Rectifying and Registering Images Using IRAF

Lisa A. Wells

April 1994

This document is presented as a user's guide to the different methods available in IRAF for rectifying and registering images. The discussion is based on the tasks available in IRAF Version 2.10.3. It is assumed that the reader has some preliminary knowledge about IRAF.

Contents

1 Introduction

This document is presented as a user's guide to the different methods available in IRAF for rectifying and registering images. The discussion is based on the tasks available in IRAF Version 2.10.3. It is assumed that the reader has some preliminary knowledge about IRAF, including the package structure, getting help, listing and editing task parameters, and executing tasks in general. An introductory document to IRAF is listed in Appendix D.

There are many ways of registering images in IRAF, from very simple x and y shifts to the more complex coordinate transformations. The images that are to be registered should be processed through the flat field correction. Spectra should be free of instrumental distortions.

The examples in this manual are exactly that, examples. The user must decide on the naming conventions and make the appropriate substitutions where necessary. We strongly recommend using **lpar** on every task immediately before using it, unless the user is familiar with all of the task's parameters. This is not shown in the examples but is normal practice, even among seasoned IRAF users. Task parameter sets are included in many examples throughout the manual.

Choosing the correct method and some technical issues are addressed in Section 2 as well as choosing a good reference image. Section 3 addresses the very simplest of the registration techniques—axis flipping. Section 4 describes the registration tasks for x and y-axis shifting. Section 5 highlights computing and performing more complex coordinate transformations. Tasks used in acquiring coordinate positions of objects in the images to be shifted, and computing the shift relative to the reference image are described in Section 6. Section 7 gives a brief list and description of the tasks which may be used to combine the registered images. It is left to the user to determine the best option. Using the **imtool** window to simply register images in pixel space for subsequent blinking is addressed in Appendix A. Other useful tasks are described in Appendix B. Additional issues such as some important task parameters and use of the world coordinate system (WCS) are discussed in Appendix C, and a list of useful documentation is given in Appendix D.

2 What Operation Is Best for the Data?

This section summarizes the tasks that are available in IRAF for rectifying and/or registering images.

2.1 A Task Summary

The registration tasks discussed in this manual are listed below.

- **imcopy** - flips an image axis.
- **imtranspose** - transposes axes or rotates images.
- **xregister** - uses cross-correlation techniques to register 1D and 2D images in x and y.
- **shiftlines** - shifts images along the x axis.
- **imshift** - shifts images along both the x and y axes.
- **imalign** - aligns a list of images by shifting along the x and y axes.
- **magnify** - rescales the pixel size by magnifying or miniaturizing images.

- **rotate** - rotates or shifts images.
- **register** - performs a geometric distortion correction using the input coordinate transformation (calculated by the task **geomap**), in any combination of shifting, rotating, or magnifying the images.
- **imlintran** - linearly transforms images using the input parameters to shift, rotate, or magnify in any combination.

Imcopy and **imtranspose** are simple manipulations of images and do not require a great deal of preliminary calculation. They are useful in inverting the x or y axis, or flipping an image along a diagonal axis, i.e., the x=y axis. **Imtranspose** may be used to rotate an image also. See Section 3 for more information and examples.

Of the tasks which perform x and y axis shifts, the easiest to use is **xregister**. This task performs one of several cross correlation algorithms to calculate and perform the shift of one image relative to a reference image. **Xregister** requires only the input images while many of the other shifting tasks require the shift be calculated by other means. Only x shifts are performed using **shiftlines**. The pixel shift must be input in integer or real format. **Imshift** and **imalign** operate on both the axes. The first of these requires the fractional pixel shift for the image be calculated before using the task. The task **imalign** requires initial guesses for the shifts which are used to locate a list of reference objects. The final shift is calculated by subtracting the reference image coordinates from those calculated and averaging. If the images have a suspected rotation difference, more complex tasks need to be used. The pixel scale must be the same in the images for these tasks to work well. Section 4 goes into more detail about these tasks.

Images with well known scales may be resized using the ratio of these values in **magnify**. The x and y axes are handled independently in case the pixel scale differs for the 2 axes. This task does not compute the scale factor between 2 images, it requires that the magnification in x and y be specified. Other methods such as the task **geomap** must be used to calculate the scale values. **Rotate** not only rotates by some specified angle but also applies x and y shifts for images. The rotation is performed for the specified axis position. This task again does not compute the rotation or shift necessary to register two images. The output of the task **geomap** gives the rotation in x and y. Unless the x rotation and y rotation values are the same, this may not be the best way of calculating the rotation for use with **rotate** since it specifies one angle. These tasks are explained in more detail in Section 5.

Complex transformations involving the combination of any mentioned above are handled by the tasks **imlintran** and **register**. These tasks will perform all the operations required at once. It is easiest to use the task **geomap** to compute the transformation since the **register** task takes the output map solution directly. **Imlintran** requires the values be input explicitly in the task parameters. These tasks are also described in Section 5.

2D spectra in general will only have x and y shifts, however if distortions are evident or the spectra do not lie directly along a line or column, then these must be corrected first. The spectra should be transformed to remove distortions such that the dispersion axis and direction agree. Scaling may be necessary with **dispcor** if the wavelength dispersions differ for 2 spectra. Well defined emission, absorption, or sky lines permit the determination of the shift between spectra using the task **xregister**. Caution should be used for infrared spectra. Shifting by fractional pixel values will degrade low signal to noise spectra by the use of interpolation. Use of integer pixel shifts is highly suggested in this case. Some direction is given for registering spectra in Appendix B.4, however there is no standard way of performing registration on spectral images.

2.2 Choosing a Reference Image

The first thing a user must decide upon is what image to use as the reference image. All other images will be registered to the reference. A good reference image is one that has the program object(s) well centered in reference to all the other images and has the best looking stellar profiles for computing centers. Diffuse clouds may be used to calculate shifts between images with some tasks as long as the frames have been taken through the same filter and have about the same intensity levels. The nuclei of galaxies may be used but are not as ideal as stars. Images taken in a grid pattern or by dithering the telescope must have well defined overlapping regions with stellar features. If the overlap is very small then mosaicing is a better option (see Appendix D for a manual reference).

2.3 Technical Issues and Problems

Registering images to increase signal to noise by combining them may not always give the desired result. Images with low signal to noise will be degraded further by the interpolation methods used in the transformation tasks. It is suggested that tests be performed to check the flux conservation.

In calculating the transformation, diffuse objects can grossly affect the mapping in **geomap**. Stellar profiles are smooth and well defined so it is best to use stars for calculating the necessary centers and shifts for an image. A galaxy nucleus may work as well as long as the profile is smooth, but it is highly recommended that HII regions be avoided if possible. This is true for spectra as well, and there must be enough signal to noise for the matching algorithm used.

If images are divided into regions either specified or using the *grid* option while running **xregister**, each section must contain objects which may be used to calculate shifts. The shifts calculated for each region are averaged to define the final values. If one region has only diffuse objects with no stellar images, the shift for that region should be checked carefully.

It has been found that using very high order fitting when calculating the transformation mapping using **geomap** has resulted in *Out Of Memory* errors when using this output with **register** or **imlintran**. Should this occur, try using a lower fitting order in the **geomap** parameters, *xxorder*, *xyorder*, *yxorder*, and *yyorder*.

3 Simple Axis Flipping and Axis Rotation

The simplest image manipulations are those of axis flipping or axis rotation of 90° or −90°. The tasks used here require an input image name and an output image name. Axis flipping can be done using the **imcopy** task as shown below for the x axis, the y axis, and both, respectively:

```
cl> imcopy obj0003[-*,*] obj0003xf
cl> imcopy obj0003[*,-*] obj0003yf
cl> imcopy obj0003[-*,-*] obj0003xyf
```

The last example simply flips the image about both the axes to produce the final image resulting in an image that is rotated by 180°. To exchange or transpose the axes use the task **imtranspose**.

```
cl> imtranspose obj0003 obj0003t
```

Rotating an image by 90° or −90° about the center of the image is done using the following executions of **imtranspose**, respectively:

```
cl> imtranspose obj0003[-*,*] obj0003cw
cl> imtranspose obj0003[*,-*] obj0003ccw
```

This combines reversing one of the axes with the swapping of the x and y axes. The task **rotate** may also be used to perform rotations for a specific angle. See Section 5.2 for more on **rotate**.

