The IRAF CCD Reduction Package -- CCDRED

Francisco Valdes

IRAF Group - Central Computer Services
National Optical Astronomy Observatories††
P.O. Box 26732, Tucson, Arizona 85726
September 1987

ABSTRACT

The IRAF† CCD reduction package, **ccdred**, provides tools for the easy and efficient reduction of CCD images. The standard reduction operations are replacement of bad pixels, subtraction of an overscan or prescan bias, subtraction of a zero level image, subtraction of a dark count image, division by a flat field calibration image, division by an illumination correction, subtraction of a fringe image, and trimming unwanted lines or columns. Another common operation provided by the package is scaling and combining images with a number of algorithms for rejecting cosmic rays. Data in the image header is used to make the reductions largely automated and self-documenting though the package may still be used in the absence of this data. Also a translation mechanism is used to relate image header parameters to those used by the package to allow data from a variety of observatories and instruments to be processed. This paper describes the design goals for the package and the main tasks and algorithms which satisfy these goals.

This paper is to be published as part of the proceedings of the Santa Cruz Summer Workshop in Astronomy and Astrophysics, *Instrumentation for Ground-Based Optical Astronomy: Present and Future*, edited by Lloyd B. Robinson and published by Springer-Verlag.

†Image Reduction and Analysis Facility (IRAF), a software system distributed by the National Optical Astronomy Observatories (NOAO).

July 2, 1990

---

††Operated by the Association of Universities for Research in Astronomy, Inc. under cooperative agreement with the National Science Foundation.

The IRAF CCD Reduction Package -- CCDRED

Francisco Valdes

IRAF Group - Central Computer Services
National Optical Astronomy Observatories††
P.O. Box 26732, Tucson, Arizona 85726
September 1987

1. Introduction

The IRAF CCD reduction package, **ccdred**, provides tools for performing the standard instrumental corrections and calibrations to CCD images. The major design goals were:

- To be easy to use
- To be largely automated
- To be image header driven if the data allows
- To be usable for a variety of instruments and observatories
- To be efficient and capable of processing large volumes of data

This paper describes the important tasks and algorithms and shows how these design goals were met. It is not intended to describe every task, parameter, and usage in detail; the package has full documentation on each task plus a user's guide.

The standard CCD correction and calibration operations performed are replacement of bad columns and lines by interpolation from neighboring columns and lines, subtraction of a bias level determined from overscan or prescan columns or lines, subtraction of a zero level using a zero length exposure calibration image, subtraction of a dark count calibration image appropriately scaled to the dark time exposure of the image, division by a scaled flat field calibration image, division by an illumination image (derived from a blank sky image), subtraction of a scaled fringe image (also derived from a blank sky image), and trimming the image of unwanted lines or columns such as the overscan strip. The processing may change the pixel datatype on disk (IRAF allows seven image datatypes); usually from 16 bit integer to real format. Two special operations are also supported for scan mode and one dimensional zero level and flat field calibrations; i.e. the same calibration is applied to each CCD readout line. Any set of operations may be done simultaneously over a list of images in a highly efficient manner. The reduction operations are recorded in the image header and may also be logged on the terminal and in a log file.

The package also provides tools for combining multiple exposures of object and calibration images to improve the statistical accuracy of the observations and to remove transient bad pixels. The combining operation scales images of different exposure times, adjusts for variable sky background, statistically weights the images by their signal-to-noise, and provides a number of useful algorithms for detecting and rejecting transient bad pixels.

Other tasks are provided for listing reduction information about the images, deriving secondary calibration images (such as sky corrected flat fields or illumination correction images), and easily setting the package parameters for different instruments.

This paper is organized as follows. There is a section giving an overview of how the package is used to reduce CCD data. This gives the user's perspective and illustrates the general ease of use. The next section describes many of the features of the package contributing to

---

††Operated by the Association of Universities for Research in Astronomy, Inc. under cooperative agreement with the National Science Foundation.

its ease of use, automation, and generality. The next two sections describe the major tools and algorithms in some detail. This includes discussions about achieving high efficiency. Finally the status of the package and its use at NOAO is given. References to additional documentation about IRAF and the CCD reduction package and an appendix listing the individual tasks in the package are found at the end of this paper.

