Karma User Manual

Richard Gooch
http://www.safe-mbox.com/~rgooch/

December 26, 2006

Richard Gooch may be reached by email at `rgooch@safe-mbox.com`
or on the WWW at `http://www.safe-mbox.com/~rgooch/`

Contents

Chapter 1

Introduction

Karma is a general purpose programmer's toolkit and contains KarmaLib (the structured library and API) and a large number of modules (applications) to perform many standard tasks. This manual describes the many visualisation tools which are distributed with the Karma library.

This document is written for Karma version 1.7.26 , which is probably my "experimental" version. Most of this manual will still be relevant to the previously released binary-only (or "beta") version, since binary releases come every few weeks or so. Full public releases come once or twice a year, so this document may talk about several new things not available in the last public release of Karma.

This document is available as compressed PostScript at:
`ftp://ftp.atnf.csiro.au/pub/software/karma/user-manual.ps.gz`.
You can download a fancy colour front cover for this document at:
`ftp://ftp.atnf.csiro.au/pub/software/karma/user-cover.ps.gz`.

Separate on-line documents (the Karma Reference and Programming Manuals) may be found via the world-wide-web at the **Karma home page** at `http://www.atnf.csiro.au/karma/`. The visualisation tools are available via the **Karma home page** or you can go directly to the **Karma ftp site** at `ftp://ftp.atnf.csiro.au/pub/software/karma/`.

The programmes available at the moment are:

- volume rendering of data cubes <**xray**> (chapter 5)
- play movies of data cubes <**kvis**> (chapter 3)
- inspecting multiple images and cubes at the same time <**kvis**> (chapter 3)
- slice a cube <**kslice_3d**> (chapter 7)
- superimposing images <**kvis**> (section 11.2) and <**kvis**> (section 11.3)
- interactive position-velocity slices <**kpvslice**> (chapter 6)
- look for expanding shells <**kshell**> (chapter 8)

- interactive co-ordinate placement <**koords**> (chapter 9)
- rectangular to polar gridding of images <**kpolar**> (chapter 10)

1.1 Supported Data Formats

All the visualisation tools support a variety of data formats, including:

- **AIPS** this is the format used by the AIPS package (Astronomical Image Processing System), sometimes called "Classic AIPS" or "AIPS1", not to be confused with "AIPS++" (which is sometimes called "AIPS2"). This data format uses a catalogue file which is automatically scanned by the file browser and each available dataset is given an entry in the browser, with the string "AIPS" shown in the left-hand column. By default, *all* map files in the catalogue are shown. You may set the `AIPS_ID` environment variable to your AIPS user ID to avoid seeing files belonging to other users. To see AIPS files on an AIPS disc, you will need to change directory to the AIPS data area (i.e. where all those odd-looking "`CBD??????.???;`" files are kept). You can use the file browser to change directories. Byte-swapped data are automatically detected and corrected
- **AIPS++** this is the format used by the AIPS++ package (Astronomical Information Processing System), not to be confused with "AIPS". This is a directory-based data format like Miriad. It is a new package still under development. The Karma support for AIPS++ requires the <**image2fits**> programme to be in your `PATH`. The source code is located somewhere in the AIPS++ source tree. A precompiled binary is available at
  `ftp://ftp.atnf.csiro.au/pub/software/karma/other-packages/aips++/`
  for your convenience
- **DRAO** this is the format produced by the Dominion Radio Astronomy Observatory (Penticton, B.C., Canada) reduction software. This data format uses a catalogue file which is automatically scanned by the file browser and each available dataset (whether a single image file or a "family" of files for a cube) is given one entry in the browser, with the string "DRAO" shown in the left-hand column. Unlike the DRAO software, **Karma** requires no environment variable to cope with byte-swapped data; they are automatically detected and corrected. This format supports loading of partial 3-dimensional datasets (section 1.1.3)
- **FITS** this is a popular format amongst astronomers, allowing you to move data between many different software packages. It is more efficient to use <**fits2karma**> to convert to **Karma** format first. For the file browser to recognise a **FITS** file, the extension should be either `.fits`, `.fit`, `.fts` or uppercase versions. Files with extension `.efits` are treated specially, in that only the header is read. These are called "empty" **FITS** files,

where the image data are set to all zeros. This format supports loading of partial datasets (section 1.1.3)

- **GIF** another popular image format. This requires the NetPBM tools to be installed on your machine
- **GIPSY** this is the format for the **GIPSY** data reduction package. It is more efficient to use <**gipsy2karma**> to convert to **Karma** format first. This format supports loading of partial datasets (section 1.1.3)
- **IRAF** this is the format for the **IRAF** optical astronomy analysis package. The current implementation requires **IRAF** to be installed (because it uses the **wfits** task in **IRAF**), and will create a temporary file which is automatically deleted
- **Karma** this is the native data format. Reading and writing Karma files is the most efficient. These files may be memory-mapped upon load, which can give enormous speed benefits and reduce swap (virtual memory) requirements. Note that **Karma** files have the extension `.kf` and that this extension is automatically produced by any **Karma** software
- **Miriad** this is the image format used by the **Miriad** data reduction package. It is more efficient to use <**miriad2karma**> to convert to **Karma** format first. If your **Miriad** file has no mask attached to it, then the file may be memory-mapped just like a **Karma** file, which means loading may be just as fast as loading **Karma** files. For this reason, it is worthwhile to avoid creating masks for **Miriad** files if you can help it
- **PNM/PPM/PGM** this is a common image interchange format. The format does not support anything fancy like co-ordinate systems, but it is useful for converting between the many different image formats out there
- **Simple FITS** this is a derivative of the **FITS** format which is easier for the human to manipulate. Lines in the header are separated by newline characters rather than having to be padded to 80 characters. The data directly follows the `END` keyword and must be in plain ASCII format, a single value per line. The blank value for floating point data is `1e30`. The filename extension should be `.sfits`
- **Starlink Image** this an astronomical reduction package used by many optical astronomers. The file extension is `.sdf` and requires the <**sdf2karma**> programme to be in your `PATH`. A precompiled binary is available at `ftp://ftp.atnf.csiro.au/pub/software/karma/other-packages/starlink/` for your convenience
- **Sun Rasterfile** another popular image format, having the distinct advantage of not using a patented compression algorithm (unfortunately the **GIF** format uses a patented image compression algorithm)
- **Targa TrueVision** (tga files) yet another image format. This requires the NetPBM tools to be installed on your machine

- **TIFF** another popular image format. This requires the NetPBM tools to be installed on your machine.