4 X and Y Axis Shifts

The following section deals with the tasks that do simple x and y axis shifts of images. Section 6 explains in general how to find positions of objects in the images and calculate the shifts which are input to some of these tasks. As mentioned above, **xregister** does not require that initial estimates or final shifts be calculated. It uses cross correlation routines and is the simplest to use. The other tasks in this section require that the input shift be specified.

- **xregister** - calculates the shifts using cross-correlation techniques and performs the shift.
- **shiftlines** - shifts in x only.
- **imshift** - performs x and y shifts.
- **imalign** - calculates the shifts identically to the task **imcentroid** and performs the shift.

All the registration tasks explained in this section have parameters which define the interpolation type and boundary conditions for manipulating an image. The specific choices are explained in more detail in Appendix C.

4.1 xregister

The **xregister** task is found in the IMAGES package. The images to be registered are input along with the reference image. Image sections for the cross-correlation may be specified either by listing them or using a grid. The individual shifts obtained in each region are averaged to compute a final x and y shift that is to be applied. The subsections specified in the parameter *regions* work better with point sources which will be correlated. The following four cross-correlation algorithms are available:

- **discrete** - computes the correlation coefficient in the form of the covariance divided by the product of the standard deviations.
- **fourier** - calculates the convolution using fast fourier transforms.
- **difference** - computes the error function as a function of x and y position in the search window.
- **file** - does not compute the shift but reads the previously computed shifts from a table defined by the parameter *shifts*.

Each of these functions generates a correlation image which is searched to find the peak using a centering algorithm (see Appendix C). Be aware that one or more bad regions with few or no bright objects can throw off the average shifts in x and y.

```
cl> xregister obj001,obj002 obj003 "[1:201,1:100] [1:201,101:201]" \
output="tobj001,tobj002,tobj003"
or
cl> xregister @list obj003 "grid 2 3" correlation=fourier \
function=parabolic
```

The first example uses two image regions to perform the discrete correlation and renames the output images. The second example uses a grid pattern, 2 in x and 3 in y, with the fourier correlation and the parabolic search for the peaks in the correlation image. If output image names are not specified, then only the shift is calculated and saved in the file defined by *shifts*. The interpolation type and boundary condition for the shift are explained in more detail in Appendix C.

4.2 shiftlines

The task **shiftlines**, found in the IMAGES package in IRAF, does simple x axis shifts. It will shift one image at a time or take an *input* and *output* list provided all of the images are to be shifted by the same amount. Otherwise, each image must be run through individually. This task works well with N-dimensional images also. The x value for the shift is input in the parameter *shift*. This operation may be done in place by setting both parameters to the same image name.

If two images are to be shifted, the first by 3.24 pixels and the second by 1.96 pixels, then do the following:

```
cl> shiftlines obj0013 sobj0003 3.24
cl> shiftlines obj0014 obj0014 1.96 interp=poly5 boundary=constant \
constant=1.0
```

The first line creates a new image called *sobj0013* and uses the default *linear* interpolation and the *nearest* boundary type. The second example overwrites the old image by the new shifted image, using a higher order interpolation function and a constant boundary type. The interpolation type and boundary condition for the shift are explained in more detail in Appendix C.

4.3 imshift

The **imshift** task is found in the IMAGES package and can do simple shifts in both x and y. This task will operate on a single image or on a list of images and the shifts can be the same or different for each image in the list. The text file with a list of shifts for a list of images should have the x shift and y shift in columns 1 and 2 respectively. If a text file is not specified, then the shifts specified in the parameters *xshift* and *yshift* are used. This task does not calculate the shifts, so this must be done first. Methods for doing this are explained in Section 6. The default interpolation type is *linear*, and the default boundary type is *nearest*.

```
cl> imshift obj0004 sobj0004 xshift=1.51 yshift=-2.50
or
cl> imshift @inlist @outlist shifts_file=shift2
```

In the first example, one image is input to the task, the output image is called *sobj0004*, and the shifts are given by the *xshift* and *yshift* parameters. The second example uses file lists to input the image names and a second list supplies the output names. These two files must contain the same number of lines, or an error message is generated. The shifts file contains the input shift values for the images in the object list where the first line in the shifts file corresponds to the first image in the object list. The interpolation type and boundary condition for the shift are explained in more detail in Appendix C.

4.4 imalign

The task **imalign** is found in the PROTO package (this task is really a script task that calls the task **imcentroid** discussed in Section 6.1.4). Rough shift estimates are optionally input along with the coordinates of objects in the reference image. Coordinates of the objects in the reference image are specified in a text file with one object per line with the x and y coordinates in columns 1 and 2 respectively. The format for the shifts file is one image per line with the (fractional) x and y shifts in columns one and two respectively. The other frames are then searched, the objects found, and the shifts calculated. The output images may be shifted and trimmed automatically by setting the parameters *shiftimages* and *trimimages* to *yes*. These operations are done in place and the output images are not renamed; so the original images must be copied if they are to be saved. Shift estimates for crowded fields should be calculated to within .5 pixels in x and y to avoid miss identifications. It is best to avoid stars with nearby companions. The example below shows the registration of two images to a reference image:

```
cl> type ilist
im1.imh
im2.imh
im3.imh
cl> type coord
15.71 118.11
140.07 126.94
164.18 128.04
148.51 32.61
95.98 36.48
175.90 32.20
cl> type shift
0.0 0.0
-4.0 3.5
-3.3 2.8
cl> imalign @ilist cord im1 shifts=shift
```

The output from this task shows the calculated shifts for each image, identical to the output from the task **imcentroid**. This is not saved in a file so the output must be redirected to a text file.

```
cl> type output


#            image   X-center  err    Y-center  err    num
     obj0001    180.16 (0.02)     17.81 (0.02)    1
             obj0001     25.35 (0.02)     82.19 (0.02)    2
             obj0001     93.09 (0.08)    131.77 (0.10)    3
             obj0001    148.63 (0.00)    139.76 (0.00)    4
             obj0001    148.55 (0.04)    182.64 (0.04)    5
```

```
            obj0002    174.16 (0.02)     12.81 (0.02)    1
            obj0002     19.35 (0.02)     77.19 (0.02)    2
            obj0002     87.09 (0.08)    126.77 (0.10)    3
            obj0002    142.63 (0.00)    134.76 (0.00)    4
            obj0002    142.55 (0.04)    177.64 (0.04)    5

            obj0003    178.67 (0.02)     20.30 (0.02)    1
            obj0003     23.85 (0.02)     84.69 (0.02)    2
            obj0003     91.55 (0.08)    134.30 (0.10)    3
            obj0003    147.12 (0.00)    142.25 (0.00)    4
            obj0003    147.04 (0.04)    185.13 (0.04)    5

#          reference   X-center  err    Y-center  err    num
        obj0001.imh    180.16 (0.02)     17.81 (0.02)    1
        obj0001.imh     25.35 (0.03)     82.19 (0.03)    2
        obj0001.imh     93.09 (0.08)    131.77 (0.10)    3
        obj0001.imh    148.63 (0.01)    139.76 (0.00)    4
        obj0001.imh    148.55 (0.04)    182.64 (0.04)    5

#!             image    X-shift  err     Y-shift  err     N    internal
             obj0001     -0.00 (0.03)     -0.00 (0.03)    5   (0.00,0.00)
             obj0002      6.00 (0.03)      5.00 (0.03)    5   (0.00,0.00)
             obj0003      1.51 (0.03)     -2.50 (0.03)    5   (0.00,0.00)

7 200 6 198            # trim section = [7:200,6:198]
```

The input shift estimates can be very important and must be within the specified *boxsize* value. Increasing the number of iterations may help the convergence in the centering routine. Large errors in the shift will show very high internal errors. The coordinate fits for that image should be checked for high errors and those points thrown out. The default interpolation algorithm is *spline3* and the default boundary extension is *constant*. The options are explained in more detail in Appendix C.

5 Complex Transformations

Complex transformations include magnification/reduction, rotation, and shifting in any combination. A transformation solution may be calculated using **geomap** described in Section 5.5. The task **magnify** performs one operation only, it rescales images. All other tasks described in this section perform several operations like shifting and rotation.

