CORONAL DIAGNOSTIC SPECTROMETER

# SoHO

## CDS SOFTWARE NOTE No. 6

Version 4 | 25 July 1996

# SERTS IMAGE DISPLAY ROUTINES


W. Thompson
Applied Research Corporation
NASA Goddard Space Flight Center
Laboratory for Astronomy and Solar Physics
Code 682.1
Greenbelt, MD 20771, USA

William.T.Thompson.1@gsfc.nasa.gov
pal::thompson

These routines form the part of the SERTS subroutine library pertaining to image display. The goals of this software package are:

- To free the user from having to worry about the relationship between image pixels and screen pixels, and to provide a single interface for all graphics devices capable of image display. Supported devices include, but are not limited to, Sunview, X-windows, Tektronix 4100 series terminals and above, and PostScript and HP LaserJet (PCL) printers.
- To allow the user to display multiple images, and to control in a device-independent way the position of these images on the screen.
- To give the user control over which graphics device or window will be used for image display, and which will be used for ordinary graphics.
- To allow the user to control the way in which image data is displayed, and to allow this control to be specified image-by-image or for all images.
- To allow the user to easily combine images and line-drawn graphics, and to retrieve positions from displayed images.
- To make it easy for the user to display velocity data in a special color table, where blueshifts are shown in a blue color, and redshifts are shown in a red color.
- To allow the user to display one image in one color table, and another image in another color table, e.g. an intensity image next to a velocity image, where the velocity image would be displayed using the velocity color table.

In addition, it was desired that these routines be easy to use, and that the various pieces would interact together smoothly and seamlessly. All the routines here act together as a group, and cooperate with each other.

# 1 Displaying images

The basic command for displaying images is EXPTV. The name stands for "Expand to the TV" because it was initially used for expanding small SMM/UVSP images to fill the image display screen. The calling sequence for this routine is

```
EXPTV, image
```

When called, EXPTV executes the following steps:

1. The image display screen is cleared.
2. Scaling parameters are calculated based on the size of the image, and the size of the image display screen (or window) so that the image will best fit within the available area. The details of this process are described below.
3. The image is displayed at a size based on the calculated parameters, and centered within the screen.

4. A box is drawn around the edges of the image.

EXPTV will generally try to find an integer magnification or reduction scaling parameter to best match the available image display area. For instance, if the image display screen or window is $512 \times 512$ pixels in size, and the image to be displayed is $50 \times 100$ pixels, then EXPTV will magnify this by a factor of 5 to $250 \times 500$. On the other hand, if the image is $2000 \times 320$ pixels, then EXPTV will only display every fourth pixel, so that the resulting displayed image is $500 \times 80$ pixels in size.

There are a number of optional keywords that can be used to control exactly how EXPTV displays an image. These keywords are listed in section 8.

On window displays, one can also use the routine WDISPLAY, which works like EXPTV, except that the image appears in a window of its own, sized to fit. However, WDISPLAY normally doesn't resize the image unless the RESIZE keyword is set, or an explicit size is passed (see keyword SIZE in section 8).

## 2 Displaying multiple images

The routine PUT acts like EXPTV, except that it allows the user to put multiple images on the screen at the same time. There are three different ways that this can be done:

**By position within a regular array.** The image display screen can be split up into a regular $NX \times NY$ array of subwindows, where each subwindow has the same size and dimensions. Separate images can then be displayed in the $(IX, IY)$ subwindow with the command

```
PUT, image, IX, NX, IY, NY
```

For instance, suppose that one wanted to split the screen up into a grid of five boxes across, and four boxes vertically, and display an image in the box which was second from the right, and third down from the top. The command to do this would be:

```
PUT, image, 2, 5, 3, 4
```

**By relative position within a series.** Rather than specifying exactly how the images will be arranged on the screen, the user can simply specify that the image is the $I$th in a series of $N$ images with the command

```
PUT, image, I, N
```

PUT will then calculate the equivalent parameters $IX, NX, IY, NY$ based on the size of the image and the size of the screen or window. For instance, if $N$ is 6, then PUT might decide to put the images in two rows with three images each. This only works, however, if all $N$ images have the same dimensions.