2. A User's Overview

This section provides an overview of reducing data with the IRAF CCD reduction package. There are many variations in usage depending on the type of data, whether the image headers contain information about the data which may be used by the tasks, and the scientific goal. Only a brief example is given. A more complete discussion of usage and examples is given in *A User's Guide to the IRAF CCDRED Package.* The package was developed within the IRAF system and so makes use of all the sophisticated features provided. These features are also summarized here for those not familiar with IRAF since they are an important part of using the package.

Since the IRAF system is widely distributed and runs on a wide variety of computers, the site of the CCD reductions might be at the telescope, a system at the observatory provided for this purpose, or at the user's home computer. The CCD images to be processed are either available immediately as the data is taken, transferred from the data taking computer via a network link (the method adopted at NOAO), or transferred to the reduction computer via a medium such as magnetic tape in FITS format. The flexibility in reduction sites and hardware is one of the virtues of the IRAF-based CCD reduction package.

IRAF tasks typically have a number of parameters which give the user control over most aspects of the program. This is possible since the parameters are kept in parameter files so that the user need not enter a large number of parameters every time the task is run. The user may change any of these parameters as desired in several ways, such as by explicit assignment and using an easy to learn and use, fill-in-the-value type of screen editor. The parameter values are *learned* so that once a user sets the values they are maintained until the user changes them again; even between login sessions.

The first step in using the CCD reduction package is to set the default processing parameters for the data to be reduced. These parameters include a database file describing the image header keyword translations and default values, the processing operations desired (operations required vary with instrument and observer), the calibration image names, and certain special parameters for special types of observations such as scan mode. A special script task (a command procedure) is available to automatically set the default values, given the instrument name, to standard values defined by the support staff. Identifying the instrument in this way may be all the novice user need do though most people quickly learn to adjust parameters at will.

As an example suppose there is an instrument identified as `rca4m` for an RCA CCD at the NOAO 4 meter telescope. The user gives the command

```
cl> setinstrument rca4m
```

which sets the default parameters to values suggested by the support staff for this instrument. The user may then change these suggested values if desired. In this example the processing switches are set to perform overscan bias subtraction, zero level image subtraction, flat fielding, and trimming.

The NOAO image headers contain information identifying the type of image, such as object, zero level, and flat field, the filter used to match flat fields with object images, the location of the overscan bias data, the trim size for the data, and whether the image has been processed. With this information the user need not worry about selecting images, pairing object images with calibration images, or inadvertently reprocessing an image.

The first step is to combine multiple zero level and flat field observations to reduce the

effects of statistical noise. This is done by the commands

```
cl> zerocombine *.imh
cl> flatcombine *.imh
```

The "cl> " is the IRAF command language prompt. The first command says look through all the images and combine the zero level images. The second command says look through all the images and combine the flat field images by filter. What could be simpler? Some *hidden* (default) parameters the user may modify are the combined image name, whether to process the images first, and the type of combining algorithm to use.

The next step is to process the images using the combined calibration images. The command is

```
cl> ccdproc *.imh
```

This command says look through all the images, find the object images, find the overscan data based on the image header and subtract the bias, subtract the zero level calibration image, divide by the flat field calibration image, and trim the bias data and edge lines and columns. During this operation the task recognizes that the zero level and flat field calibration images have not been processed and automatically processes them when they are needed. The log output of this task, which may be to the terminal, to a file, or both, shows how this works.

```
ccd003: Jun  1 15:12 Trim data section is [3:510,3:510]
ccd003: Jun  1 15:12 Overscan section is [520:540,*], mean=485.0
Dark:   Jun  1 15:12 Trim data section is [3:510,3:510]
Dark:   Jun  1 15:13 Overscan section is [520:540,*], mean=484.6
ccd003: Jun  1 15:13 Dark count image is Dark.imh
FlatV:  Jun  1 15:13 Trim data section is [3:510,3:510]
FlatV:  Jun  1 15:14 Overscan section is [520:540,*], mean=486.4
ccd003: Jun  1 15:15 Flat field image is FlatV.imh, scale=138.2
ccd004: Jun  1 15:16 Trim data section is [3:510,3:510]
ccd004: Jun  1 15:16 Overscan section is [520:540,*], mean=485.2
ccd004: Jun  1 15:16 Dark count image is Dark.imh
ccd004: Jun  1 15:16 Flat field image is FlatV.imh, scale=138.2
                <... more ...>
ccd013: Jun  1 15:22 Trim data section is [3:510,3:510]
ccd013: Jun  1 15:23 Overscan section is [520:540,*], mean=482.4
ccd013: Jun  1 15:23 Dark count image is Dark.imh
FlatB:  Jun  1 15:23 Trim data section is [3:510,3:510]
FlatB:  Jun  1 15:23 Overscan section is [520:540,*], mean=486.4
ccd013: Jun  1 15:24 Flat field image is FlatB.imh, scale=132.3
                <... more ...>
```

