of the image frame. The easiest way to learn about X-Windows geometry is from examples. Just remember that the upper left corner of the screen is coordinate +0+0. Here are two examples:

```
> pa image =+50+300
```

paints the upper left corner of the image 50 pixels in from the left edge, and 300 pixels down from the top of the screen. The next example is a bit more complex:

```
> pa image =30x40-50-100
```

This time, the image gets painted in a window 30 pixels wide, 40 pixels high, 50 pixels in from the right edge of the screen and 100 pixels up from the bottom. **Note:** If the image in the above example is larger than 30 x 40 pixels, you will only be able to see the upper left corner of it, unless you zoom out or re-size the window.

2.3.10 24-bit Display Devices

If you have full color image display functionality (i.e., 24-bit), **skyview** has the capability of displaying separate images into each color plane (8-bits each: red, green and blue). Your workstation must have 24-bit display capability supported by your X-server (e.g., a Sun GS Graphics Accelerator with OpenWindows version 3.0).

The **map** command allows you to map separate images into one color plane at a time. Here's an example of how to make a color composite image using 3 different images:

```
> ma b
> pa imageb1.fits
> ma g
> pa imageb3.fits
> ma r
> pa imageb4.fits
```

First, we have selected the **blue** plane, and painted **imageb1.fits** into it. Next, we selected the **green** plane, and painted **imageb3.fits** into it. Then, we selected the **red** plane, and painted **imageb4.fits** into it.

You could also select a color plane and stretch the color table associated with *that plane only*, if you want.

```
> ma b
> st 50 200
```

Or, you could select *all* three image planes at once, and stretch the color tables simultaneously:

```
> ma a
> st
```

2.3.11 Hardcopy

PostScript Once you have manipulated an image on the screen, you may want to keep a paper copy of it. You can make grey scale (or color) plots of the image as it appears on the screen. For this process we use the commonly available X program **xv**.

Creating a print of an image involves 3 steps:

- Display the image
- Create a screen dump file of the image
- Create the PostScript file of the screen dumped image

Display the Image To display an image on the screen, use the **paint** command in **skyview**:

```
> pa image.fits
```

A new window pops up with the image appearing in it. Manipulate the color table stretch (with the **st** function) until the features in the image appear the way you want them to look. Add graphics as you like.

Convert the screen image to a file Use the **screen dump** command to create a file. The **ppmbin** format is recognized by **xv**; it retains full color information and is quite compact.

```
> sd ppmbin sdout.ppm
```

You will be prompted to click the mouse in the window of the image you wish to dump. Once you have clicked the mouse in the desired window, the screen dump function will write the image, in the **ppmbin** format, to the output file you specified.

If the image is bigger than your screen, or for some reason you can't fit the entire image in the window, you can use the **fullimage** keyword to tell **sd** to dump the entire image, instead of only the portion that is displayed.

```
> sd fullimage ppmbin sdout.ppm
```

2.3.12 Create the Plot

This step runs outside of **skyview**, so run it in a separate window. A program called **xv** will convert a ppmbin file into a set of PostScript instructions and send them to a PostScript printer, producing a picture of the image.

```
% xv sdout.ppm
```

Within **xv**, you click the image with the right mouse button to bring up a control panel. then click "Print", and follow the popup panels to specify color vs. greyscale, which printer, scaling, etc. Program **xv** can also be used to convert the image to a number of other image formats.

JPEG/GIF/PPM Format The **screen dump** command can also be used to save a displayed image into a JPEG, GIF or PPM format file. Simply specify the format you want, and the output file name on the command line, like this:

```
> sd gif orion.gif
```

Again you will be prompted to move the mouse into the window you wish to dump, and then click.

2.4 Image Analysis and Data Inspection Tools

Before you start your data analysis with **skyview**, there are a few basics you'll want to know about first.

One of the most important things you should be aware of is that you should **never** move the mouse between windows while in the middle of typing in a **skyview** command. The program will not recognize the command, and you will have to re-type it.

2.4.1 Coordinate Systems

Coordinates may be specified in several forms. If a command description does not specifically state that a certain coordinate system be used, any of the following systems you prefer to work in is fine:

ecl	Ecliptic	(lon/lat)
equ	Equatorial	(lon/lat)
gal	Galactic	(lon/lat)
im	offset from reference pixel identified by CRPIX1 and CRPIX2	(sample/line)
iraf	offset from first pixel in file(first pixel=1,1)	(IRAF pixel/line)

The command

```
> cs
```

will tell you which coordinate system is currently in use. The default is *equatorial.*

```
> cs gal
```

will change the coordinate system to **galactic** degrees.

2.4.2 Current Image–Changing from one image to another

When going back and forth between multiple displayed images, the **ci** (current image) command is very helpful.

```
> ci ori.fits
```

will tell **skyview** that regardless of what order the images were painted on the screen, subsequent commands refer to the image **ori.fits**. Of course, **ori.fits** must have already been painted.

Any mouse function performed in an image window automatically changes the **current image** to the image painted in that window.

2.4.3 Graphics Overlay Colors

The default graphics overlay color for the image analysis and data inspection tools can be set to use any X-color within the table of allowable colors in your X-Windows installation (Typically, the file /usr/lib/X11/rgb.txt contains the list). The defaults given in the explanations that follow can be changed by using the `set` command. For example,

```
> set color BlueViolet
```

tells all the subsequent commands that when displaying graphics on the image, they should be blue-violet.

When there is a **color** option to a command (currently only in the **annotate** and **mark** commands), an X-color can be specified on the command line, as well. If no **color** is specified, but the default overlay color has been **set**, that default color is used. If no color is specified, and the default overlay color has **not** been set, the overlay color used is whatever the function default is–normally green.