In addition, the automatic decompression of gzipped and bzip2'ed **Karma** and **FITS** files is also supported by the file browser <**Filepopup**> widget (section 2.1).

If your data are not in one of the supported formats, you will need to convert it to a supported format like **FITS**.

1.1.1 Converting Other Data Formats to Karma Format

A number of command-line utilities are provided to convert between other data formats and **Karma**. These are described below:

- **FITS** requires the <**fits2karma**> programme, which is used:

  ```
  fits2karma <fits file> <karma file>
  ```

  <**fits2karma**> will attempt to trap most deviations from standard FITS and continue gracefully. However, a truly bizzare FITS file may cause <**fits2karma**> to reject the conversion. In this rare case, please notify Richard Gooch, who will attempt to add a trap so that the data can be converted.

  Also note that <**fits2karma**> will by default truncate axes so that they are divisible by 4 or larger. The reason for this is that this then allows the programme to tile the data (tiling is a way of organising data so that most ways of accessing it are faster). If you really don't want to loose up to 3 co-ordinate points along an axis, you can run <**fits2karma**> as follows:

  ```
  fits2karma -allow_truncation off <fits file> <karma file>
  ```

  But remember: for the sake of a little bit of data, you may be throwing away enormous speed benefits!

- **Miriad** requires the <**miriad2karma**> programme, which is used:

  ```
  miriad2karma <miriad file> <karma file>
  ```

  This programme also tries to tile data like <**fits2karma**> does.

- **GIPSY** requires the <**gipsy2karma**> programme, which is used:

  ```
  gipsy2karma <gipsy file> <karma file>
  ```

  This programme also tries to tile data like <**fits2karma**> does

1.1.2 Converting Between Data Formats

There are also command-line utilities which allow you to convert between other data formats, which are provided as a convenience. These are described below:

- <**miriad2gipsy**> will convert from **Miriad** format to **GIPSY** format, which is used thus:

  ```
  miriad2gipsy <miriad file> <GIPSY file>
  ```

  This programme is obsoleted by the more general <**ktranslate**>

- <**ktranslate**> will convert from any format supported by **Karma** to another format. It is used thus:

  ```
  ktranslate <input file> <output file>
  ```

  The input and output formats are guessed by the programme, based on the filename extensions. You can force the format if you wish. An example of this is:

  ```
  ktranslate -intype fits -outtype miriad - <input file> <output file>
  ```

  The choice of supported output formats is limited to **Karma**, **PPM**, **FITS**, **Sun Rasterfile**, **Miriad** and **GIPSY**. If possible, the conversion process is "streamed", to avoid the need to allocate large amounts of virtual memory.

1.1.3 Loading partial files

Some data formats support loading of partial datasets. With these formats, if part of the data are missing, the part of the dataset that is available can be loaded (if desired). This is useful if you want to see a preview of a dataset that is currently being written by another process.

1.2 Adding your own Data Formats

The visualisation tools also provide a mechanism for you to add support for other data formats not already supported by the **Karma** library. You will need a programme that converts from your data format to either **Karma**, **FITS** or **PNM** format, and for some data formats (in particular directory-based formats) you will need to provide a tester programme which determines if a dataset is of a particular type.

To set this up, you need to have a file `~/.karma/data-filters` which contains rules on how to convert data formats. A system-wide file maintained by your local **Karma** installer is also searched, and is called `$KARMABASE/site/share/data-filters`. Finally, the file `$KARMABASE/share/data-filters` is scanned to load any filters which come with the Karma distribution. Your own data filters override the system data filters, which in turn override the Karma distribution filters.

A typical file would contain:

```
# Extension     Converter       Tester          Output  Format Name
imh             iraf2fits       -               FITS    IRAF Image
sdf             sdf2karma       -               Karma   Starlink Image
DIRECTORY       dd2fits         isdd            FITS    A directory dataset
```

```
DIRECTORY       aips++2fits     isaips++        FITS    AIPS++ Image
gif             giftopnm        -               PNM     GIF Image
tga             tgatoppm        -               PNM     Targa TrueVision
```

This would use the <**iraf2fits**> programme to convert files with `.imh` extensions to **FITS**. The **FITS** data would then be read in with the standard **FITS** reader in **Karma**. Similarly, files with `.sdf` extensions would be converted to **Karma** format using the <**sdf2karma**> programme, which would write out **Karma** data. Note that it is much more efficient to have a filter which generates **Karma** than **FITS**.

Note the <**giftopnm**> and <**tgatoppm**> filters are both registered as producing **PNM** files, whereas <**tgatoppm**> appears to produce **PPM** files. This is not a problem, since **PNM** (Portable aNy Map) is the generic format and **PPM** (Portable Pixel Map) is a specific subset of **PNM**.

The converter programmes are called with a single argument, that being the name of the input file. The output **Karma**, **FITS** or **PNM** data must be written to the standard output.

You will note that the tester programme supplied for the `.imh` and `.sdf` formats is `-` which means no tester is needed for these formats.

The special extension name `DIRECTORY` signifies that this data format is a directory-based one (examples of such formats are Miriad and AIPS++ images). Directory-based formats *require* a tester programme in order to determine if the directory contains a dataset (in which case it should be loaded) or just a plain directory which should be entered. The tester programme must exit with status 0 if it recognises the dataset, otherwise it should exit with some other value. If it returns the value 16 no warning will be given. It is given the name of the dataset directory as a single argument.