The log gives the name of the image and a time stamp for each entry. The first image is ccd003. It is to be trimmed to the specified size given as an *image section*, an array notation used commonly in IRAF to specify subsections of images. The location of the overscan data is also given by an image section which, in this case, was found in the image header. The mean bias level of the overscan is also logged though the overscan is actually a function of the readout line with the order of the function selected by the user.

When the task comes to subtracting the zero level image it first notes that the calibration image has not been processed and switches to processing the zero level image. Since it knows it is a zero level image the task does not attempt to zero level or flat field correct this image. After the zero level image has been processed the task returns to the object image only to find that the flat field image also has not been processed. It determines that the object image was

obtained with a V filter and selects the flat field image having the same filter. The flat field image is processed through the zero level correction and then the task again returns to the object image, ccd003, which it finishes processing.

The next image, ccd004, is also a V filter observation. Since the zero level and V filter flat field have been processed the object image is processed directly. This continues for all the object images except for a detour to process the B filter flat field when the task first encounters a B filter object image.

In summary, the basic usage of the CCD reduction package is quite simple. First, the instrument is identified and some parameters for the data are set. Calibration images are then combined if needed. Finally, the processing is done with the simple command

```
cl> ccdproc *.imh&
```

where the processing is performed as a *background job* in this example. This simplicity was a major goal of the package.

3. Features of the Package

This section describes some of the special features of the package which contribute to its ease of use, generality, and efficiency. The major criteria for ease of use are to minimize the user's record keeping involving input and output image names, the types of images, subset parameters such as filters which must be kept separate, and the state of processing of each image. The goal is to allow input images to be specified using simple wildcards, such as "*.imh" to specify all images, with the knowledge that the task will only operate on images for which it makes sense. To accomplish this the tasks must be able to determine the type of image, subset, and the state of processing from the image itself. This is done by making use of image header parameters.

For generality the package does not require any image header information except the exposure time. It is really not very much more difficult to reduce such data. Mainly, the user must be more explicit about specifying images and setting task parameters or add the information to the image headers. Some default header information may also be set in the image header translation file (discussed below).

One important image header parameter is the image type. This discriminates between object images and various types of calibration images such as flat field, zero level, dark count, comparison arcs, illumination, and fringe images. This information is used in two ways. For most of the tasks the user may select that only one type of image be considered. Thus, all the flat field images may be selected for combining or only the processing status of the object images be listed. The second usage is to allow the processing tasks to identify the standard calibration images and apply only those operations which make sense. For example, flat field images are not divided by a flat field. This allows the user to set the processing operations desired for the object images without fear of misprocessing the calibration images. The image type is also used to automatically select calibration images from a list of images to be processed instead of explicitly identifying them.

A related parameter specifies the subset. For certain operations the images must have a common value for this parameter. This parameter is often the filter but it may also apply to a grating or aperture, for example. The subset parameter is used to identify the appropriate flat field image to apply to an image or to select common flat fields to be combined into a higher quality flat field. This is automatic and the user need not keep track of which image was taken with which filter or grating.

The other important image header parameters are the processing flags. These identify when an image has been processed and also act as a history of the operation including calibration images used and other parameter information. The usage of these parameters is obvious; it allows the user to include processed images in a wildcard list knowing that the processing will

not be repeated and to quickly determine the processing status of the image.

Use of image header parameters often ties the software to the a particular observatory. To maintain generality and usefulness for data other than that at NOAO, the CCD reduction package was designed to provide a translation between parameters requested by the package and those actually found in the image header. This translation is defined in a simple text file which maps one keyword to another and also gives a default value to be used if the image header does not include a value. In addition the translation file maps the arbitrary strings which may identify image types to the standard types which the package recognizes. This is a relatively simple scheme and does not allow for forming combinations or for interpreting values which are not simple such as embedding an exposure time as part of a string. A more complex translation scheme may prove desirable as experience is gained with other types of image header formats, but by then a general header translation ability and/or new image database structure may be a standard IRAF feature.