- **magnify** - rescales the pixel size by magnifying or miniaturizing images.
- **rotate** - rotates and/or shifts images.
- **register** - performs a geometric distortion correction using the input coordinate transformation (calculated by the task **geomap**), in any combination of shifting, rotating, or magnifying the images.
- **imlintran** - linearly transforms images using the input parameters to shift, rotate, or magnify in any combination.

All the registering tasks in this section have parameters which define the interpolation type and boundary conditions for manipulating an image. Conservation of flux is also an option. The specific choices for these parameters are explained in more detail in Appendix C.

5.1 Changing the Pixel Scale: magnify

The **magnify** task is found in the IMAGES package. Unlike most of the other tasks which include other types of transformations, this task is used solely for rescaling the pixels in x and y. The *xmag* and *ymag* values may be used directly from the output of the **geomap** task (see Section 5.5). The pixel scale for each axis is independent. A section of the input image may be scaled using the parameters *x1*, *x2*, *y1*, and *y2*. If these values are not specified, the whole image is used. For example, magnifying a 201x201 pixel image by two in both x and y gives an image which is 401x401 pixels. However, using the center 100x100 of the 201x201 image gives a 201x201 pixel output image.

```
cl> magnify obj0003 obj0003m 2 2
obj0003m
Magnify image obj0003 to image obj0003m.
Interpolation is linear.
Boundary extension is constant.
Boundary pixel constant is 0..
Output coordinates in terms of input coordinates:
x1 = 1., x2 = 201., dx = 0.5
y1 = 1., y2 = 201., dy = 0.5
cl> imhea *obj*.imh
obj0003m.imh[401,401][short]:  m51 B 600s
obj0003.imh[201,201][short]:  m51 B 600s
cl> magnify obj0003 obj0003m2 x1=50 x2=150 y1=50 y2=150
X magnification factor (2.):  CR
Y magnification factor (2.):  CR
obj0003m2
Magnify image obj0003 to image obj0003m2.
Interpolation is linear.
Boundary extension is constant.
Boundary pixel constant is 0..
Output coordinates in terms of input coordinates:
x1 = 50., x2 = 150., dx = 0.5
y1 = 50., y2 = 150., dy = 0.5
cl> imhea obj*.imh
obj0003.imh[201,201][short]:  m51 B 600s
obj0003m.imh[401,401][short]:  m51 B 600s
obj0003m2.imh[201,201][short]:  m51 B 600s
```

The task gives detailed information of what it is doing when it is run, and the output may be saved by specifying a *logfile*. The output image size may be different than expected as explained in more detail in the help page for this task. The default interpolation algorithm is *linear* and the default boundary extension is *constant*. More on the options for these parameters is found in Appendix C.

5.2 Rotation of Images: rotate

The task **rotate** is found in the IMAGES package. If shifting and rotation is to be performed on an image, then this is the task to use. The important parameters here are the rotation angle and the x and y shift given in this case by the difference in the origin of the rotation and origin of the output image. It is also a good idea to preserve the dimensions of the input image given by the parameters *ncols* and *nlines*.

If the rotation is a non-negligible value, output from the task **geomap** gives the rotation in x and y with respect to the center of the image. This information could be used for the *rotation* parameter in the task **rotate**. Within **rotate** however, any point may be chosen as the center of the rotation axis; the default is the center of the image.

`cl>` **rotate obj0002 robj0002 rotation=43.2 ncols=800 nlines=800**

In the above example the output 800x800 pixel image will be rotated 43.2° about the center of the image. If an image has a shift and a rotation, then use the center coordinates in the initial origin parameters and compute the shift for the new origin, i.e., add or subtract the x and y shifts from the center value in the image. For example, if the shift is 3.43 in x and -1.45 in y, then run the task using:

`cl>` **rotate obj0002 robj0002 rotation=43.2 xin=400.5 yin=400.5 **
xout=403.93 yout=399.05

This example combines the x and y shift with the rotation using the center of the image as the axis for the rotation. The image size may also be included in this command.

`cl>` **rotate obj0002 robj0002 rotation=43.2 xin=1.0 yin=1.0**

This command rotates the image about the origin specified by *xin* and *yin*. In each of these examples the output image is given a new name, however it is possible to do the transformation in place. The default for the interpolation is *linear* and the boundary is *nearest*. The options are described in detail in Appendix C.

5.3 register

The **register** task is found in the IMAGES package. The input and output files from **geomap** are used to perform the transformation. The transformation function may be set to either *linear* or *geometric*. Again, it is possible to specify the size of the output image using the parameters *xmin*, *xmax*, *ymin*, and *ymax*. The parameters *nxblock* and *nyblock* are best left at their default values for small images. These parameters define the amount of memory that is being used by the task during its execution. For large images you may want to make these parameters as large (less than or equal to the image size) as possible without getting "Out of Memory" errors since the larger the "buffer" size the more quickly the task will run. For large elongated images you may try setting the *nxblock* parameter to the size of the x axis and then make *nyblock* as large as possible. The format for the transformation file is the output file from **geomap** and the coordinate list is identical to the list input to **geomap**.

`cl>` **register obj0002 tobj0002 trans02 coord02**
`cl>` **register @inlist @outlist trans coords**

If a list of images was input to the **geomap** task then the output file contains multiple transformations, and a list of images should be used as input and output to **register** as in the second example above. The two files must have the same number of images, and the *transform* parameter must have an equivalent number of data records.

The task **geotran** (in the IMAGES package as well) is a slightly more complex version of **register**. It allows the user to override the linear transformation in the **geomap** output—it is advised that these options be used with great care. It may be best to use **register** unless you absolutely need the complexities of **geotran**.

5.4 imlintran

The **imlintran** task is found in the IMAGES package. This task does not take the direct output file from **geomap** but will transform a list of images with the same transformation solution (see the next section for more on **geomap**). The task will take any combination of the possible operations. Also, this task only performs the linear parts of the transformation so if high orders were used, they will be ignored. The rotation is given by *xrotation* and *yrotation*, the scaling by *xmag* and *ymag*, and the shift in pixels given by *xout* and *yout*. The transformation is done with respect to the center of the image unless the parameters *xin* and *yin* are specified. The examples use the origin as the axis of rotation.

```
cl> imlintran obj0004 tobj0004 42.5 42.5 .84 .95 xin=0.0 yin=0.0 \
xout=1.51 yout=-2.5
cl> imlintran @ilist @olist 42.5 42.5 .84 .95 xin=0.0 yin=0.0 \
xout=1.51 yout=-2.5
```

If the transformation is representative for a number of images, then use an input list for both the *input* and *output* parameters. The two files must have the same number of images. Before transforming an image, it is possible to check the mapping using the task **lintran** which will take the coordinate position of the control points and perform the transformation to the set of objects. The output values from **lintran** should agree with the reference positions (see Appendix B.3).

5.5 Getting Complex Map Solutions: geomap

The task **geomap**, found in the IMAGES package, was written to compute a spatial transformation of image coordinates to those of a reference image. The output file created by **geomap** is used as input to the task **register** (see Section 5.3). The format of the *input* file to **geomap** should include one object per line with the x and y coordinates from the reference image, followed by the x and y coordinates of an image to be registered, all entries separated by a space. A method for creating an input table for **geomap** is shown in Appendix B.2. The size of the output image, which also defines the region of validity of the fit, should be specified by the parameters *xmin*, *xmax*, *ymin*, and *ymax*, otherwise the task defaults to a box size equivalent to the minimum and maximum values in the x and y reference values found in the input file. If this region of validity for the fit is too small, then the coordinate transformation may not be reliable at the edges of the image. The output file is written containing the transformation parameters which will allow other tasks to register the image to the reference image.

The fitting function must best represent the analytic surface to be fit; the choices are *legendre*, *chebyshev*, and *polynomial*. The task defaults to a linear fit excluding the cross terms. Beware though that using too high an order can produce errors (see Section 2.3). The calculation may be done using real or double precision numbers. Using an 800x800 pixel image named *obj0002* with an input list of coordinates called *coord02* and an output file called *tran02*, run the task:

```
cl> geomap coord02 tran02 1 800 1 800
cl> geomap @clist @tlist 1 800 1 800
```

The second example shows the input in list formats. A plot will appear in interactive mode with the coordinates of the image marked over a grid, showing the transformation for the image. If the residuals are high, i.e., much greater than 0.05, then increasing the orders or including the cross terms may improve the fit. The output file will contain the parameters for the shift, the rotation, and the scaling necessary to register "obj0002" to the reference image. The important commands for interactive mode are:

- **d** - delete point nearest the cursor.
- **f** - fit the data and redraw graph.
- **l** - print fit parameters.
- **u** - undelete point nearest the cursor.
- **?** - print the help menu.
- **:xxorder** [**value**] - change the xorder.
- **:xyorder** [**value**] - change the cross term order.
- **:yyorder** [**value**] - change the yorder.
- **:yxorder** [**value**] - change the cross term order.
- **:xxterms** [**y/n**] - include/exclude the cross terms.
- **:yxterms** [**y/n**] - include/exclude the cross terms.
- **q** - quit the task.