The file browser described in section 2.1 will show all files which are supported by data filters, making the loading of extra data formats transparent to the user.

If you want to write your own data filters, see the section on **datafilters** (`http://www.atnf.csiro.au/karma/programmer-manual/datafilters.html#pubdatafilters`) in the Karma Programming Manual.

The converter and tester programmes may be specified with absolute pathnames or you may give a plain filename and the `PATH` is searched. As well as the normal absolute pathnames (i.e. those with a leading "/" character) you may also specify an expression, such as:
`$KARMABASE/site/$MACHINE_OS/bin/sdf2karma`
or, alteratively:
`${KARMABASE:-/usr/local/karma}/site/bin/sdf2karma`
which is useful if you want to specify `$KARMABASE/site/bin/sdf2karma` if the `KARMABASE` environment variable is defined, otherwise use
`/usr/local/karma/site/bin/sdf2karma`
instead. This is a good way of providing a default location.

1.3 Units

Karma follows the FITS conventions for units. In other words, SI units are used. Please see the FITS standard for more information.

1.4 Co-ordinate Systems

The Karma data format supports simple regular co-ordinate systems as well as irregularly-spaced co-ordinates. In addition, the curvilinear co-ordinate systems (i.e. Right Ascension/Declination and Galactic Longitude/Galactic Latitude) commonly found in astronomy are supported. Many common projections for these curvilinear co-ordinate systems are supported, as is the polynomial-based "projection" used for DSS (Digital Sky Survey) images.

The co-ordinate system support is tightly integrated with much of the Karma internal infrastructure. In general, mixing of data with different projections is handled correctly. For example, displaying overlays and contours over a greyscale image will work correctly even if the datasets have different rotations, scales and projections. Support for mixing different co-ordinate systems (i.e. RA/DEC overlaid with GLON/GLAT) is not yet supported, nor are data of mixed epochs.

Chapter 2

Common Features

Some features of different programmes are the same. You may also notice that parts of one programme may also be available as a separate programme or also appear in other programmes. This is due to the modular approach provided by **Karma**. Most of the window-based control panels you see are in fact "widgets", and these are reused many times in the same programme or in different programmes.

2.1 Loading Data

All programmes have the same interface to load data. Clicking (left) on **Load** or **File** or whatever it is called, will show a window with the files that you can load. This will be all files with extension `.kf`, `.fts` or `.fits`. Also subdirectories and the parent directory are shown. Clicking on these changes the directory. If a directory is in fact a **Miriad** image dataset, clicking on it will load that dataset. Note that **GIPSY** files consist of two files, one with a `.descr` extension and another with a `.image` extension. Most of the software will simply display a `.gipsy` file instead, since that makes the directory browser look less cluttered.

The <**Filepopup**> widget provides the file browsing interface. This widget allows the user to browse a directory tree and select files. All the files and directories the application wants the user to see are displayed. Any one of these may be selected by clicking the left mouse button over the filename. The "D" character is used to denote directories, while the "F" character is used to denote ordinary files. Above the list of files and directories is shown the current directory for the browser. If you edit the text and press return the current directory is changed appropriately. The "~" notation is supported, as well as other simple Bourne Shell-like expansions of environment variables using the `$variable`, `${variable}` and `${variable:-word}` notations. The following controls are provided:

- **close** close the window
- **rescan** rescan the directory in case a new file has appeared
- **pin** enable to force the fileselector window to stay "pinned" up even if files are selected. The default is to close the window once a file is selected

- **new** create a new file browser. The current directory will be independent of the original file browser
- **filter** if enabled, selected files are not automatically loaded, rather, the <**LoadAndDecimate**> widget (section 2.1.1) is displayed
- **partial** if enabled, support loading of partial files. This is useful if you want to see a preview of a dataset that is currently being written by another process. See the section on **partialdata** (`http://www.atnf.csiro.au/karma/user-manual/partialdata.html#pubpartialdata`) data in the Karma User Manual
- **writable** if enabled, the file may be modified later (i.e. by the image editing facility)
- **pull** change the directory in this browser to match the most recent directory change in *any* browser (i.e. pull the most recent directory change)
- **push** change the directory in all other browsers to match the directory for this browser (i.e. push this directory to others)

Convenience buttons for jumping to commonly-used directories may be configured by the local Karma administrator or by the user. The following files are searched:

- `$KARMABASE/share/browser-dirbuttons`
- `$KARMABASE/site/share/browser-dirbuttons`
- `$HOME/.karma/browser-dirbuttons`

in that order. The format of this file is a sequence of ASCII lines, with each line containing the name of the button followed by the directory to jump to. For example:

```
survey  /nfs/survey
mydata  /nfs/data1/users/fred
```

See Figure 2.1 for a screen snapshot.

2.1.1 Subsetting/Decimation of Array Data

Sometimes, your image or datacube is too big to handle efficiently. If you have a file which is much bigger than your physical memory (RAM), then most of the visualisation tools will run slower than a lame dog. It generally does not make sense to visualise a dataset bigger than RAM. To get around this, the visualisation tools provide a mechanism whereby you can decimate a 2-dimensional (image) or 3-dimensional (cube) array. Decimation involves taking every `N` values along an axis and averaging them together to produce a single output value, then jumping to the next group of `N` input values. If you use a decimation factor of 2 for each axis, the output cube is 8 times smaller (4

Figure 2.1: Screen snapshot of fileselector panel

times smaller for an image). You can also extract a subcube, if you are only interested in a small section of the data.

Note that the decimator allocates some temporary 2-dimensional buffers, so when decimating a 2-dimensional array, the memory requirements may be significantly higher than reported for the output array.