2.4.4 Erase

The **erase** command is used to erase one or all of the displayed images, one or all graphics overlay colors, or overlay 'bundles'.

```
> er
```

with the cursor in a particular image display window will erase only that image. Another way to erase only one window is to specify the frame number with the erase command. The frame number is always given in parentheses on the title bar of the image window. For example,

```
> er 2
```

erases only the window named **(frame 2)**.

```
> er
```

with the cursor in the **skyview** command window will erase ALL displayed images.

To erase a graphics overlay color, just specify the color on the erase command line:

```
> er purple
```

erases all the purple overlays currently being displayed.

If you want to erase all the graphics overlays,

```
> er ao
```

stands for '**e**rase **a**ll **o**verlays'.

An overlay bundle is a group of graphics overlays displayed on the screen with one particular execution of a **skyview** command. Each graphics drawing made on the screen is kept track of by the program, so it can be erased simply by clicking the left button on the mouse, in the same way you would use the **pick** command. Some of the things you can erase: The crosshairs left on the image from a **pick**, marks left on the image by the **mark** command, or a defined area. With any graphics bundle you wish to erase, you enter the same command:

```
> er bundle
```

Move the mouse to place the cursor directly on top of the graphics you wish to erase. Clicking the left button erases only that bundle. You can continue erasing bundles until you have erased all you want. Then, either click the middle mouse button, or hit the 'Return' key on the keyboard.

2.4.5 The Magnifier Window

The **mag** command is used to turn the **magnifier window** on and off. This window allows you to get a close-up view of a small area around where the cursor is currently positioned.

```
> mag on
```

turns on the magnifier window. By default, the image in the magnifier window is 8 times the zoom factor of the image window. If you wish to change that factor, simply type the magnification factor you would like to see in the magnifier window as a parameter of the **mag on** command:

```
> mag on 12
```

Now the magnification will be **12** times the zoom factor of the image until it is changed again. *So, guess what command you use to turn it off?*

```
> mag off
```

Of course!

2.4.6 Take files: Execute a Group of skyview Commands from a File

The **take** command allows you to execute a sequence of **skyview** commands contained within a file[7]. First, you have to create the command file. Using the editor of your choice, create a file which contains all the **skyview** commands you wish to execute. For example, I want to **paint** each of the four IRAS wavelength survey-coadded images of M31 in a separate window, get the fluxes of 9 known positions in each wavelength, and get a **contour plot** of the 100-micron image. Since I want fluxes for the same 9 positions on each image, I will create 10 **aliases**. I will alias each position's coordinates to a separate alias name. Then I will create a tenth alias which will do a **pick** on each position. I will, of course, keep all this information in my **session history file**, and print it out when I'm all done. I'll name the command file **m31.tak**. This is what my **take** file named **m31.tak** would look like:

```
sh on create
fr adv
er;ra 1% 99%; pa m31b1;pa m31b2;pa m31b3;pa m31b4
al c1 "0h39m58.1s 41d01m29s"
al c2 "0h40m13.4s 40d58m44s"
al c3 "0h39m38.8s 41d04m56s"
al c4 "0h40m12.1s 41d08m38s"
al c5 "0h41m33.5s 41d18m55s"
al c6 "0h40m53.5s 41d08m7s"
al c7 "0h38m36.4s 40d33m27s"
al c8 "0h38m19.1s 40d47m30s"
al c9 "0h42m50.8s 41d36m53s"
al pcx "pi c1;pi c2;pi c3;pi c4;pi c5;pi c6;pi c7;pi c8;pi c9"
ci m31b1;pcx
ci m31b2;pcx
ci m31b3;pcx
ci m31b4;pcx
co m31b4
ph;unix
```

Now, to execute this file, I just start up the skyview program, and type:

```
> ta m31.tak
```

All the commands will execute automatically, and the printout of my history file and contour plot will probably be coming off the printer by the time I walk down to the computer room to get it!

---

[7]The **take** file is just a group of commands in a file that **skyview** reads. If you are interested in calling **skyview** from a program, see **Section 3**.

Here’s another example: I want to define 5 concentric circles around the same position in all four IRAS wavelength images for the Large Magellanic Cloud, and examine each one. I will name the take file **lmc.tak**. Along with the line numbers which I’ve added for reference, here’s what **lmc.tak** will look like:

```
<1> sh on
<2> er;ra 1% 99%;fr adv; pa lmcb1;pa lmcb2;pa lmcb3;pa lmcb4
<3> ci lmcb1
<4> da 1 li;ci 250;5h15m -68d45m
<5> ex 1
<6> da 2 li;ci 200;5h15m -68d45m
<7> ex 2
<8> da 3 li;ci 150;5h15m -68d45m
<9> ex 3
<10> da 4 li;ci 100;5h15m -68d45m
<11> ex 4
<12> da 5 li;ci 50;5h15m -68d45m
<13> ex 5
<14> ci lmcb2
<15> ex 1;ex 2;ex 3;ex 4;ex 5
<16> ci lmcb3
<17> ex 1;ex 2;ex 3;ex 4;ex 5
<18> ci lmcb4
<19> ex 1;ex 2;ex 3;ex 4;ex 5
<20> ph;unix
```

- Line 1 turns on the session history logging.
- Line 2 turns on the frame advance option and paints each image with a range from 1 to 99 percent. With the frame advance on, each image is displayed in a separate window.
- Line 3 sets the current image (i.e., tells the program which image should be used on all future functions).
- Line 4 is where area #1 gets defined, and line 5 is where it gets examined.
- Lines 6 through 13 define and examine the other 4 areas.
- Line 14 sets the current image to the band-2 image.
- Line 15 examines the same 5 areas on the next image.
- We then proceed through the other two images in lines 16 through 19, and on line 20 the session history file is printed, and the **skyview** program is ended.