See the help page for more information about these and other interactive commands.

6 Finding Centers and Computing Shifts

6.1 Getting Object Coordinates

The tasks **shiftlines**, **imshift**, and **imalign** require input coordinates or initial guesses for the shifts to calculate and perform the final transformations. The coordinates of objects in an image, used to compute these transformations, are calculated using any one of the following tasks:

- **rimcursor** - reads the cursor position (center not calculated) from the display device.
- **imcntr** - independent axis centroid algorithm for one set of coordinates at a time.
- **imexamine** - computes moments of the marginal distributions using the display device.
- **imcentroid** - independent axis centroid algorithm.
- **center** - several complex algorithms available.
- **fitpsf** - several complex algorithms available.

The coordinates for a good selection of objects with good spatial coverage in all the frames should be obtained, otherwise the image mapping for the edges will be extrapolated resulting in bad transformations at the edges of the images. These tasks may have different algorithms available for centering (see Appendix C for a discussion of these algorithms). The first three tasks, **rimcursor**, **imcntr**, and **imexamine**, are simple-to-use tasks and usually satisfy most needs. The output file must however be revised and the shifts between images must be calculated using other methods. If only x and y shifts are required, use the task **imcentroid** which performs this calculation. More complex transformations must be reformatted for input to other tasks such as **geomap**.

6.1.1 rimcursor

The **rimcursor** task is found in the LISTS package. Unlike the sophisticated centering algorithms used by the other tasks, **rimcursor** simply reads the position of the image cursor, without calculating the center. The position is good to only .5 pixels in x and y. This is a good method for getting quick positions for x and y shifting. The positions are output to the terminal but they can also be redirected to a file. An output file will look like the following:

```
cl> type rimcoord
379.5 66.5 101 \040
347.5 188.5 101 \040
224.5 130.5 101 \040
385.5 254.5 101 \040
405.  273.75 101 \040
217.75 439.  101 \040
```

This file may have to be edited for use as input to other tasks. The output formats for the coordinates may also be changed by setting *wxformat* and *wyformat*. See the help page for this task for more information about these parameters. To run **rimcursor** the image must first be displayed in the *display* window:

```
cl> display obj0003 1
cl> rimcursor obj0003
or
cl> rimcursor obj0003 > coords1
```

The cursor will move to the *display* window ready for interactive commands when **rimcursor** is executed. Move the cursor to the position of an object and hit any key. To exit this task once coordinates have been obtained for all the objects, type ˆ **d** or ˆ **z** where the ˆ character stands for the "control" key—press both at the same time. Repeat this procedure for all the other images and be sure to keep track of the order of the objects and their positions (e.g., marking them in the same order each time). The default coordinate system is *logical* or pixel units, however this may be changed to obtain coordinates in another WCS (world coordinate system) by setting the parameter called *wcs* to some other coordinate system value.

6.1.2 imcntr

The task **imcntr** is found in the PROTO package. Centering is performed by summing up the lines and columns in the specified box around the objects (parameter *cboxsize*) and finding the peak. If the resultant values deviate by more than 1 pixel from the initial input coordinates (parameters *x_init* and *y_init*), then a second iteration is performed. The x and y coordinates are computed independently. The task requires approximate positions as input which can be obtained using the the task **rimcursor** mentioned above or using the the cursor in the display window. The task only computes one object center at a time.

Once the first-guess positions are obtained, then run the task for each object, inputing one initial x and y position at a time. Sample parameters are shown below along with example runs of the task.

```
cl> lpar imcntr
       input =                Image names
      x_init =                Approx x position of star
      y_init =                Approx y position of star
   (cboxsize = 5)             Size of extraction box
       (mode = "ql")
```

```
cl> imcntr obj0003 177.5 20.0
[obj0003] x:  180.12 y:  17.81
cl> imcntr obj0003 23.0 84.5
[obj0003] x:  25.30 y:  82.17
```

This task does not write to a logfile so the user must write down the results or redirect the output to a file. Redirection is done in the following way:

```
cl> imcntr obj0003 177.5 20.0 > poslist
cl> imcntr obj0003 23.0 84.5 >> poslist
cl> type poslist
[obj0003] x:  180.12 y:  17.81
[obj0003] x:  25.30 y:  82.17
```

This file will have to be edited accordingly for input to other tasks which calculate or perform the transformation operation.

6.1.3 imexamine

The **imexamine** task is found in the IMAGES.TV package and can be used in interactive mode with the image display. The operation of finding the object centers is governed by two parameter sets for this task, the **imexamine** parameters themselves and the *rimexam* pset[1] parameters. The standard **imexamine** parameters set up the environment for executing the task, while the *rimexam* parameters specify the options for the use of the *r* and *a* commands used to measure the centers. Output by default goes to the terminal screen so the *keeplog* parameter in *imexamine* must be set to *yes* and the *logfile* name specified if you want to save the output in a file. This logging facility can be turned on and off interactively.

The centering parameter is found in the *rimexam* parameters, and must be set to **yes**. The cursor position is used as the initial point for computing the center moments of the marginal distributions in x and y. The marginal distributions are obtained from a square aperture with edge dimensions of twice the aperture radius parameter. Only pixel values above the mean are used in the computation and another iteration is done if the central moments are in a different pixel than that used for extracting the marginal distributions.

[1] Psets are parameter sets within parameter sets. Psets provide a way to group a long list of parameters into smaller, more manageable parameter lists that can then be shared by several tasks in a package.

The task may be run interactively on all your images, and the coordinates may be appended to the specified *logfile*. First edit the parameters for this task and then run it for each image to be registered. **Imexamine** will also take an input list of images or an "@file".

```
cl> lpar imexamine
       input = "dev$pix"       images to be examined
       frame = 1               display frame
       image = "m51"           image name
    (logfile = "junk2")        logfile
    (keeplog = yes)            log output results
     (defkey = "a")            default key for cursor list input
 (autoredraw = yes)            automatically redraw graph
  (allframes = yes)            use all frames for displaying new images
    (nframes = 0)              number of display frames (0 to autosense)
     (ncstat = 5)              number of columns for statistics
     (nlstat = 5)              number of lines for statistics
   (graphcur = "")             graphics cursor input
   (imagecur = "")             image display cursor input
        (wcs = "logical")      Coordinate system
    (xformat = "")             X axis coordinate format
    (yformat = "")             Y axis coordinate format
   (graphics = "stdgraph")     graphics device
    (display = "display(image='$1',frame=$2)") display command template
(use_display = yes)            enable direct display interaction
       (mode = "ql")

cl> lpar rimexam
     (banner = yes)            Standard banner
      (title = "")             Title
     (xlabel = "Radius")       X-axis label
     (ylabel = "Pixel Value")  Y-axis label
    (fitplot = yes)            Overplot gaussian fit?
     (center = yes)            Center object in aperture?
 (background = yes)            Fit and subtract background?
     (radius = 5.)             Object radius
     (buffer = 1.)             Background buffer width
      (width = 2.)             Background width
     (xorder = 1)              Background x order
     (yorder = 1)              Background y order
    (magzero = 23.)            Magnitude zero point
      (rplot = 8.)             Plotting radius
         (x1 = INDEF)          X-axis window limit
         (x2 = INDEF)          X-axis window limit
         (y1 = INDEF)          Y-axis window limit
         (y2 = INDEF)          Y-axis window limit
  (pointmode = yes)            plot points instead of lines?
     (marker = "plus")         point marker character?
   (szmarker = 1.)             marker size
       (logx = no)             log scale x-axis
       (logy = no)             log scale y-axis
        (box = yes)            draw box around periphery of window
 (ticklabels = yes)            label tick marks
      (majrx = 5)              number of major divisions along x grid
      (minrx = 5)              number of minor divisions along x grid
      (majry = 5)              number of major divisions along y grid
      (minry = 5)              number of minor divisions along y grid
```

```
(round = no)             round axes to nice values?
 (mode = "ql")
```

The cursor commands which are most important are:

- **a** - centers and performs circular aperture photometry for the star.
- **r** - the **a** key plus plotting the radial profile of the star.
- **w** - toggle writing to the output file.
- **?** - print help menu.
- **q** - quit the task.