**By normalized coordinates.** PUT can be instructed to place the image within a user defined subregion which is specified using normalized coordinates. This is done through the command

```
PUT, image, X1, X2, Y1, Y2, /NORMAL
```

The parameters $X1, X2, Y1, Y2$ give the coordinates of the corners of the selected subregion. For instance, if one wanted to display the image within an area using only the top 80% of the screen, with 5% margins on either side, then the command would be

```
PUT, image, 0.05, 0.95, 0.2, 1, /NORMAL
```

PUT takes the same optional keywords as EXPTV (see section 8).

# 3 Maintaining separate image and graphics displays

Often one wants to display images in one window or device while simultaneously generating plots in another window or device. One good example of this is when one has a dedicated image display device controlled from a regular graphics terminal sitting next to it. However, the principal also holds when using a windows device.

The command TVDEVICE specifies the graphics device or window to be used for displaying images. Once this is specified, all SERTS image display operations will be redirected to this device or window. TVDEVICE basically does for image display what SET_PLOT and WSET do for graphics. For instance, to redirect images to an Ivas image display screen, enter

```
TVDEVICE,'IVAS'
```

To redirect image display to window number 2, then enter

```
TVDEVICE,2
```

Note that when specifying a window as the image display device, the window must already exist.

It's not necessary for TVDEVICE to be called before using the SERTS image display routines– the image operations will just occur on whatever the current default device is. Redirection can also be turned off with the command

```
TVDEVICE, /DISABLE
```

or turned back on again with

```
TVDEVICE, /ENABLE
```

The SERTS image display routines also allow for redirection to be temporarily turned off with the /DISABLE keyword, e.g.

```
EXPTV, image, /DISABLE
```

In windowing systems such as X-windows, SunView, or Microsoft Windows, the easiest way to display both images and graphics is to use the command

```
TVSCREEN
```

This is a simplied interface to TVDEVICE which creates a default image display window (#2) in the upper left hand corner of the console. This window will be either $512 \times 512$ or $256 \times 256$ pixels in size depending on the total size of the console screen. It has been found that this makes a convenient complement to the default graphics window (#0) in the upper right corner of the screen.

There are several routines which perform functions which are already standard in IDL, but automatically use the device or window selected by TVDEVICE instead of the default. Currently, these include TVERASE, TVZOOM, and LOAD, which are equivalent to the standard routines ERASE, ZOOM, and LOADCT.

## 4 Manipulating flags

Almost all of the SERTS image routines, such as EXPTV and PUT, allow for the user to control their behavior with optional keywords. For instance, one can specify that all pixels with the value 32000 actually represent missing data, and should not be used for scaling the image, but should be displayed as black on the screen. This could be done by passing the MISSING keyword to EXPTV. For example:

```
EXPTV, image, MISSING=32000
```

Sometimes one wants to display an entire series of images, all of whom will have missing pixels represented by the value 32000. In such cases, it is more desirable to be able to specify this once rather than separately for each image. This can be done through the command SETFLAG. For example, the command

```
SETFLAG, MISSING=32000
```

would define the value 32000 as representing missing pixels for all subsequent images. This could then be later disabled with the command

```
UNSETFLAG, /MISSING
```

The command

```
ENABLEFLAG, /MISSING
```

would reenable a previously disabled missing pixel flag without reloading the value associated with it. Finally, the command

```
SHOWFLAGS
```

would show the values of all currently set image flags.

The quantities that can be loaded with the SETFLAG command are listed in section 8.

# 5 Combining images and graphics

The SERTS library allows the user to easily combine image display with line graphics. This can be done in one of three ways.

## 5.1 By using special commands

One way that the SERTS library allows images and line graphics to be combined is by substituting a SERTS routine for each of the regular IDL graphics routines. The available basic routines, and their standard IDL graphics equivalents are:

| Standard | TV equivalent |
|---|---|
| OPLOT | TVPLT |
| XYOUTS | TVOUT |
| CURSOR | TVPOS |
| AXIS | TVAXIS |
| CONTOUR | CONTV |

These routines operate in an image pixel coordinate system, which is separate from the more traditional IDL device, data, and normalized coordinate systems. For instance, if the image being displayed had $2000 \times 320$ pixels, the image pixel coordinates would run from 0–1999 and 0–319 regardless of how large the image was displayed on the screen.

