ALSVID
Algorithms for Visualization and Processing of Image Data

John Kielkopf & Karen Collins

December 18, 2009

1 Introduction

The programs described here result from the development of methodology for processing image data with high precision, in a flexible command line system useful for astronomical and laboratory image processing. This is a work in progress, with its roots in basic code for handling CCD images acquired under Linux, and its branches in precision photometry, spectroscopy, and 3-D visualization. We have adopted the acronym **ALSVID** both to have a simple name for many somewhat disconnected components, and to recognize that astronomers and physicists alike need a swift and sure means of reaching their goal of understanding what the data contain. Alsvid is an old Norse name meaning "Very Quick" for one of the two horses who pull the Chariot of Sol across the sky.

The programs which are presently offered are Bash scripts and C-language functions to perform operations on Flexible Image Transport System (FITS) data files under Linux. These programs depend on the CFITSIO library which must be installed separately. The programs and libraries are included in XmCCD and would normally be installed if that package is used for image acquisition. However, they are useful in a broader context, and are available separately as well. The executables are statically linked, and were compiled under OpenSuse 11.2. They should run on most current Linux distributions. The scripts require the Bash shell, but they may also serve as a model for other scripting systems.

These image processing programs and scripts are intended for use alongside other Open Source code: SAOImage ds9 and ImageJ for image display, SWARP for handling large image sets, astrometry.net for coordinates, and GRACE for graphics. We currently use DAOphot as well, but it is not Open Source. ImageJ has plug-ins for photometry that produce good results but require real-time interaction during processing. We expect to have Open Source photometry support within this system soon that will allow batch processing of large data sets, primarily for automated handling of exoplanet transit photometry. These programs are independent of the MIDAS, IRAF, or IDL systems that may achieve the same results in their own unique environments.

Three-dimensional visualization tools for the analysis of very large image data sets that include a third (or fourth) dimension of time (or spectrum) are under development to make use of multi-core parallel image processing hardware now widely available in desktop systems.

2 File Storage

Images are held in FITS files which contain a header and binary data. Data are kept with the headers in single files, and the header is updated with the history of the processing. Typical header information at a minimum identifies the target, the acquistion instrument, the exposure duration, filters, and the time at which the exposure was taken. Exposure time in the header may be used in scaling dark files for dark subtraction, and in calibration for photometry. In the case of astronomical data, additional *World Coordinate System* (WCS) data may be included to relate each pixel to a particular direction on the sky. WCS positions are used to identify known objects in photometric reduction scripts.

Camera data are expected in 16-bit unsigned integers with a bias to insure that no pixels have negative data. A pixel will therefore range from 0 to 65535. With processing it is possible that values will, on average, go below zero. When several images are added, the information content may exceed 16-bits as well. Therefore, all processed images are stored is 32-bit floating point data. While this increases the file size, since the original data are in the smallest possible files, and the reduced data are selected and may be averages or sums of several original files, on the whole the resulting data set size is still acceptable. A disadvantage of using floating point image

files is that, with a size that is 2 times larger that the same image as in integer format, the read time of the file is about 2 times longer. When many files are processed off of the disk this is a signifcant increase in overhead. Therefore, for processing large sets of many files in the same way we will include programs that minimize the reading of files.

3 Command Line Programs

imarith

Perform arithmetic operation on 2 input images creating a new output image. Supported arithmetic operators are add, sub, mul, div (first character required

Usage:

imarith image1 image2 oper outimage

Example:

imarith in1.fits in2.fits a out.fits - add the 2 files

imaverage

Find the average image from several images and output a 32-bit floating point image.

Usage:

imaverage out in1 in2 ...

Example:

imaverage flatda.fits flat1d.fits flat2d.fits flat3d.fits

imbin

Bin an image by a factor

Usage:

imbin input.fits factor output.fits

Example:

for a 1024x1024
imbin image.fits 4 newimage.fits
generates a 256x256 newimage.fits

imbuild

Build a 2D 16-bit image file from 1D data files

Usage:

imbuild nskip maxsignal out.fits in1.dat in2.dat ...