The <**LoadAndDecimate**> widget provides the decimation interface. This widget controls the process of loading and decimating a datacube. The user can skip/average data values along each dimension as well as extract a subcube. The following controls are provided:

- **Cancel** close the window
- **Load** load the file using the specified parameters
- **Abort** abort the loading process
- **Auto Save** if enabled, the decimated subcube is automatically saved to disc
- **X Start** select the starting position of the subcube in the X dimension
- **Y Start** select the starting position of the subcube in the Y dimension
- **Z Start** select the starting position of the subcube in the Z dimension
- **X End** select the end position of the subcube in the X dimension
- **Y End** select the end position of the subcube in the Y dimension
- **Z End** select the end position of the subcube in the Z dimension
- **X Skip** select the skip factor along the X dimension. The default value of 2 will make every 2 input values along X averaged together
- **Y Skip** select the skip factor along the Y dimension. The default value of 2 will make every 2 input values along Y averaged together
- **Z Skip** select the skip factor along the Z dimension. The default value of 2 will make every 2 input values along Z averaged together
- **Output filename** change this to select the name of the file which is saved to disc

Under the first row of buttons, the size of the file (in data elements) is displayed and the memory size required by the decimated subcube is displayed. Underneath these, the subcube bottom-left corner (BLC) and top-right corner (TRC) co-ordinates are displayed, and also the pixel increment (INC).

As you change the slider parameters, the memory requirement changes, as do the BLC, TRC and INC values, and these are reported to you. It is wise to not use more than about half the physical RAM (Random Access Memory) available in your machine.

As the file is loaded and decimated, a simple progress meter shows you how much longer you have to wait. Remember: you can always abort.

See Figure 2.2 for a screen snapshot.

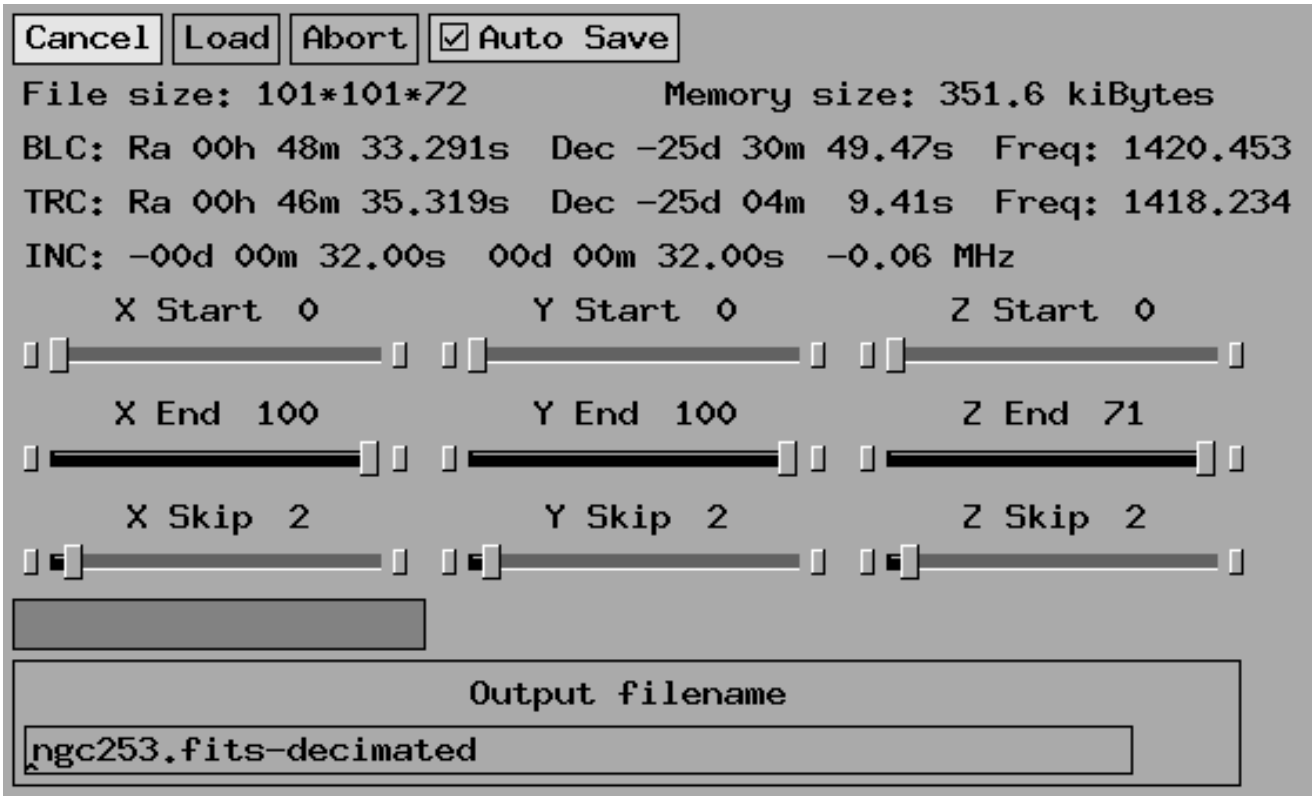

Figure 2.2: Screen snapshot of decimation control panel

2.2 Intensity Scaling

With many images that you display, you will want to change the colourtable used. The colourtable control appears after selecting the **PseudoColour (8 bit)** option in the **Intensity** menu in most tools.

The <**Cmapwinpopup**> widget provides the colourtable control interface. This widget controls a colourtable for a "PseudoColour" display canvas. Data values are used to index into a table of colours. This is typically used for "false colour" image display. The following controls are provided:

- **Close** close the window
- **Reverse** this reverses the colour table; the colours of the minimum and maximum are swapped
- **Invert** this inverts every colour (e.g. white becomes black, green becomes purple, blue becomes yellow and so on)
- **Save** this brings up a popup window which allows you to save a colourtable
- **Load** this brings up a file browser popup which allows you to load a previously saved colourtable

You can manipulate the colours in the colourtable by moving the round dot in the big square box (left mouse button). Usually moving the dot horizontally changes the apparent position of the colours, while moving it vertically changes the range of colours.