Subsequent executions of this task will reopen the *logfile* and append to the named file. The output file includes more than just the coordinates:

```
# [1] test - m51  B  600s
#   COL    LINE   RMAG     FLUX      SKY   N  RMOM ELLIP    PA     PEAK  FWHM
 379.10   66.84  18.71  32739.0    78.00  78  3.11 0.018 -57.3  3997.23  2.52
 347.61  188.79  16.89 176180.0   144.00  80  3.09 0.019 -45.6 21803.54  2.52
 224.30  131.13  19.05  23914.0   106.00  78  3.08 0.047 -15.2  3072.85  2.43
 404.53  274.25  18.69  33450.0   110.00  80  3.12 0.019  87.8  3956.36  2.62
 217.16  439.40  20.79   4846.0   138.00  79  2.74 0.131 -31.2   463.33  3.08
```

It is up to the user to keep track of the order in which the objects are marked in the images. Each time **r** is used, a title line is output to the *logfile*, requiring that these lines be edited out. The **a** option prints out a title line only once at the beginning of the logfile as in the example above.

6.1.4 imcentroid

The **imcentroid** task is found in the PROTO package in IRAF. The task measures the x and y coordinates of a list of objects in a list of images, and then finds the mean x and y shifts between the images and a specified reference image. The centering algorithm is the same one used by **imcntr**, where the lines and columns are summed up in the region defined by the *boxsize* parameter, and the x and y centers are calculated independently. The task is sensitive to nonstellar object profiles, sources with a low signal to noise ratio, and complicated shapes that may appear in the background.

Approximate positions of the objects in the reference image are required as input to the task and may be obtained using the **rimcursor** task mentioned above. The initial x and y positions should be written to a file, one object per line with the x and y coordinates in columns one and two, respectively. This file should be named in the *coords* parameter. The object coordinates for the reference image are then used to find the object coordinates in the the other images during two passes using the values of *bigbox* and *boxsize* to constrain the centering algorithm. An optional *shifts* file can be input containing coarse shifts for each image with respect to the reference image to assist with the initial pass; otherwise **imcentroid** computes the initial shifts using *bigbox* to constrain the centering. The output from this task goes to the text window so you must redirect the output to a file if you wish it to be saved. The only other parameters that may be important are the number of iterations for the centering algorithm, *niterate*, and the *tolerance* for the convergence of the centering algorithm.

The parameters for the task are listed below, along with an example of its execution.

```
cl> lpar imcentroid
      images =                  List of images
      coords =                  Coordinate file
  (reference = "")              Reference image
     (shifts = "")              Initial shift file
    (boxsize = 7)               Size of the small centering box
     (bigbox = 11)              Size of the big centering box
   (negative = no)              Are the features negative?
 (background = INDEF)           Reference background level
      (lower = INDEF)           Lower threshold for data
      (upper = INDEF)           Upper threshold for data
   (niterate = 3)               Maximum number of iterations
  (tolerance = 0)               Tolerance for convergence
    (verbose = yes)             Print the centroids for every source?
       (mode = "ql")
```

cl> **imcentroid > output**
List of images (obj0001,obj0002,obj0003): **CR**
Coordinate file (coords): **CR**
cl> **type output**

```
#            image   X-center   err    Y-center   err    num
           obj0001    180.16 (0.02)     17.81 (0.02)     1
           obj0001     25.35 (0.02)     82.19 (0.02)     2
           obj0001     93.09 (0.08)    131.77 (0.10)     3
           obj0001    148.63 (0.00)    139.76 (0.00)     4
           obj0001    148.55 (0.04)    182.64 (0.04)     5

           obj0002    174.16 (0.02)     12.81 (0.02)     1
           obj0002     19.35 (0.02)     77.19 (0.02)     2
           obj0002     87.09 (0.08)    126.77 (0.10)     3
           obj0002    142.63 (0.00)    134.76 (0.00)     4
           obj0002    142.55 (0.04)    177.64 (0.04)     5

           obj0003    178.16 (0.02)     19.81 (0.02)     1
           obj0003     23.35 (0.02)     84.19 (0.02)     2
           obj0003     91.09 (0.08)    133.77 (0.10)     3
           obj0003    146.63 (0.00)    141.76 (0.00)     4
           obj0003    146.55 (0.04)    184.64 (0.04)     5

#        reference   X-center   err    Y-center   err    num
       obj0001.imh    180.16 (0.02)     17.81 (0.02)     1
       obj0001.imh     25.35 (0.02)     82.19 (0.02)     2
       obj0001.imh     93.09 (0.08)    131.77 (0.10)     3
       obj0001.imh    148.63 (0.00)    139.76 (0.00)     4
       obj0001.imh    148.55 (0.04)    182.64 (0.04)     5

#!           image    X-shift   err     Y-shift   err     N    internal
               im1     -0.00 (0.03)     -0.00 (0.03)     5   (0.00,0.00)
               im2      6.00 (0.03)      5.00 (0.03)     5   (0.00,0.00)
               im3      2.00 (0.03)     -2.00 (0.03)     5   (0.00,0.00)
```

```
7 200 6 198            # trim section = [7:200,6:198]
```

The calculated x and y shifts are found in the last section of the output file. Since this is done by the task, go on to Section 4.4 to perform the shifting operation on the list of objects.

6.1.5 center

The task **center** is found in the NOAO.DIGIPHOT.APPHOT package. **Center** computes accurate centers for a set of objects in an image whose rough coordinates are read interactively from the image cursor in the *display* window or from a text file. The centering algorithms are explained in more detail in Appendix C. The x and y positions are saved in a text file named by the *output* parameter. If this is left as *default*, the name will be the image name appended with a *ctr.1*. The number is automatically incremented if the task is run many times on the same image. The output format for this file is like the output from the photometry tasks with a large header with the parameters used in running the task and then the tabulated information as shown below:

```
#K IRAF       = NOAO/IRAFV2.10EXPORT    version    %-23s
#K USER       = lwells                  name       %-23s
#K HOST       = ursa                    computer   %-23s
#K DATE       = 01-19-94                mm-dd-yr   %-23s
#K TIME       = 11:55:54                hh:mm:ss   %-23s
#K PACKAGE    = apphot                  name       %-23s
#K TASK       = center                  name       %-23s
#
#K SCALE      = 1.                      units      %-23.7g
#K FWHMPSF    = 1.                      scaleunit  %-23.7g
#K EMISSION   = yes                     switch     %-23b
#K DATAMIN    = 0.                      counts     %-23.7g
#K DATAMAX    = 28000.                  counts     %-23.7g
#K EXPOSURE   = exptime                 keyword    %-23s
#K AIRMASS    = airmass                 keyword    %-23s
#K FILTER     = filters                 keyword    %-23s
#K OBSTIME    = ut                      keyword    %-23s
#
#K NOISE      = poisson                 model      %-23s
#K SIGMA      = INDEF                   counts     %-23.7g
#K GAIN       = gain                    keyword    %-23s
#K EPADU      = 7.                      e-/adu     %-23.7g
#K CCDREAD    = rdnoise                 keyword    %-23s
#K READNOISE  = 5.2                     e-         %-23.7g
#
#K CALGORITHM = centroid                algorithm  %-23s
#K CBOXWIDTH  = 6.                      scaleunit  %-23.7g
#K CTHRESHOLD = 8.                      sigma      %-23.7g
#K MINSNRATIO = 1.                      number     %-23.7g
#K CMAXITER   = 10                      number     %-23d
#K MAXSHIFT   = 6.                      scaleunit  %-23.7g
#K CLEAN      = no                      scaleunit  %-23b
#K RCLEAN     = 1.                      scaleunit  %-23.7g
#K RCLIP      = 2.                      scaleunit  %-23.7g
#K KCLEAN     = 3.                      sigma      %-23.7g
#
#N IMAGE               XINIT     YINIT     ID   COORDS                 LID     \
#U imagename           pixels    pixels    ##   filename               ##      \
#F %-23s               %-10.3f   %-10.3f   %-5d %-23s                  %-5d
#
```

```
#N XCENTER    YCENTER    XSHIFT  YSHIFT  XERR    YERR    CIER CERROR          \
#U pixels     pixels     pixels  pixels  pixels  pixels  ##   cerrors         \
#F %-14.3f    %-11.3f    %-8.3f  %-8.3f  %-8.3f  %-8.3f  %-5d %-13s
#
dev$pix                379.000   67.000    1    nullfile             0        \
   379.120    66.813     0.120   -0.187  0.016   0.016   0    No_error
dev$pix                349.000   189.000   2    nullfile             0        \
   347.588    188.757    -1.412  -0.243  0.006   0.006   0    No_error
dev$pix                224.000   131.000   3    nullfile             0        \
   224.298    131.166    0.298   0.166   0.017   0.018   0    No_error
dev$pix                405.000   274.000   4    nullfile             0        \
   404.496    274.294    -0.504  0.294   0.013   0.015   0    No_error
dev$pix                218.000   439.000   5    nullfile             0        \
   217.188    439.328    -0.812  0.328   0.044   0.043   0    No_error
```