The routine SETIMAGE can be used to switch between multiple images. For instance, if one used PUT to display two images side by side, and then wanted to read a position off the first image, the following commands would be used:

```
PUT, image1, 1, 2, 1, 1         ;First image on left
PUT, image2, 2, 2, 1, 1         ;Second image on right
SETIMAGE, 1, 2, 1, 1            ;Select first image
TVPOS, X, Y                     ;Read a position into X,Y
```

## 5.2 By using regular data coordinates

Effective September 1993, the SERTS routines allow associating data coordinates with images displayed using EXPTV or PUT. The data coordinate system is set up through the optional keywords ORIGIN and SCALE, or by using the routine TVAXIS (See also section 5.4). For example, the command

```
EXPTV, image, ORIGIN=[-2.5,3], SCALE=0.1
```

would associate the data values $X = -2.5$ and $Y = 3$ with the first (lower left) pixel in the image (specifically with the *center* of the pixel). Each pixel would have a height and width of 0.1 data values. One could then use ordinary graphics commands, such as OPLOT or the SERTS routine OCONTOUR, to overlay line graphics on the image.

Although data coordinates are automatically associated with displayed images, they have to specially activated for IDL to use them. This is to avoid interference with normal line graphics operations. To activate the data coordinates, use the SETIMAGE routine with the /DATA switch. For example, in the following example

```
TVSCREEN                              ;Creates window #2 for image display
PUT, image1, 1, 2, 1, 1
PUT, image2, 2, 2, 1, 1
SETIMAGE, 1, 2, 1, 1, /DATA
SETWINDOW, 2
OPLOT, x-array, y-array
SETWINDOW, 0
```

vector graphics are overlaid in the left-hand image. Note also that SETWINDOW is used instead of WSET. This routine, from the SERTS graphics devices library, keeps track of the values of the plotting system variables as a function of window, and is used internally by SETIMAGE,/DATA. When used with the /DATA qualifier, the SETIMAGE routine also can take the /DISABLE keyword as appropriate (see section 3).

The /DATA switch can also be applied to those routines which display images (e.g. EXPTV, PUT, WDISPLAY) and to TVAXIS, to signal that the data coordinates for those images should be activated immediately. For example

```
EXPTV, image, /DATA
OPLOT, x-array, y-array
```

### 5.3 By using PLOT_IMAGE and OPLOT_IMAGE

The routine PLOT_IMAGE will display an image with axes around it. Then, ordinary graphics calls, such as CURSOR, OPLOT, etc., can be used instead of the equivalents given in section 5.1. As with EXPTV, PLOT_IMAGE can also be used to associate physical (data) values with the coordinates of image pixels, by specifying the position of the first pixel in the image, and size of the pixels. For example, the command

```
PLOT_IMAGE, image, ORIGIN=[-2.5,3], SCALE=0.1
```

specifies that the first pixel in the image is at the position (-2.5,3) in data units, and that the pixels are each $0.1 \times 0.1$ units in size. (The scale could also be specified to be different in the two dimensions.)

The basic difference between this routine and EXPTV is that PLOT_IMAGE is intended to act as much like an ordinary graphics routine, such as PLOT or CONTOUR, as possible. Unlike EXPTV, the size of the image will have no simple relationship to the number of device pixels across the window. Instead the image size will be controlled by the same factors that control the size of a regular graphics operation, such as !X.MARGIN and !Y.MARGIN. In fact, just about anything that can be done with PLOT can also be done with PLOT_IMAGE. There is also an OPLOT_IMAGE routine, which acts like OPLOT.

Because PLOT_IMAGE and OPLOT_IMAGE are treated like a graphics operation, they are not subject to image redirection (section 3), unlike the other SERTS image display routines. Nor does one use SETIMAGE to display multiple images with PLOT_IMAGE—instead one uses !P.MULTI or SETVIEW from the SERTS graphics devices routines.

### 5.4 By using TVAXIS

The TVAXIS routine is used to place labels on displayed images. One way to use TVAXIS is to use the coordinate system established when the image was displayed, i.e. through the ORIGIN and SCALE keywords (or their defaults if not passed).