This feature has proven useful at NOAO. During the course of developing the package the data taking system was modernized by updating keywords and adding new information in the image headers, generally following the lines laid out by the **ccdred** package. However, there is a period of transition and it is also desirable to reduce preexisting data. There are several different formats for this data. The header translation files make coping with these different formats relatively easy.

A fundamental aspect of the package is that the processing modifies the images. In other words, the reduction operations are performed directly on the image. This "feature" further simplifies record keeping, frees the user from having to form unique output image names, and minimizes the amount of disk space required. There are two safety features in this process. First, the modifications do not take effect until the operation is completed on the image. This allows the user to abort the task without leaving the image data in a partially processed state and protects data if the computer crashes. The second feature is that there is a parameter which may be set to make a backup of the input data with a particular prefix; for example "b", "orig", or "imdir$" (a logical directory prefix). This backup feature may be used when there is sufficient disk space, when learning to use the package, or just to be cautious.

In a similar effort to efficiently manage disk space, when combining images into a master object or calibration image, there is an option to delete the input images upon completion of the combining operation. Generally this is desirable when there are many calibration exposures, such as zero level or flat field images, which are not used after they are combined into a final calibration image.

The goal of generality for many instruments at different observatories inherently conflicts with the goal of ease of use. Generality requires many parameters and options. This is feasible in the CCD reduction package, as well as the other IRAF packages, because of the IRAF parameter handling mechanism. In **ccdred** there still remains the problem of setting the parameters appropriately for a particular instrument, image header format, and observatory.

To make this convenient there is a task, **setinstrument**, that, based on an instrument name, runs a setup script for the instrument. An example of this task was given in the previous section. The script may do any type of operation but mainly it sets default parameters. The setup scripts are generally created by the support staff for the instrument. The combination of the setup script and the instrument translation file make the package, in a sense, programmable and achieves the desired instrument/observatory generality with ease of use.

4. CCD Processing

This section describes in some detail how the CCD processing is performed. The task which does the basic CCD processing is call **ccdproc**. From the point of view of usage the task is very simple but a great deal is required to achieve this simplicity. The approach we take in describing the task is to follow the flow of control as the task runs with digressions as appropriate.

The highest level of control is a loop over the input images; all the operations are performed successively on each image. It is common for IRAF tasks which operate on individual images to allow the operation to be repeated automatically over a list of input images. This is important in the **ccdred** package because data sets are often large and the processing is generally the same for each image. It would be tedious to have to give the processing command for each image to be processed. If an error occurs while processing an image the error is printed as a warning and processing continues with the next image. This provides protection primarily against mistyped or nonexistent images.

Before the first image is processed the calibration images are identified. There are two ways to specify calibration images; explicitly via task parameters or implicitly as part of the list of images to be processed. Explicitly identifying calibration images takes precedence over calibration images in the input list. Specifying calibration images as part of the input image list requires that the image types can be determined from the image header. Using the input list provides a mechanism for breaking processing up into sets of images (possibly using files containing the image names for each set) each having their own calibration images. One can, of course, selectively specify input and calibration images, but whenever possible one would like to avoid having to specify explicit images to process since this requires record keeping by the user.

The first step in processing an image is to check that it is of the appropriate image type. The user may select to process images of only one type. Generally this is object images since calibration images are automatically processed as needed. Images which are not of the desired type are skipped and the next image is considered.

A temporary output image is created next. The output pixel datatype on disk may be changed at this point as selected by the user. For example it is common for the raw CCD images to be digitized as 16 bit integers but after calibration it is sometimes desirable to have real format pixels. If no output pixel datatype is specified the output image takes the same pixel datatype as the input image. The processing is done by operating on the input image and writing the results to a temporary output image. When the processing is complete the output image replaces the input image. This gives the effect of processing the images in place but with certain safeguards. If the computer crashes or the processing is interrupted the integrity of the input image is maintained. The reasons for chosing to process the images in this way are to avoid having to generate new image names (a tiresome record keeping process for the user), to minimize disk usage, and generally the unprocessed images are not used once they have been processed. When dealing with large volumes of data these reasons become fairly important. However, the user may specify a backup prefix for the images in which case, once the processing is completed, the original input image is renamed by appending it to the prefix (or with an added digit if a previous backup image of the same name exits) before the processed output image takes the original input name.