The required values *xcenter* and *ycenter* may be extracted from this large table using the task **txdump** (see Section 6.2). In interactive mode, the user can examine, adjust and save the algorithm parameters.

If the task is to be used interactively, the image must be displayed in the *display* window before running the task:

ap> **display obj0003 1**
ap> **center obj0003**

The following commands may be used to get center positions for the stars interactively:

- **d** - plot the radial profile for the star.
- **i** - setup center parameters interactively using the cursor.
- **v** - verify the center parameters.
- *space bar* - fit the center and output the result.
- **?** - print help menu.
- **:show** - list the parameters.
- **q** - quit the task.

The help menu is exited using **q**, returning to interactive cursor mode. The **space bar** is the main command to obtain object coordinates in interactive mode. The shifts between the frames must still be calculated (see Section 6.2).

6.1.6 fitpsf

The **fitpsf** task is found in the NOAO.DIGIPHOT.APPHOT package. **Fitpsf** fits the center of a stellar profile using the specified algorithm. The centering algorithms are explained in more detail in Appendix C. Rough coordinates are read interactively from the image cursor in the *display* window or from a text file. The x and y positions are saved in a text file named by the *output* parameter. If this is left as *default*, the name will be the image name appended with a *psf.1*. The number is automatically incremented if the task is run many times on the same image. The output looks the same as for the **center** task explained in the previous section. Pulling the *xcenter* and *ycenter* out of the *center* output files can be done using the task **txdump** (see Section 6.2). In interactive mode, the user can examine, adjust and save the algorithm parameters.

Display the image in the *display* window first, and then run the task:

ap> **display obj0003 1**
ap> **fitpsf obj0003**

A selected list of interactive commands are given below:

- **d** - plot the radial profile for the star.
- **i** - setup center parameters interactively using cursor.
- **v** - verify the center parameters.
- *space bar* - fit the center and output the result.
- **?** - print help menu.
- **:show** - list the parameters.
- **q** - quit the task.

Quit the help menu with **q** to get back the interactive cursor. The objects may be marked using the **space bar**. Exit the task by typing **q** twice in a row. The shifts between the frames must still be calculated (see the next section).

6.2 Computing the Shifts

Now that you have all of these coordinate files from the various tasks discussed above how do you actually compute the shifts needed by some of the registration tasks? There is really no one task in IRAF that will compute the shifts once you have measured the centers for several objects in your frames. This section covers some of the options.

If you have only a few shifts to compute then a calculator may be the easiest and quickest approach, though this may be tedious. As shown in Section 6.1.4 the **imcentroid** task computes the shifts directly. This is the easiest and best way to obtain the required values.

The files generated by the centering tasks **center** and **fitpsf** are rather complex files, and the coordinates of the objects (*xcenter* and *ycenter*) need to be extracted from these files using the task **txdump** found in the NOAO.DIGIPHOT.APPHOT package. The output of **txdump** is printed to the terminal by default so you must redirect it to a file. The resultant file will have only the x and y center positions. For example:

ap> **txdump obj0002.ctr.1 xcenter,ycenter yes > coord02**

The file *coord02* now has the x and y positions of all the marked objects. Calculations must be done to compute the shifts between the reference and the images to be registered to it. There is a task called **filecalc** in the CTIO package (not distributed with the main IRAF system) that will allow the user to do computations on files or columns of data (see Appendix B) so you may want to investigate this option. It also has information and examples for creating tables in the correct format for such tasks as **geomap**. See Appendix B for an example of using the STSDAS.TABLES package for a way to compute shifts on large quantities of data.

7 Combining Registered Images

There are many tasks in IRAF which may be used to combine images once they have been registered. This section gives a brief description without examples for the possible options. It is left to the user to choose among and experiment with these tasks.

- **imarith** - (IMAGES package) Two or more images may be used in this task with the available operations, +, -, *, and /. The resultant image name must be specified. Input file lists may be used as long as the number of images in each file are the same for one of the operands and the output image. This task is very useful for differencing broad and narrow band images.
- **imsum** - (IMAGES package) Many images may be summed or averaged using the sum, average or median *option*. The calculation may be done in integer or real values, but beware - using the **sum** option with integer values may cause overflow error. Rejection of low or high values may be set with the *low_reject* and *high_reject* parameters. This task also updates the header parameters specified in *hparams*.
- **combine** - (IMRED.CCDRED package) Many images may be averaged using the **average** or **median** options in the *combine* parameter. This task has many other features which allow thresholding, masking and other rejection algorithms. Weighting and offsetting may also be performed to images with different exposure times and backgrounds. Many of the options are explained in more detail in the manual for fixing bad pixels and cosmic rays (see Appendix D).
- **imcombine** - (IMAGES package) This task is identical in operational aspects to **combine** described above.
- **scombine** - (IMRED.KPNOSLIT and other spectral packages) Any number of spectra may be summed or averaged using the *sum*, *average*, or *median* options. The spectra are combined using a common dispersion sampling, interpolating where necessary, so the spectra must be wavelength calibrated. The starting and ending wavelength for the output spectrum may be specified as well as the wavelength increment and length in pixels, otherwise the first spectrum's dispersion solution is used (if *first* is set to **yes**). This task has rejection options for removing deviant pixel values due to bad regions or cosmic rays (see Appendix D for the manual on fixing bad pixels and cosmic rays).

All of these tasks have help pages which define the parameters and give examples of its use.

A Using the Imtool Display Window for Image Registration

Quick comparisons of images may be done using the *imtool* display window. Two cases are discussed below. In the first scenario the images are not necessarily aligned, but the user can force coarse alignment using *imtool* commands. In the second scenario the images are roughly aligned but they have been zoomed and panned and the user wishes to re-align them quickly again (in pixel space).

Begin by displaying the images in the *imtool* window using the task **display**:

```
cl> display obj0003 1
z1 = 35., z2 = 321.6826
cl> display obj0004 2
z1 = 54., z2 = 391.2728
cl> display obj0005 3
z1 = 42., z2 = 416.7854
```

For the first scenario:

- Move the cursor from the *gterm* to the *imtool* window.
- Press and hold the **right** mouse button while moving the cursor to set the transfer function for the frame displayed. Release the button when satisfied.
- Type ˆ **f** to go to the next frame, and without moving the cursor press the **right** mouse button; repeat for the next frame. This sets each frame to a common display background (transfer function).
- Zoom the desired region of the first image by moving the cursor to the area of interest and pressing the **middle** button on the mouse twice in a row.
- Again use the ˆ **f** to page through the frames, clicking twice on the **middle** mouse button for each frame for zooming and then centering the object of interest by placing the cursor on the object and single clicking the **middle** mouse button.
- Now use one of the options below to blink between frames.