There are many "colour functions" available to set the colourmap, the default one being "Greyscale1". Some colour functions produce smoothly-varying colourtables, while others have sharp boundaries.

A final way of manipulating the colourtable is provided by the **Red Scale**, **Green Scale** and **Blue Scale** sliders. With these you can turn down the amount of each primary colour for all the colours in the colourtable.

See Figure 2.3 for a screen snapshot.

In many cases you will want to change the intensity scale in an image. This may be because the image has a very large dynamic range or has a few "outlier" values, and the control provided by the colourtable is inadequate. Most programmes have an **Intensity** menu with a selection named **IScale for Dataset** which will pop up a <**Dataclip**> widget.

This widget allows the user to specify a number of data "regions" (ranges). It it most often used to give control over a single region, so that an intensity range may be specified. The widget displays a histogram of the data values in the array, with the vertical axis being logarithmic. The following controls are provided:

- **Close** close the window
- **Blank** enable to blank data outside specified region (not all applications will show this toggle)

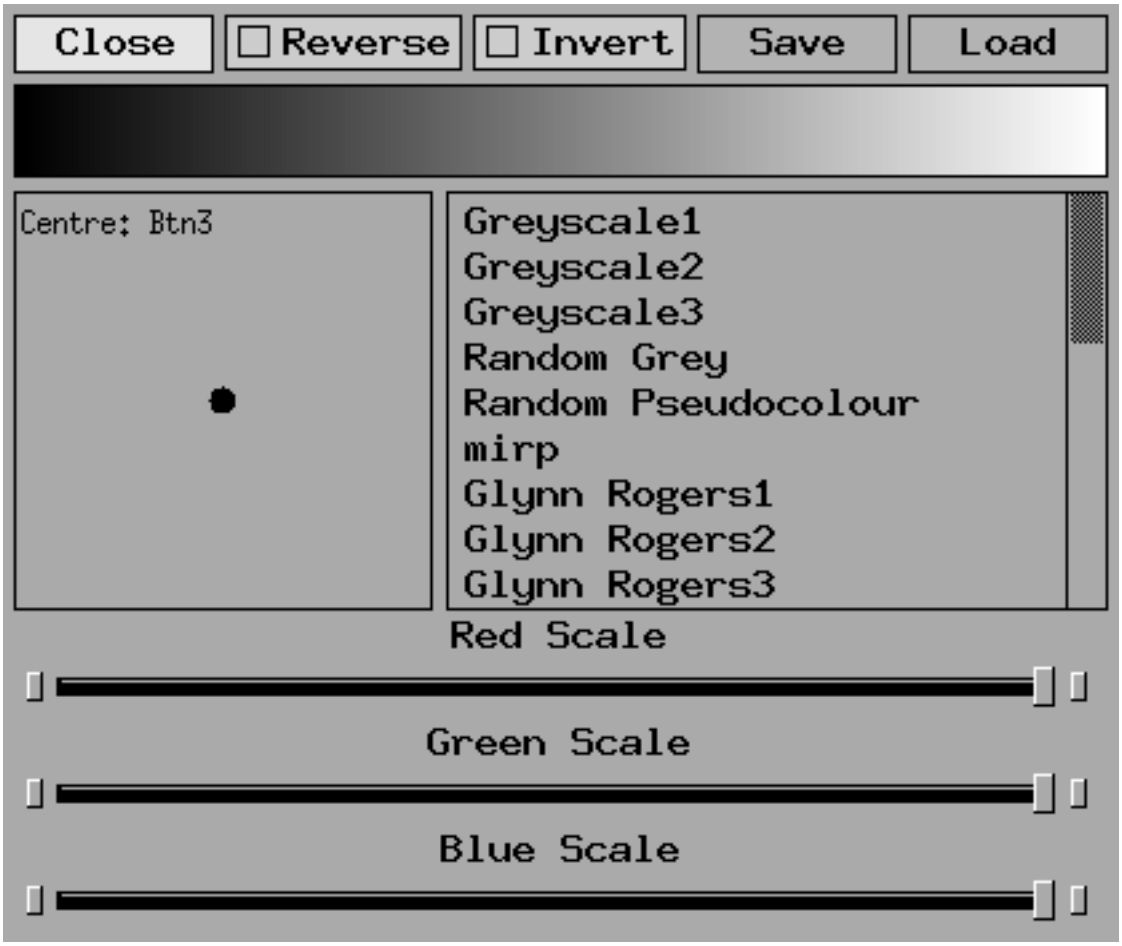

Figure 2.3: Screen snapshot of colourtable control panel

- **Zoom** zoom the histogram display to the range of the current region
- **Unzoom** unzoom the histogram display to the full range of the data
- **Apply** apply any changes made to the regions (most applications will then rescale and display their main image)
- **Auto Apply** enable this to automatically apply every change as they are made
- **Subset** enable this to use only a subset of the input data. This will speed up the histogram computation, and may avoid unwanted outlier values
- **Full Range** set the region to the full range of the data
- **95%** set the region to clip to the inner 95% of the data values (i.e. clip the lower 5% and upper 5% of the data values). This is very handy when your data contains "outliers" (small numbers of data values which lie far away from the real data)
- **98%** set the region to clip to the inner 98% of the data values
- **99%** set the region to clip to the inner 99% of the data values
- **99.5%** set the region to clip to the inner 99.5% of the data values
- **99.9%** set the region to clip to the inner 99.9% of the data values
- **99.99%** set the region to clip to the inner 99.99% of the data values
- **Save** save a new file which has been clipped to the specified range
- **Output Type Menu** change the output type of the array to save (not all applications will show this menu)
- **Filename string** the name of the file which will be saved. This may be changed prior to saving

If you click with the left mouse button in the histogram display, you define the lower boundary of a region. If no region exists, one is created. If you then click with the right mouse button, the upper boundary is defined. Once the lower and upper boundaries are defined, you will see two vertical lines depicting the boundaries, with two diagonal lines joining them. You will also see a circle, the height of which indicates the width of the region relative to the width of the entire histogram. If you click with the middle mouse button, the horizontal cursor position will control the midpoint of the region (i.e. change the lower and upper boundaries at the same time), and the vertical cursor position changes the width of the region. You can drag any of the three mouse buttons to see things continuously change.