The next step is to determine the image geometry. Only a subsection of the raw image may contain the CCD data. If this region is specified by a header parameter then the processing will affect only this region. This allows calibration and other data to be part of the image. Normally, the only other data in a image is overscan or prescan data. The location of this bias data is determined from the image header or from a task parameter (which overrides the image header value). To relate calibration images of different sizes and to allow for readout of only a portion of the CCD detector, a header parameter may relate the image data coordinates to the full CCD coordinates. Application of calibration image data and identifying bad pixel regions via a bad pixel file is done in this CCD coordinate system. The final geometrical information is the region of the input image to be output after processing; an operation called trimming. This is defined by an image header parameter or a task parameter. Trimming of the image is selected by the user. Any or all of this geometry information may be absent from the image and appropriate defaults are used.

Each selected operation which is appropriate for the image type is then considered. If the operation has been performed previously it will not be repeated. If all selected operations have

been performed then the temporary output image is deleted and the input image is left unchanged. The next image is then processed.

For each selected operation to be performed the pertinent data is determined. This consists of such things as the name of the calibration image, scaling factors, the overscan bias function, etc. Note that at this point only the parameters are determined, the operation is not yet performed. This is because operations are not performed sequentially but simultaneously as described below. Consider flat fielding as an example. First the input image is checked to see if it has been flat fielded. Then the flat field calibration image is determined. The flat field image is checked to see if it has been processed. If it has not been processed then it is processed by calling a procedure which is essentially a copy of the main processing program. After the flat field image has been processed, parameters affecting the processing, such as the flat field scale factor (essentially the mean of the flat field image), are determined. A log of the operation is then printed if desired.

Once all the processing operations and parameters have been defined the actual processing begins. One of the key design goals was that the processing be efficient. There are two primary methods used to achieve this goal; separate processing paths for 16 bit integer data and floating point data and simultaneous operations. If the image, the calibration images, and the output image (as selected by the user) are 16 bit integer pixel datatypes then the image data is read and written as integer data. This eliminates internal datatype conversions both during I/O and during computations. However, many operations include use of real factors such as the overscan bias, dark count exposure scaling, and flat field scaling which causes the computation to be done in real arithmetic before the result is stored again as an integer value. In any case there is never any loss of precision except when converting the output pixel to short integer. If any of the images are not integer then a real internal data path is used in which input and output image data are converted to real as necessary.

For each data path the processing proceeds line-by-line. For each line in the output image data region (ignoring pixels outside the data area and pixels which are trimmed) the appropriate input data and calibration data are obtained. The calibration data is determined from the CCD coordinates of the output image and are not necessarily from the same image line or columns. The input data is copied to the output array while applying bad pixel corrections and trimming. The line is then processed using a specially optimized procedure. This procedure applies all operations simultaneously for all combinations of operations. As an example, consider subtracting an overscan bias, subtracting a zero level, and dividing by a flat field. The basic kernel of the task, where the bulk of the CPU time is used, is

```
do i = 1, n
    out[i] = (out[i] - overscan - zero[i]) * flatscale / flat[i]
```

Here, *n* is the number of pixels in the line, *overscan* is the overscan bias value for the line, *zero* is the zero level data from the zero level image, *flatscale* is the mean of the flat field image, and *flat* is the flat field data from the flat field image. Note the operations are not applied sequentially but in a single statement. This is the most efficient method and there is no need for intermediate images.

Though the processing is logically performed line-by-line in the program, the image I/O from the disk is not done this way. The IRAF virtual operating system image interface automatically provides multi-line buffering for maximal I/O efficiency.

In many image processing systems it has been standard to apply operations sequentially over an image. This requires producing intermediate images. Since this is clearly inefficient in terms of I/O it has been the practice to copy the images into main memory and operate upon them there until the final image is ready to be saved. This has led to the perception that in order to be efficient an image processing system *must* store images in memory. This is not true and the IRAF CCD reduction package illustrates this. The CCD processing does not use intermediate images and does not need to keep the entire image in main memory. Furthermore, though

of lesser importance than I/O, the single statement method illustrated above is more efficient than multiple passes through the images even when the images are kept in main memory. Finally, as CCD detectors increase in size and small, fast, and cheap processors become common it is a distinct advantage to not require the large amounts of memory needed to keep entire images in memory.

There is one area in which use of main memory can improve performance and **ccdproc** does take advantage of it if desired. The calibration images usually are the same for many input images. By specifying the maximum amount of memory available for storing images in memory the calibration images may be stored in memory up to that amount. By parameterizing the memory requirement there is no builtin dependence on large memory!