However, TVAXIS can also be used to associate new data coordinates with already displayed images. There are two ways that TVAXIS can specify the data coordinates associated with an image. One way (as of 20 March 1995) is to use the same ORIGIN and SCALE keywords as are used with EXPTV or PLOT_IMAGE. These keywords work in exactly the same way as they do in the other SERTS image display routines.

Two other keywords which can be used in TVAXIS to control the conversion of pixel coordinates to data coordinates are XRANGE and YRANGE. These, however, work in a somewhat different fashion than ORIGIN and SCALE. Instead, they are written to act like their counterparts in the standard AXIS routine for line graphics. They specify the entire range of the image in the X and Y directions, *including the component due to the width of the pixels.* Thus, the values of XRANGE represent the data coordinates from the left edge of the leftmost pixel to the right edge of the rightmost pixel. This is in contrast to the ORIGIN keyword, which refers to the *center* of the first pixel.

As an example, suppose that the first pixel in a $10 \times 10$ image was at coordinate (1,1), and that the spacing between pixels was 1 in both directions. One could achieve the same effect by either using the commands

```
TVAXIS, XAXIS=0, YAXIS=0, ORIGIN=[1,1], SCALE=1
```

or by using

```
TVAXIS, XAXIS=0, YAXIS=0, XRANGE=[0.5,10.5], YRANGE=[0.5,10.5]
```

The centers of the pixels range from 1 to 10 in both directions. To calculate the total range, one needs to subtract half a pixel width from the first pixel position and add half a pixel width to the last pixel position.

One can use TVAXIS to associate data coordinates with an already displayed image without actually drawing any axes. To do this, simply call TVAXIS without the XAXIS or YAXIS keywords.

## 6 Converting between coordinate systems

The routine IM_CONV_COORD can be used to convert between image pixels and other coordinates, in a manner similar to the standard IDL routine CONVERT_COORD. It recognizes four different coordinate systems:

**DATA:** The standard IDL data coordinate system, if activated through the /DATA keyword (Section 5.2), or through the PLOT_IMAGE command (Section 5.3).

**DEVICE:** The standard IDL device coordinate system.

**NORMAL:** The standard IDL normalized coordinate system.

**IMAGE:** The image pixel coordinate system.

Keywords are used to specify the input and output formats. For example, if one were processing a widget event position, which is given in device coordinates, and wished to convert these to image pixel coordinates, one would use commands like the following:

```
X = EVENT.X
Y = EVENT.Y
IM_CONV_COORD, X, Y, /DEVICE, /TO_IMAGE
```

which would convert the variables X and Y from device to image pixel coordinates.

# 7 Displaying velocity images, and manipulating color tables

The SERTS routines can also be used to display velocity images in a special color table where blueshifted velocities are displayed in blue, and redshifted velocities in red. The following commands will display a velocity image using this special table.

```
LOAD_VEL
EXPTV, image, /VELOCITY
```

The /VELOCITY keyword is required for the EXPTV command, because the image must be specially scaled so that pixels with the velocity value 0 are in the center of the color table (grey), and the positive (blue) and negative (red) velocities are scaled around it. Also, the bottommost color in the velocity color table is reserved for missing pixels (black) and the topmost color is reserved for drawing graphics and labels (white).

It is also possible for intensity images and velocity images to be displayed side by side, each in its own color table. For example:

```
LOADCT, 3                                ;Use red temperature table
COMBINE_VEL                              ;Combine with velocity table
PUT, int, 1, 2, /COMBINED                ;Put intensity on left
PUT, vel, 2, 2, /COMBINED, /VELOCITY     ;And velocity on right
```

One can also combine two ordinary color tables. For example:

```
LOADCT, 3                                ;Select first color table
COMBINE_COLORS, /LOWER                   ;Save lower color table
LOADCT, 5                                ;Select second color table
COMBINE_COLORS                           ;Combine the color tables
PUT, int1, 1, 2, /COMBINED, /LOWER       ;Display first image on left
PUT, int2, 2, 2, /COMBINED               ;And second image on right
```

The velocity and combined color tables can be manipulated through the routines VEL_STRETCH and INT_STRETCH, which work in a similar fashion to the standard IDL routine STRETCH.