If you have not yet defined a region, you may first click with the right button to define the upper boundary. The lower boundary is automatically set to the data minimum. This is a convenience facility when you want to set the lower

boundary to the data minimum and set the upper boundary manually. You can get the same effect by pressing **Full Range** first and then clicking with the right button.

See Figure 2.4 for a screen snapshot.

You can also change the intensity control policy by selecting the **Intensity Policy** option in the **Intensity** menu. This will pop up an <**IntensityPolicy**> widget.

This widget controls the intensity policy to be used for displaying image data on a drawing canvas. The following controls are provided:

- **Close** this will close the window
- **Apply** apply control changes and refresh the window
- **Auto Refresh** if enabled, each control change is automatically applied and the window refreshed
- **Auto Intensity Scale** if enabled, the intensity range of the (sub)image being viewed is mapped onto the colourmap, otherwise the intensity range of the entire image/movie is mapped onto the colourmap
- **Reset Intensity Upon File Load** if enabled, the intensity scale is reset when a new dataset is loaded (it is reset to the full data range). If disabled, the intensity scale is not changed on file load
- **Intensity Scale Menu** this menu allows you to change between different intensity scales. By default, the intensity scale is linear, but you can select a logarithmic intensity scale or a square-root intensity scale. If you are using a logarithmic intensity scale, you can set the **Log Cycles** value to control the visible dynamic range in powers of 10

See Figure 2.5 for a screen snapshot.

2.3 Sequences of Images

If you load a datacube into the <**kvis**> programme, you may wish to play it as a movie (sequence of images). A window to control this sequence is under **View->Animate**.

The <**AnimateControl**> widget provides the animation control interface. This widget controls movie loops. It allows the user to control the speed and direction of movies, as well as the ability to skip frames. The following controls are provided:

- **Close** close the window
- **Previous Frame** show the previous frame
- **Next Frame** show the next frame
- **Start Movie** start the movie