The **Blink on** option in the frame menu may be used; this is a toggle switch. The *imtool* display has other commands which will allow blinking between frames:

- ˆ **f** - blinks frames forward in order.
- ˆ **r** - moves back in frame order.
- ˆ **b** - cycles through the frames set up by the *imtool* setup parameter *Frames to be blinked.*

For our second scenario, suppose we have zoomed and panned around several images displayed in the *imtool* window. Now you would like to line them all up for subsequent blinking to match the current displayed image. The *register* option in the *imtool* frame menu simply adjusts the pan offset of all the frames to match that of the image currently being displayed. In other words, it merely matches the pixel coordinates between the images.

B Other Useful Tasks

There are several useful tasks in IRAF which may be used to do table manipulation and column arithmetic. Another task may be used to check transformation values. Described below are:

- **ttools** - manipulates files in the STSDAS tables format.
- **filecalc** - performs column arithmetic on one or multiple files.
- **lintran** - checks transformation values derived for the registering of images using coordinate lists for the objects.

B.1 The TTOOLS Package

The TTOOLS package is found in the TABLES package that is developed and distributed by STScI. This package is add-on software to the basic IRAF package and is available by anonymous ftp to stsci.edu (130.167.1.2).

Most of the IRAF tasks generate simple ASCII or text files as output or by redirection. The TABLES package supports a different format—a binary format—that will be referred to as tables for this discussion. The TTOOLS package contains a set of table manipulating tasks. To use these tasks however, the ASCII text files must be converted to the tables format. We will discuss one way to do this conversion.

Let's assume that we have two files, *coord1* and *coord2*, that contain the x and y coordinates of objects in the reference image and another image that we want to shift to the same coordinate system as the reference image. We assume that the objects are ordered the same in each file. The conversion process would be simplified if the two files were joined into one file. The task **join** in the PROTO package can be used for this purpose. For example,

```
cl> join coord1 coord2 output=coordj
cl> type coordj
379.120 66.813 177.5 20.0
347.588 188.757 23.0 84.5
224.298 131.166 90.5 134.0
404.496 274.294 146.0 142.0
217.188 439.328 146.4 186.5
```

The next step is to generate a text file that defines the column descriptions and data for the tables file. Let us generate a file called *data.cd* that looks like the following (use your favorite editor to do this).

```
cl> type data.cd
xref r f14.5 pixels
yref r f14.5 pixels
xin r f14.5 pixels
yin r f14.5 pixels
```

Now we are ready to create a tables file.

```
cl> tcreate coord.tab data.cd coordj
```

The two major tasks to use for table manipulations are **tprint** which types the table to the terminal, and **tcalc** which performs table arithmetic. All the relevant values to be manipulated must be contained in one table.

Now we are ready to compute the shifts between these two images.

tt> **tcalc coord.tab xshift "xref-xin"**
tt> **tcalc coord.tab yshift "yref-yin"**

The above two executions of **tcalc** have calculated the x and y shifts and added two columns to the input table called *xshift* and *yshift*. The average of these two shift values can now be used as input to the **imshift** task.

A TTOOL table file can also be converted back to an ASCII file. To do this, use the task **tdump** which writes the contents of the tables file to an ASCII file either by specifying the output file or piping the output to a filename. The column labels are included in the ASCII file. See the on-line help pages for these tasks for more details.

B.2 The Task filecalc

The **filecalc** task operates on ASCII text files. The task can be found in the CTIO package, another add-on package to IRAF. Since the CTIO package is not part of the general IRAF distribution, it can be acquired by anonymous ftp to iraf.noao.edu (140.252.1.1), in a file called ctio.tar.Z in the iraf.old directory. The installation instructions are found in the readme.ctio file.

Using the output files from either **fitpsf** or **center** as examples, the x and y shifts of an image can be calculated using the following procedure. First pull out the *xcenter* and *ycenter* values from the output files of these tasks using the task **txdump** as shown in Section 6.2. These output coordinate files are then input to the task **filecalc**.

To calculate the shifts for a pair of images, we assume obj0001 was chosen as the reference image, obj002 is the image needing shifting, and the files *coord01* and *coord02* are the coordinate files created using **txdump**. The shifts are calculated and output to a text file using:

ct> **filecalc coord01,coord02 "$1@1-$2@1;$1@2-$2@2" **
format="%5.2f\t%5.2f" > 2shift1

The file called *2shift1* will have the form x shift and y shift, separated by a tab character. Replacing the *t* with a single space will separate the output values by a space. The formatting is optional. More information is given in the help page for this task. The format of the output file should be checked using **type 2shift1** or **page 2shift1**. For this task to work properly, the objects must have been marked in the same order in each image such that each line is a corresponding object in the images. The file now has a list of the shifts at each object position and these values must be averaged. This is done on a list of numbers only so first the columns must be separated by simply grabbing the values from the output file using **filecalc** and inputting these values to the task **average**. Do this by typing:

ct> **filecalc 2shift1 "$1@1" format="%5.2f" | average >> shift2**
ct> **filecalc 2shift1 "$2@1" format="%5.2f" | average >> shift2**

The average of the first column (x coordinate) will be output to the file *shift2*. Appending the y coordinate will create another line with the y shift. This file must be edited to combine these two values on one line if it is to be used as an input file for the task **imshift**, for example.

The input file for **geomap** may be put together using **filecalc** as well. The following example simply puts the two files together by pulling out the reference object coordinates and the coordinates of the image to be registered. The results are saved by piping output to a file:

ct> **filecalc coord01,coord02 "$1@1;$2@1;$1@2;$2@2" **
format="%6.2f %6.2f %6.2f %6.2f" > map1.2

Check this file to be sure it looks right. The format used puts spaces in between the values for the object coordinates. For this task to work properly, the objects must have been marked in the same order in each image such that each line is a corresponding object in the images.

B.3 Checking the Transformation Parameters Using lintran

The **lintran** task tests the transformation using only the object coordinates of the image to be registered to the reference image; it does not perform the transformation on an image. The output from the task **geomap** contains the necessary values to input to **lintran** in the *x2*, *y2*, *xscale*, *yscale*, and *angle* parameters. Given the coordinates of the objects in an image and its corresponding transformation parameters, **lintran** outputs the transformed coordinates which should agree with the object coordinates of the reference image. The input is given as a parameter or is piped to the task:

> `cl>` **lintran coords02 x1=100.0 y1=100.0 x2=103.43 y2=98.55**
> `cl>` **lintran coords02 x1=0.0 x2=0.0 x2=3.43 y2=-1.45 > shifted**

The first example shows the simple x and y shifts using some arbitrary position in the image. The second example shows how the shifts are applied to the coordinates in the file given by the values of *x2* and *y2*. The results are output to the terminal, so if an output file is required, another pipe must be used to create it.

If the rotation is a non-negligible value, then caution must be taken since the output from the task **geomap** gives the rotation in x and y with respect to the center of the image while **lintran** uses the origin and a simple angle. For a case where rotation alone is important, then the task is run the the same way using the following,

> `cl>` **lintran coord02 x1=400.5 y1=400.5 x2=400.5 y2=400.5 angle=43.2**

Assuming the center of an 800x800 pixel image is 400.5x400.5, the output values will be rotated 43.2° about the center of the image. If an image has a shift and a rotation, then use the center coordinates in the initial origin parameters, and compute the shift for the new origin, i.e., add or subtract the x and y shifts from the center value in the image.

> `cl>` **lintran coord02 x1=400.5 y1=400.5 x2=403.93 y2=399.05 angle=43.2**

This has combined the x and y shifts above with the rotation, using the center as the axis for the rotation. The output values *xmag* and *ymag* from the task **geomap** may also be input in the **lintran** parameters *xscale* and *yscale*, respectively.

B.4 Positions from Spectral Images

The task **implot**, found in the PLOT package, can be used for obtaining positions in a 2D spectrum. The spectra must be corrected for any distortions that may exist. For highly undersampled spectra, i.e., infrared spectra, shifting by integer pixels is suggested to avoid any interpolation or rebinning of the data.

> `cl>` **implot obj0004**

In interactive mode, the **p** command is used to find the center of a profile by marking the left and right edges of the feature to be measured. This task does not save the calculated positions in a file so write them down.

The task **splot** is found in all the spectral reduction packages, like NOAO.IMRED KPNOSLIT and NOAO.IMRED.SPECRED. This may also be used to plot 2D spectra with the help of the colon commands. To change to a cut across the stellar position on the slit, the **:dispaxis #** command must be used to set the other axis. The **:imsum #** command sums the number of lines or columns specified. Interactively, this task gives the center of a profile using the **e** or **k** commands. The specified *line* along the dispersion direction is plotted by default.