On displays which support X-windows widgets there is a very powerful interface to all these SERTS color table routines. This widget procedure is called XLOAD, and replaces the standard XLOADCT which comes with IDL.

# 8 Keywords

There are a number of keywords which are commonly used in the SERTS image display routines. A list of those keywords which show up in a number of these routines are shown below. Those marked with "†" can be manipulated with the flag routines described in section 4. There are also other more routine-specific keywords that aren't listed here. In particular, the graphics routines will support some of the standard IDL graphics keywords.

Note that, when a keyword is described as being "set", this means that the keyword is passed in the form "/keyword". Not being set means that the keyword is not passed at all.

**ADJUST:** Used in EXPTV and PUT to adjust the pixel size separately in the two dimensions, so that the physical scale given by the SCALE parameter is the same along both axes. For example, if a 100x100 image is displayed with

```
EXPTV, A, SCALE=[2,1], /ADJUST
```

then it will be shown twice as wide as it is high. Use of this keyword forces NOEXACT to also be set. Also, NOSQUARE is ignored.

**COLOR:** Color index used for drawing graphics and/or text.

**COMBINED†:** Signals that the image is to be displayed in one of two combined color tables. Can be used by itself, or in conjunction with the VELOCITY or LOWER keywords.

**DATA:** Specifies that the data coordinates associated with a displayed image are to be activated. This allows ordinary IDL graphics routines to interact with the displayed images. See also ORIGIN and SCALE.

**DISABLE:** Used to disable redirecting image display functions to a special device or window. When set, the current default device or window is used instead.

**LOWER:** If set, then the image is placed in the lower part of the color table, rather than the upper. Used in conjunction with COMBINED keyword. When used with the COMBINE_COLORS routine, specifies that the current color table is to be loaded as the lower part of a combined color table.

**MAX†:** The maximum value to be considered in scaling the image, as used by BYTSCL. The default is the maximum value of the image.

**MIN†:** The minimum value to be considered in scaling the image, as used by BYTSCL. The default is the minimum value of the image.

**MISSING†:** Value representing missing pixels.

**NOBOX†:** If set, then no box is drawn around the image. Setting this also has the effect of allowing more space to be used for the image, because otherwise the software will reserve a small amount of border area around the image. If NOBOX is set, then the image can extend all the way to the edge of the screen.

**NOEXACT†:** Ordinarily, the size of the image will be scaled up or down by integer scaling factors. If NOEXACT is set, then scaling by noninteger factors is allowed. Generally, this allows the image to fill the maximum amount of space available.

**NORMAL:** Used with SETIMAGE and PUT to signal that normalized coordinates are being used to define the subarea to be used instead of the regular matrix description ordinarily used.

**NOSCALE†:** If this keyword is set, then the image is not scaled into a byte array. It is assumed that the image has already been scaled somehow, for example through the BSCALE function.

**NOSQUARE†:** Normally, images are scaled equally in both directions, so that image pixels appear square on the screen. When NOSQUARE is set, the scaling is independently calculated in the two directions, and image pixels are allowed to appear rectangular. Generally used when the two dimensions of an image are in different units, e.g. spatial in one dimension and wavelength in the other.

**ORIGIN:** The value of this keyword is a two dimensional array containing the coordinate value in physical units of the center of the first pixel in the image, i.e. the pixel in the lower left corner. If not passed, then `ORIGIN=[0,0]` is assumed.

**SCALE:** This keyword gives the pixel scale (i.e. the size of a pixel) in physical units. It can have either one or two elements—if a single number then the scale is assumed to be the same both directions. If not passed, then the scale is assumed to be unity.

**RELATIVE†:** Size of area to be used for displaying the image, relative to the total size available. The value passed must be between 0 and 1. The default is 1.

**SIZE†:** This keyword performs two functions: as an input parameter it allows the user to specify the scaling relationship between image pixels and device pixels, and as an output parameter it returns the same scaling relationship.

As an example of using SIZE as an input parameter, the command

```
WDISPLAY, image, SIZE=2
```

would display the image at twice true size (two device pixels for every image pixel), while

```
WDISPLAY, image, SIZE=0.5
```

would only display every other image pixel on the screen.