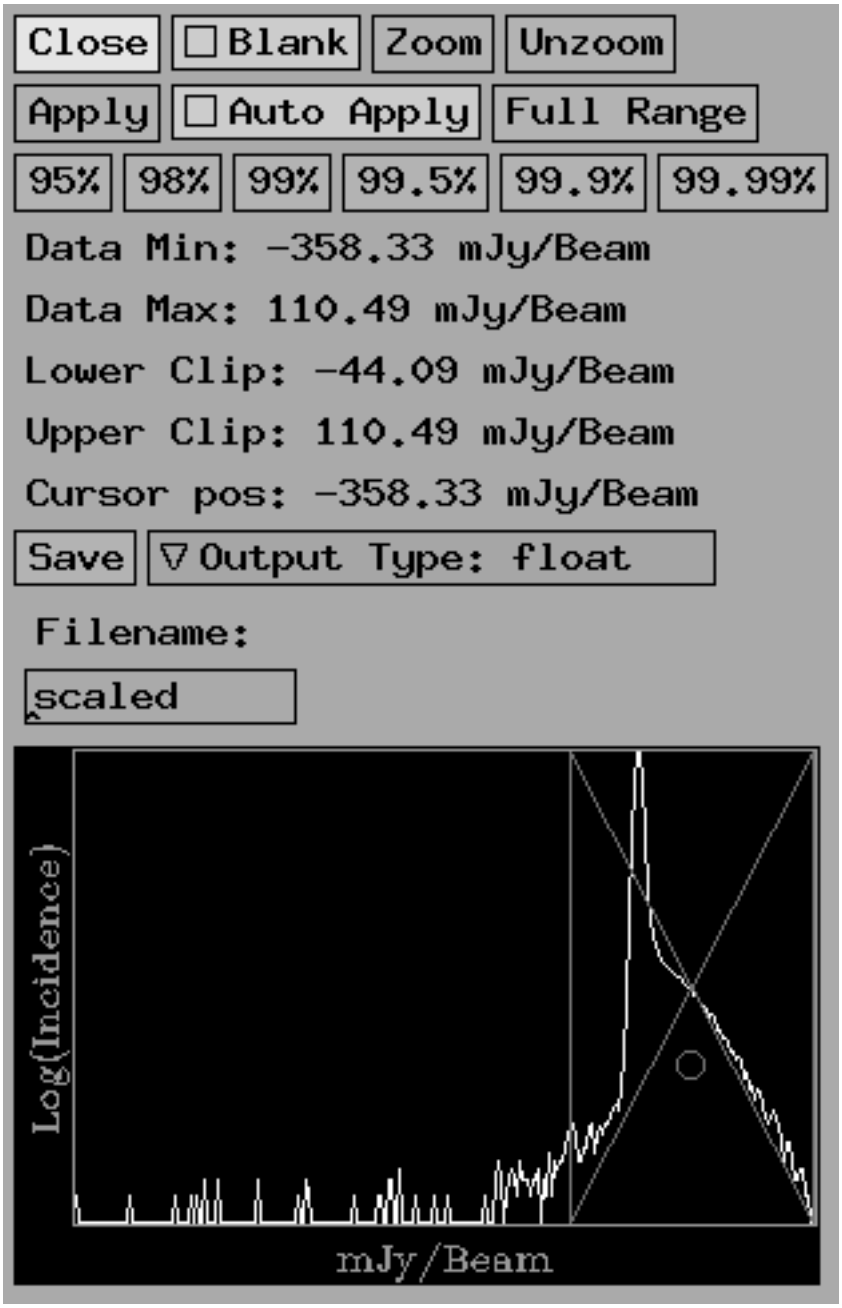

Figure 2.4: Screen snapshot of intensity zoom control panel

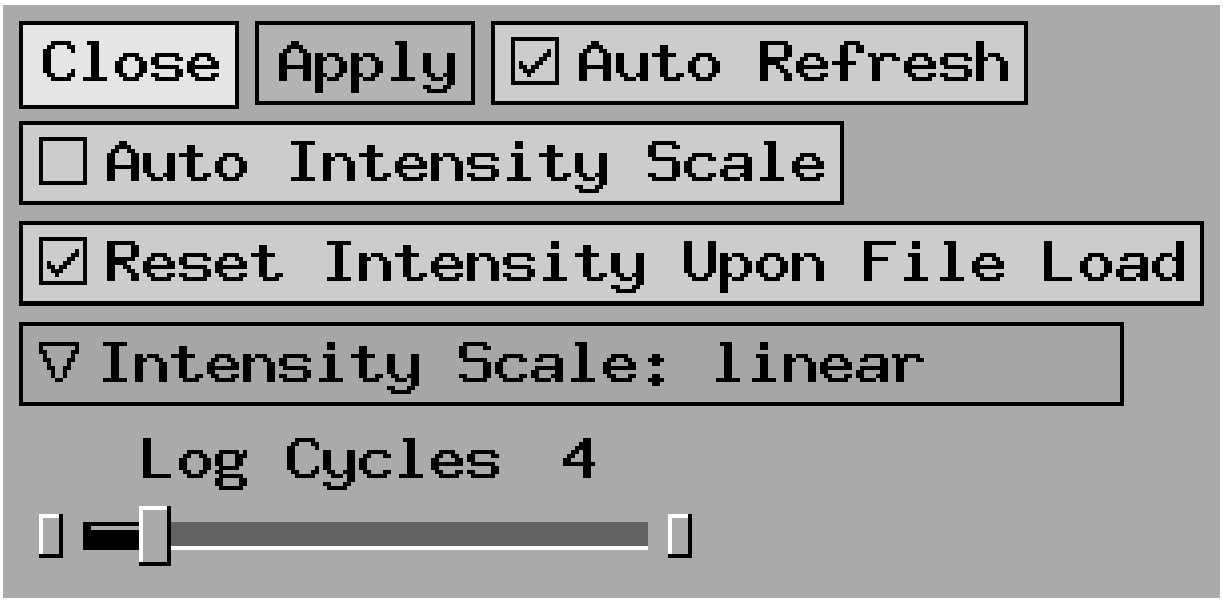

Figure 2.5: Screen snapshot of intensity policy control panel

- **Stop Movie** stop the movie
- **Sync** wait for each frame to be drawn before sending the next draw request. Normally, this is enabled. You can make movies run a little bit faster (not much) by disabling this, but you run the risk of developing a backlog of display requests in the display machinary. A large backlog will make the interactive response extremely poor
- **Spin Mode** you can select forward or backward spin, or alternatively spin forwards->backwards->forwards->...
- **Anti Flicker** remove the flicker observed when overlays (i.e. contours and/or annotations) and/or axis labels are shown. This is normally disabled because it slows down movie playing and interactive display updates. Also, flicker is only noticeable under some conditions (complex overlays and/or slow display hardware)
- **Frame Interval** the time interval between between frame draws. A low interval will give fast movies
- **Skip Factor** allows you to skip frames. A skip factor of 2 would skip every second frame
- **Goto Frame** jump to the frame number shown in the **Current Frame** control
- **Current Frame** the current frame number
- **Starting Frame** the starting frame number in the movie loop

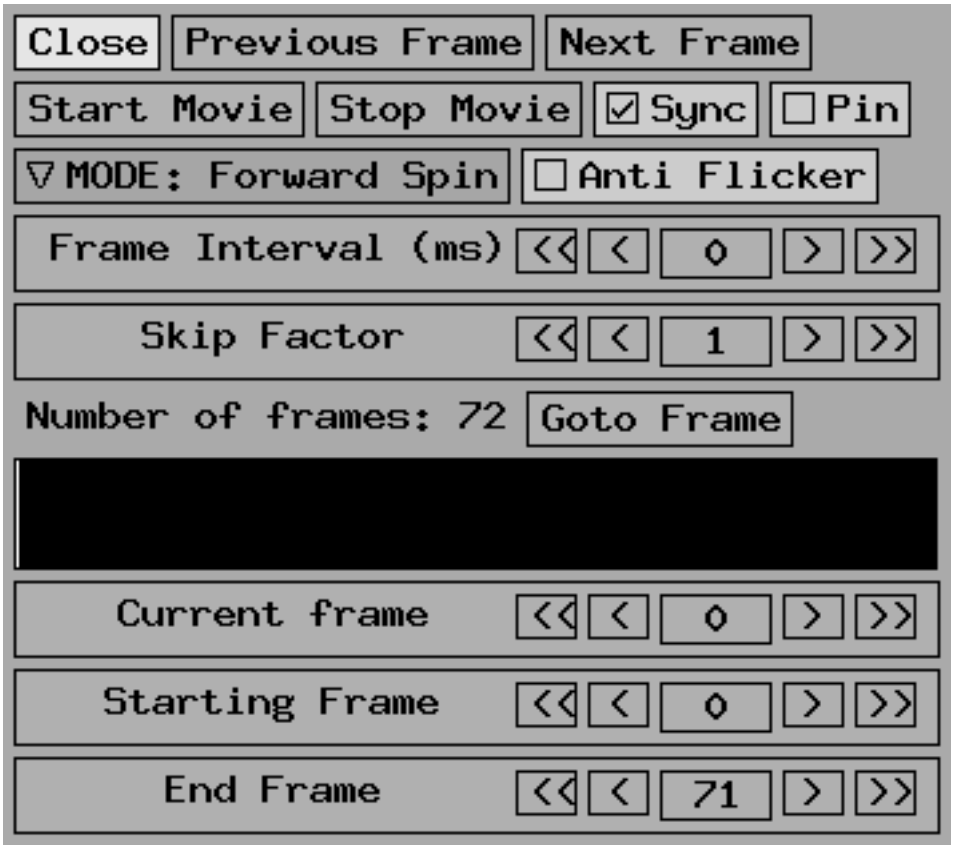

Figure 2.6: Screen snapshot of movie control panel

- **End Frame** the end frame number in the movie loop

You can also step through the cube by clicking (left) for the previous frame or click (right) for the next frame in the black rectangle which appears below the **Goto Frame** button. In addition, you can click (middle) and drag in this black rectangle, allowing you to dynamically slide through the movie.