After processing the input image the last steps are to log the operations in the image header using processing keywords and replace the input image by the output image as described earlier. The CCD coordinates of the data are recorded in the header, even if not there previously, to allow further processing on the image after the image has been trimmed.

5. Combining Images

The second important tool in the CCD reduction package is a task to combine many images into a single, higher quality image. While this may also be done with more general image processing tools (the IRAF task **imsum** for example) the **ccdred** tasks include special CCD dependent features such as recognizing the image types and using the image header translation file. Combining images is often done with calibration images, which are easy to obtain in number, where it is important to minimize the statistical noise so as to not affect the object images. Sometimes object images also are combined. The task is called **combine** and there are special versions of this task called **zerocombine, darkcombine**, and **flatcombine** for the standard calibration images.

The task takes a list of input images to be combined. As output there is the combined image, an optional sigma image, and optional log output either to the terminal, to a log file, or both. A subset or subsets of the input images may be selected based on the image type and a subset parameter such as the filter. As with the processing task, this allows selecting images without having to explicitly list each image from a large data set. When combining based on a subset parameter there is an output image, and possibly a sigma image, for each separate subset. The output image pixel datatype may also be changed during combining; usually from 16 bit integer input to real output. The sigma image is the standard deviation of the input images about the output image.

Except for summing the images together, combining images may require correcting for variations between the images due to differing exposure times, sky background, extinctions, and positions. Currently, extinction corrections and registration are not included but scaling and shifting corrections are included. The scaling corrections may be done by exposure times or by computing the mode in each image. Additive shifting is also done by computing the mode in the images. The region of the image in which the mode is computed can be specified but by default the whole image is used. A scaling correction is used when the flux level or sensitivity is varying. The offset correction is used when the sky brightness is varying independently of the object brightness. If the images are not scaled then special data paths combine the images more efficiently.

Except for medianing and summing, the images are combined by averaging. The average may be weighted by

```
weight = (N * scale / mode) ** 2
```

where *N* is the number of images previously combined (the task records the number of images combined in the image header), *scale* is the relative scale (applied by dividing) from the exposure time or mode, and *mode* is the background mode estimate used when adding a variable offset.

The combining operation is the heart of the task. There are a number algorithms which may be used as well as applying statistical weights. The algorithms are used to detect and reject deviant pixels, such as cosmic rays. The choice of algorithm depends on the data, the number of images, and the importance of rejecting cosmic rays. The more complex the algorithm the more time consuming the operation. The list below summarizes the algorithms. Further algorithms may be added in time.

Sum - sum the input images

The input images are combined by summing. Care must be taken not to exceed the range of the 16 bit integer datatype when summing if the output datatype is of this type. Summing is the only algorithm in which scaling and weighting are not used. Also no sigma image is produced.

Average - average the input images

The input images are combined by averaging. The images may be scaled and weighted. There is no pixel rejection. A sigma image is produced if more than one image is combined.

Median - median the input images

The input images are combined by medianing each pixel. Unless the images are at the same exposure level they should be scaled. The sigma image is based on all the input images and is only an approximation to the uncertainty in the median estimates.

Minreject, maxreject, minmaxreject - reject extreme pixels

At each pixel the minimum, maximum, or both are excluded from the average. The images should be scaled and the average may be weighted. The sigma image requires at least two pixels after rejection of the extreme values. These are relatively fast algorithms and are a good choice if there are many images (>15).

Threshold - reject pixels above and below specified thresholds

The input images are combined with pixels above and below specified threshold values (before scaling) excluded. The images may be scaled and the average weighted. The sigma image also has the rejected pixels excluded.

Sigclip - apply a sigma clipping algorithm to each pixel

The input images are combined by applying a sigma clipping algorithm at each pixel. The images should be scaled. This only rejects highly deviant points and so includes more of the data than the median or minimum and maximum algorithms. It requires many images (>10-15) to work effectively. Otherwise the bad pixels bias the sigma significantly. The mean used to determine the sigmas is based on the "minmaxrej" algorithm to eliminate the effects of bad pixels on the mean. Only one iteration is performed and at most one pixel is rejected at each point in the output image. After the deviant pixels are rejected the final mean is computed from all the data. The sigma image excludes the rejected pixels.

