Siril user's guide

F. Meyer
dulle@free.fr

September 3, 2005

SIRIL is an abbreviation for nothing except IRIS reversed with a trailing L added for linux(though Siril has nothing linux-specific). This document refers to siril v0.4.

1 Who will need Siril?

The amateur astronomer who wants a reasonably user- friendly interface but who also wants to keep an eye on the processings, and keep those processings under control. Basically, it is more IRIS-like than Registax- like.

2 Philosophy of operation

You have a working image space, on which basic operations are done in memory. For example, commands like the soper, ioper, crop, wavelet commands act on that working space and not on image files.

From the programmer's point of view, this is represented by the fits structure pointed to by `com.g`. `com` is an all-purpose structure, instantiated only once and that is basically intended to provide easy access to a set of global variables that may be needed by the different modules of the software; and the working image space is one of those variables.

3 Command summary

3.1 load

`load filename [rvbag]`

`load` loads the fits image `filename` ; it first attempts to load `filename`, then `filename.fit` and finally `filename.fits`, aborting if none of these are found. These scheme is appliable to every siril command implying reading files. Fits headers MIPS-HI and MIPS-LO are read and their values given to the current viewing levels (the second, optional argument is reserved for a future use).

3.2 save

`save filename [rvbag]`

`save` writes the currently displayed image to `filename.fit`. Fits headers MIPS-HI and MIPS-LO are added with values corresponding to the current viewing levels (the second, optional argument is reserved for a future use).

3.3 ioper commands : `iadd,isub,imul,idiv`

`ioper filename`

ioper (i for 'image') commands respectively adds, substract, multiplies or divides current image by the fits image filename, the resulting image is displayed but not saved.

3.4 soper commands : `sadd,ssub,smul,sdiv`

`soper scalar`

soper commands ('s' for scalar) respectively adds, substract, multiplies or divides current image by `scalar`, the resulting image is displayed but not saved.

3.5 ioper2 commands : iadd2,isub2,imul2,idiv2

```
ioper2 genname outname filename number
```

ioper2 commands respectively adds, substract, multiplies or divides the sequence of `number` images with generic name `gename` by the fits image filename, the resulting images are stored in the corresponding sequence with generic name `outname`.

3.6 soper commands : sadd2,ssub2,smul2,sdiv2

```
soper2 genname outname scalar number
```

soper2 commands respectively adds, substract, multiplies or divides the sequence of `number` images with generic name `gename` by the given scalar, the resulting images are stored in the corresponding sequence with generic name `outname`.

3.7 fdiv

```
fdiv filename scalar
```

`fdiv` divides current image by the image `filename` and multiplies the result by `scalar` ; this cant be done with idiv and smul because floats must be used here.

3.8 log

```
log
```

`log` computes and applies a logarithmic scale to the current image.

3.9 threshlo, threshhi, thresh

These are threshold functions:

- `threshlo 0` replaces negative values with 0s;
- `threshhi 1000` replaces values above 1000 with 1000;
- `thresh 0 1000` does both.

3.10 nozero

```
nozero level
```

`nozero level` replaces null values by `level` values. Useful before an `idiv` or `fdiv` operation.

3.11 ddp

```
ddp level coef sigma
```

`ddp` performs a DDP (digital development processing) as described first by Kunihiko Okano. This implementation is the one described in IRIS:
http://www.astrosurf.org/buil/us/iris/iris5.htm
It combines a linear distribution on low levels (below `level`) and a non-linear on high levels. It uses a gaussian filter of sigma `sigma` and multiplies the resulting image by `coef`.

3.12 register

```
register genname number [x y width height]
```

`register` register the given image sequence as a function of the specified image region (either working space mouse selection or as command line parameters, as specified in the prototype. `register` creates a file with calculated image shifts and image quality. This file is named `genname.shift`, and can be user for further operations, typically `composit`, that will read the shift file on the fly to produce an aligned stacking of the initial image set.

3.13 `composit` also known as `stack`

`conposit genname number [shiftfile]`

This command stacks the given image sequence of n images to the current working image. If shiftfile is specified, x and y shift values are read on the fly (the sequence must have been registered first); if no shiftfile is given, stacking is done with no shift.

3.14 `shift`

`shift x y`

This command shifts current image by x and y.

3.15 `shift2`

`shift2 genname outname number x y`

This command shifts genname image sequence by x and y, outputting the result to the outname sequence.

3.16 `rshift2`

`rshift2 genname outname number [shiftfile]`

This command shifts genname image sequence by x and y, outputting the result to the outname sequence. x and y shift are read on the fly from shiftfile (the sequence must have been registered first).

3.17 `medstack`

`medstack genname number`

This commands puts in working image the median stack of the given sequence. `number` is preferably odd.

3.18 unsharp, unsharp2

```
unsharp sigma coef
```

unsharp applies to the working image an unsharp mask with sigma **sigma** and coefficient **coef**.

```
unsharp2 sigma coef genname outname number
```

unsharp2 is the sequence version of tt unsharp, applying **unsharp** to the sequence **genname** the result goes in the sequence **outname**.

3.19 gauss

```
gauss sigma
```

gauss performs a gaussian filter with the given sigma.

3.20 wavelet

```
wavelet plan_number type (1=linear 2=spline)
```

computes the wavelet transform on **plan_number** plans using linear (**type = 1**) or bspline (**type=2**) version of the 'a trous' algorithm. The result is stored in a file as a structure containing the planes, ready for weighted reconstruction with **wrecons**.

3.21 wrecons

```
wrecons c1 c2 ... cn
```

wrecons reconstruct to current image from the planes previously computed with **wavelet** and weighted with coefficients **c1, c2, ... cn** according to the number of planes used for wavelet transform.

3.22 animate

```
animate genname start_image end_image delay stride
```

animate animates the **genname** sequence of images, starting at image number **start_image**, ending at **end_image** with an optional delay between images and optional stride (displays 1 image each **stride** images). **animate** loops until interrupted by **ESC** key.