If the input value of SIZE is undefined, or less than or equal to zero, then the value of SIZE is ignored and the image is sized normally. The calculated size used for displaying the image (from SCALE_TV) is then returned in SIZE as an output. This value could then be used as input to subsequent calls to force the image pixel size to be the same for multiple images, or to define a specific relationship between the size of one image's pixels and anothers. For example,

```
SIZE = 0                                ;Make SIZE an output parameter
PUT, image1, 1, 3, SIZE=SIZE            ;Get the SIZE based on image #1
PUT, image2, 2, 3, SIZE=SIZE            ;Image #2 has the same sized pixels
PUT, image3, 3, 3, SIZE=2*SIZE          ;Image #3's pixels are twice as big
```

**SMOOTH†:** If set, then bilinear interpolation (via the CONGRDI routine) is used to expand the image instead of nearest neighbor interpolation (via CONGRID). The effect is that the displayed image will vary smoothly rather than showing the pixel boundaries. If the image is not expanded when displayed, then this keyword has no practical effect. This keyword is also ignored when writing PostScript output.

**TOP†:** The maximum value of the scaled image array, as used by BYTSCL. The default is to use the entire range of colors available (!D.N_COLORS-1).

**VELOCITY†:** If set, then the image is scaled using FORM_VEL as a velocity image. Can be used in conjunction with the COMBINED keyword.

**XALIGN†:** Alignment within the image display area. Valid inputs range between 0 (left) to 1 (right). The default is 0.5 (centered).

**YALIGN†:** Alignment within the image display area. Valid inputs range between 0 (bottom) to 1 (top). The default is 0.5 (centered).

Not all keywords are compatible. For instance, one wouldn't use both SIZE and RELATIVE at the same time as input parameters. Nor would one use both SIZE (either input or output) and NOSQUARE at the same time.

## 9 Printing images

The routines described here are compatible with all graphics devices supported by IDL. However, the appearance may be somewhat different from one device to another, since the number of pixels and the aspect ratio are not the same. For instance, images may appear bunched closer together when printed on a PostScript printer than they do on the screen. Also, text labels or graphics written with reference to normalized coordinates will appear in slightly different places relative to the images when displayed on the screen and in the printed output.

One thing that can be done to help make the output look more the same between different devices is to use the /NOEXACT keyword and set RELATIVE to some appropriate value less than one (e.g. 0.95). Using PUT or SETIMAGE with the /NORMAL option is also useful for a completely device-independent look.

Color tables have to be loaded separately for the screen and for the printer—otherwise, you may get an incorrect or distorted color table on your printed output. On PostScript devices, the color tables must be loaded *before* displaying any images. Therefore, it is recommended that the color table be loaded (and adjusted if necessary) immediately after opening the plot file.

The number of available colors in the graphics file may be different from that of the screen. Color PostScript printers generally support 256 colors. Windowing systems such as X-windows, Sunview or Microsoft Windows usually have somewhat fewer colors (typically around 230) because a certain number of colors will be reserved for the window manager, or for other programs also using the same display. Thus the color associated with any particular index will be different on different displays. For example, the color represented by the color index 50 will be different on the screen than it is in the PostScript output. The difference could be small if the numbers of colors between the two devices are comparable, and the color table is smoothly varying, or the difference could be quite important.

On most screens the images are typically displayed against a black background with text and graphics usually drawn in white. However, in PostScript output the images are displayed against a white background, and the text and graphics are generally drawn in black. If a white color is desired, for example to overplot on top of an image, then using "COLOR=!D.N_COLORS-1" will force the use of the top color (which in most color tables is white) regardless of the graphics device. For example

```
EXPTV, image1
CONTV, image2, COLOR=!D.N_COLORS-1
```

will produce the same output on all graphics devices.

If you absolutely must have a black background on your PostScript output, one way to do it is to first display a completely black image that takes up the entire screen, and then display over it. For example

```
EXPTV, INTARR(10,10), /NOEXACT, /NOBOX, /NOSQUARE
EXPTV, image, /NORESET
```

## 10 Miscellaneous routines

Some additional image display and manipulation routines included in the SERTS library include:

**ADJUST_COLOR:** Routine for adjusting the color table based on the position of the graphics cursor. Equivalent to ADJCT from IDL version 1.