Example:

imbuild 16 16000 newimage.fits data1.dat data2.dat ... data256.dat
generates an image with 256 rows
each data file is expected to have 16 header lines
output image will be scaled to a maximum of 16000

imcols

Write a data file with a sum of fits image columns

Usage:

imcols image.fits out.dat start stop

Example:

imcols image.fits cols.dat 100 200

imdark

Perform dark subtraction and output a 32-bit floating point image

Usage:

imdark in.fits dark.fits out.fits

Example:

imdark image.fits dark.fits imaged.fits

imdarki

Perform dark subtraction and output a 16-bit integer image

Usage:

imdarki in.fits dark.fits out.fits

Example:

imdarki image.fits dark.fits imaged.fits

imdarkx

Perform exposure-time scaled dark subtraction and output a 32-bit floating point image.

Usage:

imdarkx in.fits dark.fits out.fits

Example:

imdarkx image.fits dark.fits imaged.fits

imflat

Perform flat field correction for a floating point flat field assumed to have a mean value of 1.0, and output a 32-bit floating point image.

Usage:

imflat in.fits flat.fits out.fits

Example:

imflat image.fits flat.fits imagef.fits

imflati

Perform flat field correction, normalize by the mean value of flat image, and output a 16-bit integer image.

Usage:

imflati in.fits flat.fits out.fits

Example:

imflati image.fits flat.fits imagef.fits

imlevel

Remove the average gradient from an image Output 32-bit floating point image

Usage:

imlevel in.fits out.fits

Example:

imlevel flat.fit flatflat.fit

imlisthead

List the FITS header keywords in a single extension. If ext is not given, list the keywords in all the extensions.

Usage:

imlisthead filename[ext]

Examples:

listhead file.fits – list every header in the file
listhead file.fits[0] – list primary array header
listhead file.fits[2] – list header of 2nd extension
listhead file.fits+2 – same as above
listhead file.fits[GTI] – list header of GTI extension

immedian

Find the median image from several images.

Usage:

immedian out.fits in1.fits in2.fits ...

Example:

immedian flatdm.fit flat1d.fit flat2d.fit flat3d.fit

immodhead

Write or modify value of a header keyword and print current value if new value not specified.

Usage:

immodhead filename[ext] keyword newvalue

Examples:

modhead file.fits dec – list the DEC keyword
modhead file.fits dec 30.0 – set $DEC = 30.0$

imnorm

Perform image normalization to a mean value of 1.0 .

Usage:

imnorm in.fits out.fits

Example:

imnorm image.fits imagen.fits

imrows

Write a data file with a sum of fits image rows.

Usage:

imrows in.fits out.fits start stop

Example:

imrows image.fits rows.dat 100 200

imscale

Scale signal s for each pixel in a fits image to

$$a_0 + a_1 \cdot s + a_2 \cdot s^2$$

Usage:

imscale inpu.fits output.fits a0 a1 a2

Example:

imscale image.fit linear.fit 0. 1. 1.123e-7

4 Batch Programs

imdarkall

Create a master average of all dark exposures and perform a dark subtraction on all images.

- No input parameters are required for imdarkall.
- Place the image files in a selected directory.
- Place the dark files in the subdirectory `./Darks/`.
- Run imdarkall.
- An average master dark will be saved in `./Darks/dark.fits`.
- All images in the working directory will be dark subtracted.
- The results will be in `./Dark_Subtracted/filename_d.fits`.

imdarkflatall

Create a master average of all dark exposures, perform a dark subtraction on all images, and perform a flat correction for all images.

- No input parameters are required for imdarkall.
- Place the image files in a selected directory.
- Place the dark files in the subdirectory `./Darks/`.
- Place the master flat file `./Flats/flat.fits`.
- Run imdarkflatall.
- An average master dark will be saved in `./Darks/dark.fits`.
- All images in the working directory will be dark and flat corrected.
- The dark-subtracted results will be in `./Dark_Subtracted/filename_d.fits`.
- The flat-corrected results will be in `./DarkFlat/filename_df.fits`.