`kp>` **splot obj0006 36**

This example plots line number 36 along the dispersion direction of the image obj0006. Colon commands used will appear at the bottom of the plotting window.

One dimensional spectra may be handled in the same way using the task **splot** in the NOAO.ONEDSPEC package.

`on>` **splot obj0006.0001**

In interactive mode, the **e** and **k** commands are used to find the center of a profile by marking the left and right edges of the feature to be measured. The positions marked are saved in the file defined by the parameter *save_file*. For both tasks, *q* quits the interactive mode.

Shifts between 2D spectra may be calculated using the xregister task discussed in Section 4.1. Tests should be performed to check the statistics if this is to be used with infrared spectra. For 1D and multispec format spectra which have been dispersion corrected, an Angstrom shift may be applied to one or more spectra using the task **specshift**. The specified shift is added to the existing world coordinates. For linear coordinate systems this has the effect of modifying CRVAL1, for linear multispec coordinate systems this modifies the dispersion coordinate of the first physical pixel, and for nonlinear multispec coordinate systems this modifies the shift coefficient(s). True resampling of spectra is done using **dispcor** if this is more desirable. Note however that tasks such as **scombine** do not require that the spectra cover the same spectral range. See the help pages for these tasks for more information about them.

C Some Important Parameters

C.1 The Centering Algorithms

The tasks used in the APPHOT package have various centering algorithms available. The task **center** uses the pset parameters found in **centerpars** and **datapars**. It is possible that the centering option is turned off, i.e., *none*, so this must be checked. The task **fitpsf** uses the **datapars** psets only, and has its own selection of centering functions. The following sections are taken from the on-line help pages for the specific tasks. The algorithms are explained in more detail in the document, "*Specifications for the Aperture Photometry Package*", by Lindsey Davis (see Appendix D).

The following algorithms are options in the task **center**. *Centroid* is the default algorithm for this task.

- **none** assumes the initial positions are the true centers, no centering is performed.
- **centroid** computes the marginal distributions in x and y for the region defined by the boxsize (*cbox*). The intensity weighted mean and mean error of the x and y marginal distributions are computed using only points in the marginal array above the minimum data pixel plus a threshold value which is either the mean or defined by *cthreshold* (if *cthreshold* is greater than zero). Note that this is the only centering algorithm for which the choice of the parameter *datapars.fwhmpsf* is not crucial.
- **gauss** computes the new centers by fitting a 1D Gaussian function to the marginal distributions in x and y using a fixed *datapars.fwhmpsf* parameter. Initial guesses for the fit parameters are made from the data values. The routine then iterates until a best fit solution is achieved.
- **ofilter** employs a variation of the optimal filtering technique in which the profile is simulated by a triangle function of width *datapars.fwhmpsf*.

The following algorithms are options in the task **fitpsf**. The default function for this task is *radgauss*.

- **radgauss** fits a 2D radial Gaussian. The fit is sensitive to the parameters *fwhmpsf*, *sigma*, and *threshold*.
- **elgauss** is a 2D elliptical Gaussian function.
- **moments** computes the 0th, 1st, and 2nd order moments to estimate the x and y center values. The parameter *threshold* determines the intensity threshold above which the moment analysis is performed.

The following options are available in the **xregister** task. The parameters *xcbox* and *ycbox* define the width of the centering box which are used to compute the fractional pixel x and y center. The correlation image is searched for objects using the following centering algorithms:

- **none** takes the maximum pixel in x and y as the center.
- **centroid** obtains positions from the intensity weighted mean and mean error of the x and y marginal distributions.
- **sawtooth** convolves slices in x and y with a sawtooth and uses the point at which the peak is bisected to determine the center.

- **parabolic** fits a 1D parabola to slices in x and y, then the fitted coefficients are used to compute the peak.
- **mark** uses the interactive mode such that the user may mark the peak with the graphics cursor.

C.2 Boundary Type Options

The *boundary* type parameter in all the transformation tasks have the same options. This parameter defines how pixel values moving in or out of the edge of an image are handled. The possibilities are:

- **nearest** takes the nearest value in the row or column and propagates it.
- **constant** specifies that the value of the *constant* parameter in the task should be used.
- **reflect** uses reflection of the pixels values along the line or column.
- **wrap** will take pixel values exiting the edge of the image and wrap them around to the other side of the image.

C.3 Interpolation Types

If fractional pixels are used, then the interpolation type may become important since this is used to rebin the image. For undersampled data, generally use a linear interpolation. But be aware that any interpolation used on undersampled data may degrade the image more. The possible interpolation options are:

- **nearest** does not interpolate but uses the value of the nearest pixels.
- **linear** is a bilinear interpolator in x and y.
- **poly3** interpolates using a third order polynomial.
- **poly5** interpolates using a fifth order polynomial.
- **spline3** interpolates using a bicubic spline.

For well sampled data, the polynomial interpolators may be best.

Aperture photometry comparisons (conducted by Dr. David Silva) for an image were done using the *linear*, *spline3*, and *poly5* transformation algorithms. Histograms of the difference in the photometry for each star compared with the original image, in all three cases gave a gaussian distribution centered at 0.0. The stars in the wings were at the faint end of the magnitude scale as expected. This may be due to changes in the sky, or the under sampling of the fainter stars, or a combination of the two.

C.4 Flux Conservation

If the images are to be averaged to perform photometry on faint objects, then the flux conservation option should be turned on when registering images. The output pixels values are multiplied by the Jacobian of the coordinate transformation. This option is available in the tasks **magnify**, **register**, and **imlintran**.

C.5 Tasks Updating the World Coordinate System

Automatic updating of the world coordinate system (*WCS*) information in the image header is now enabled by default. This is true for all the tasks in the IMAGES package that are discussed in this manual. Note however that if the transformation has nonlinear terms, the *WCS* will not be updated for these terms, since there is as yet no agreed upon way to represent nonlinear terms of the *WCS* in a FITS header[2]. Only linear terms in the transformation will be updated properly in the image header. For more information on the *WCS* see the file iraf$sys/mwcs/MWCS.hlp or the *IRAF Newsletters #9* and *#12*, or check out the tasks **wcsedit** and **wcsreset** in the PROTO package.

[2]The IRAF image header is not a FITS header. But a problem will arise when the IRAF image is converted to a FITS image.

D Other Useful Documentation

On-line help pages are available for all the tasks mentioned in this document. More information on topics covered in this manual may be found in the following documentation that is accessible by anonymous ftp to **iraf.noao.edu** (or 140.252.1.1):

- **Rectifying and Registering Images Using IRAF**, by Lisa Wells, March 1994 (this manual). (iraf/docs/register.ps.Z)
- **A Beginner's Guide to Using IRAF[Draft]**, by Jeannette Barnes, August 1993. (pub/beguide.ps.Z)
- **A User's Guide to CCD Reductions with IRAF**, by Philip Massey, June 1992. (iraf/docs/ccduser2.ps.Z)
- **Specifications for the Aperture Photometry Package**, by Lindsey Davis, October 1987. (iraf/docs/apspec.ps.Z)
- **IRAF Newsletter Number 12**, *World Coordinate System Support in IRAF V2.10*, by Lindsey Davis & Frank Valdes, July 1992. (iraf/docs/newslet_12.ps.Z)
- **IRAF Newsletter Number 9**, *The New World Coordinate System (MWCS) Interface*, by Doug Tody, Lindsey Davis & Frank Valdes, February/June 1990. (iraf/docs/newslet_9.ps.Z)
- **Cleaning Images of Bad Pixels and Cosmic Rays Using IRAF**, by Lisa Wells. (in progress)
- **Mosaicing Images Using IRAF**, by Lisa Wells. (in progress)

For more information about the *imtool* display server see the on-line manual page (**man imtool**) at the Unix level.

The following documentation is accessed by anonymous ftp to **stsci.edu** (or 130.167.1.2):

- **STSDAS Users Guide**, Nov 1991. (documents/stsdas-docs/user/UserGuide/UserGuide_12.ps)
- **STSDAS Calibration Guide.** (documents/stsdas-docs/user/CalibrationGuide/CalibGuide_12.ps)
- **Quick Reference Cards for IRAF and STSDAS.** (documents/stsdas-docs/user/UserQuickRef.ps)