**BLINK:** Blinks two images against each other by manipulating the color tables.

**COLOR_BAR:** Displays a color bar on the image display screen. The position and size can be passed explicitly, or interactively selected by the user.

**CROSS_CORR2:** Calculates the cross correlation matrix of two images.

**TVSUBIMAGE:** Uses TVBOX to select a subimage from a displayed image.

**GOOD_PIXELS:** Function which returns all the pixels in an image array which are not equal to the missing pixel flag. Generally used when plotting histograms, or performing other statistical operations on the data.

**HISCAL:** Performs histogram equalization on images.

**INTERP2:** Performs two dimensional interpolation on a series of data points. Used by routines such as TVPROF.

**LABEL_IMAGE:** Writes a label next to a displayed image. The label can be either above, below, or to either side.

**POLY_VAL:** Returns the values and positions of all the pixels within a polygon drawn on top of an image.

**SIGRANGE:** Rescales the image based on the most significant 90% of the data. The lowest 5% and highest 5% of the data are rescaled to the limits of the central 90%. This routine is particularly useful for data where a few of the pixels are at extreme values compared to most. The fraction of pixels making up the most significant range can be adjusted.

**TVBOX:** Interactively selects the coordinates of a rectangular area from a displayed image.

**TVPOINTS:** Selects a series of points on a displayed image, to define a path or polygon. Used by routines such as TVPROF and POLY_VAL.

**TVPRINT:** Reads the pixels and color table from the image display screen, and generates PostScript output, which is sent to the printer or saved in a file.

**TVPROF:** Allows the user to draw a series of points on a displayed image, and then extracts a profile from the image along the drawn path. Uses the additional routines TVPOINTS, PROF, and INTERP2.

**TVPROFILE:** Interactively plots horizontal or vertical profiles from a displayed image.

**TVREAD:** Read the contents of the image display screen into an array.

**TVVALUE:** Allows the user to point to pixels and obtain their position and value.

**XBLINK:** Uses XMOVIE to blink together two images.

**XGAMMA:** Widget routine which allows the user to adjust the gamma value for the screen intensity.

**XMOVIE:** Simplified interface to the widget animation routine XINTERANIMATE.

# 11 Internal routines

These routines are used internally by the SERTS image display software. The user shouldn't have to call these routines directly. The exception would be when developing new routines that interact with this software.

**SCALE_TV:** Calculates the scaling parameters for displaying an image.

**EXPAND_TV:** Expands or contracts the image, and displays it according to the parameters calculated by SCALE_TV.

**STORE_TV_SCALE:** Stores the scaling parameters calculated by SCALE_TV for later recall.

**GET_TV_SCALE:** Recovers the scaling parameters stored by STORE_TV_SCALE.

**BSCALE:** This routine scales image arrays into byte arrays that can be displayed on the screen. Similar to the function BYTSCL except that it can be used with velocity and combined color tables.

**FORM_INT:** Scales intensity images that are to be used in a combined color table.

**FORM_VEL:** Scales velocity images.

**GET_IM_KEYWORD:** Gets the values of image flags as stored by SETFLAG, or as passed by a keyword.

**IM_KEYWORD_SET:** Similar to the standard function KEYWORD_SET, but also looks for values stored by SETFLAG.

**TVSELECT:** Select the device defined by TVDEVICE.

**TVUNSELECT:** Restore the device active before TVSELECT was called.

**ADJUST:** Similar to using a statement like "min > array < max", but leaves pixels with the missing pixel flag value alone.

**CONGRDI:** Used to generate a smoothed interpolation of the image when the /SMOOTH keyword is set. Similar to CONGRIDI from IDL version 1, but uses a different interpolation strategy.

In distributed copies of this software, there may also be included additional routines from other parts of the SERTS library which are used by the SERTS image display routines.

It should also be noted that many of the SERTS image display routines allow for the passing of the image scaling parameters directly. It is anticipated that this capability will only be used in extremely rare circumstances.