Avsigclip - apply a sigma clipping algorithm to each pixel

The input images are combined with a variant of the sigma clipping algorithm which works well with only a few images. The images should be scaled. For each line the mean is first estimated using the "minmaxrej" algorithm. The sigmas at each point in the line are scaled by the square root of the mean, that is a Poisson scaling of the noise is assumed. These sigmas are averaged to get a line estimate of the sigma. Then the sigma at each point in the line is estimated by multiplying the line sigma by the square root of the mean at that point. As with the sigma clipping algorithm only one iteration is performed and at most one pixel is rejected at each point. After the deviant pixels are rejected the file mean is computed from all the data. The sigma image excludes the rejected pixels.

The "avsigclip" algorithm is the best algorithm for rejecting cosmic rays, especially with a small number of images, but it is also the most time consuming. With many images (>10-15) it might be advisable to use one of the other algorithms ("maxreject", "median", "minmaxrej")

because of their greater speed.

This task also has several design features to make it efficient and versatile. There are separate data paths for integer data and real data; as with processing, if all input images and the output image are of the same datatype then the I/O is done with no internal conversions. With mixed datatypes the operations are done as real. Even in the integer path the operations requiring real arithmetic to preserve the accuracy of the calculation are performed in that mode. There is effectively no limit to the number of images which may be combined. Also, the task determines the amount of memory available and buffers the I/O as much as possible. This is a case where operating on images from disk rather than in memory is essential.

6. Status and Conclusion

The initial implementation of the IRAF **ccdred** package was completed in June 1987. It has been in use at the National Optical Astronomy Observatories since April 1987. The package was not distributed with Version 2.5 of IRAF (released in August 1987) but is available as a separate installation upon request. It will be part of future releases of IRAF.

At NOAO the CCD reduction package is available at the telescopes as the data is obtained. This is accomplished by transferring the images from the data taking computer to a Sun workstation (Sun Microsystems, Inc.) initially via tape and later by a direct link. There are several reasons for adopting this architecture. First, the data acquisition system is well established and is dedicated to its real-time function. The second computer was phased in without disrupting the essential operation of the telescopes and if it fails data taking may continue with data being stored on tape. The role of the second computer is to provide faster and more powerful reduction and analysis capability not required in a data acquisition system. In the future it can be more easily updated to follow the state of the art in small computers. As CCD detectors get larger the higher processing speeds will be essential to keep up with the data flow.

By writing the reduction software in the high level, portable, IRAF system the users have the capability to process their data from the basic CCD reductions to a full analysis at the telescope. Furthermore, the same software is widely available on a variety of computers if later processing or reprocessing is desired; staff and visitors at NOAO may also reduce their data at the headquarters facilities. The use of a high level system was also essential in achieving the design goals; it would be difficult to duplicate this complex package without the rich programming environment provided by the IRAF system.

7. References

The following documentation is distributed by the National Optical Astronomy Observatories, Central Computer Services, P.O. Box 26732, Tucson, Arizona, 85726. A comprehensive description of the IRAF system is given in *The IRAF Data Reduction and Analysis System* by Doug Tody (also appearing in *Proceedings of the SPIE - Instrumentation in Astronomy VI*, Vol. 627, 1986). A general guide to using IRAF is *A User's Introduction to the IRAF Command Language* by Peter Shames and Doug Tody. Both these documents are also part of the IRAF documentation distributed with the system.

A somewhat more tutorial description of the **ccdred** package is *A User's Guide to the IRAF CCDRED Package* by the author. Detailed task descriptions and supplementary documentation are given in the on-line help library and are part of the user's guide.

8. Appendix

The current set of tasks making up the IRAF CCD Reduction Package, **ccdred**, are summarized below.

```
badpiximage - Create a bad pixel mask image from a bad pixel file
  ccdgroups - Group CCD images into image lists
   ccdhedit - CCD image header editor
```

```
      ccdlist - List CCD processing information
      ccdproc - Process CCD images
      combine - Combine CCD images
  darkcombine - Combine and process dark count images
  flatcombine - Combine and process flat field images
  mkfringecor - Make fringe correction images from sky images
   mkillumcor - Make flat field illumination correction images
  mkillumflat - Make illumination corrected flat fields
     mkskycor - Make sky illumination correction images
    mkskyflat - Make sky corrected flat field images
setinstrument - Set instrument parameters
  zerocombine - Combine and process zero level images
```