See Figure 2.6 for a screen snapshot.

2.4 Hardcopies

To make a PostPcript hardcopy of the display, click on the menu button **Export** in the visualisation tools and select **PostScript**. This will display the <**Postscript**> widget. This widget provides controls for generating PostScript hardcopy. It supports different orientations and trays and allows you to queue directly to a printer. The following controls are available:

- **close** this will close the window
- **save .ps** save a PostScript file

- **save .eps** save an Encapsulated PostScript file
- **print** queue directly to a printer
- **Tray Menu** you can choose between default, paper or transparency trays
- **Size Menu** you can choose between a variety of media sizes (A3, A4, A5, letter and so on). The "`A4/letter`" size will fit on both A4 and letter size media. If you select "`user defined`" then you can define the bounding box using the offset and size controls described below. The default is taken from the "`PAPER_SIZE`" environment variable
- **Keep Aspect** if enabled, screen pixels will always be drawn square in the PostScript output
- **Orientation Menu** you can choose between portrait and landscape orientation, or you can let the choice be *automatic* so that a best fit is made (e.g. a display window that is much wider than it is high would be displayed in landscape mode on A4 size media)
- **hoffset** controls the horizontal offset of the output
- **voffset** controls the vertical offset of the output
- **hsize** controls the horizontal size of the output
- **vsize** controls the vertical size of the output
- **Auto Increment** when enabled, existing PostScript files are not overwritten, rather, the outfile filename has an incrementing sequence number added to it
- **Lock Filename** when enabled, the application will never automatically change the suggested output filename
- **Output file** controls the basename of the output file
- **Printer queue** controls which printer queue to print to. The default is taken from the "`PRINTER`" environment variable

See Figure 2.7 for a screen snapshot.

2.5 Exporting Data

To save data you are looking at, click(left) and hold the **Export** button. This will show the <**ExportMenu**> widget.

This widget allows you to export the currently displayed data to some other format. The following options are available:

- **PostScript** generate hardcopy output of the contents of the drawing canvas. This will pop up the <**Postscript**> widget (section 2.4)

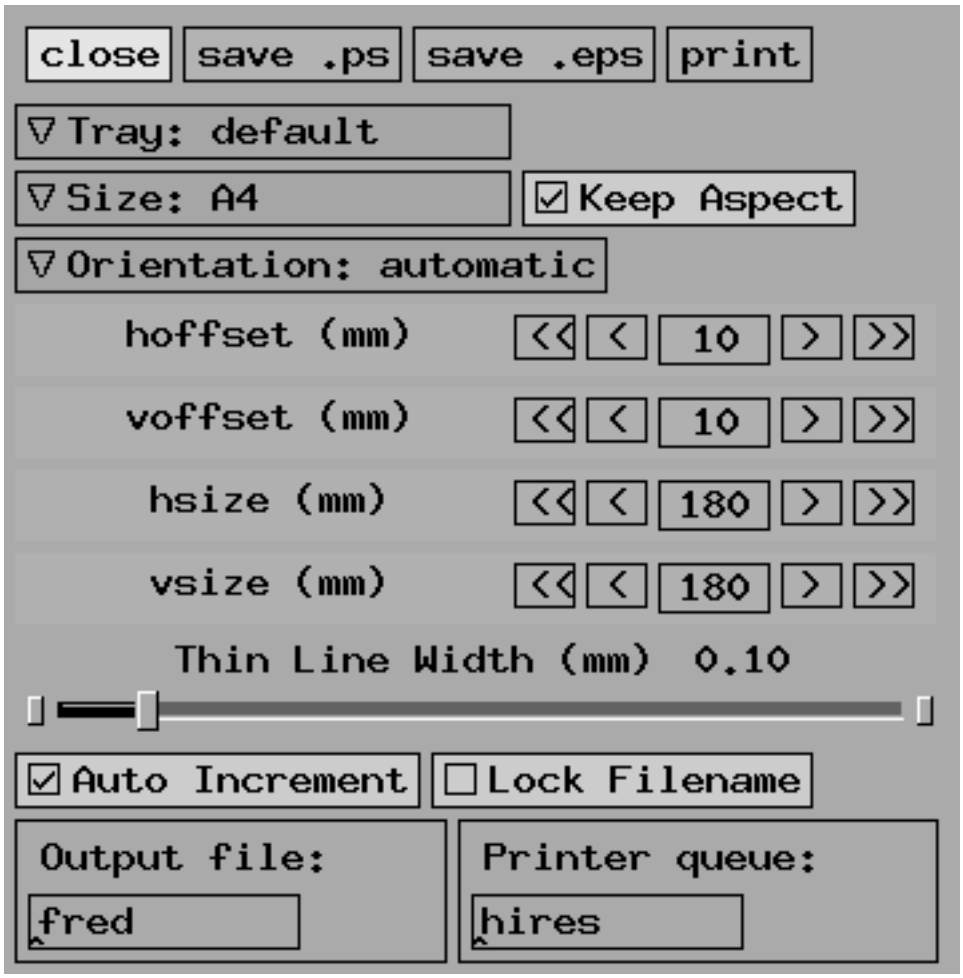

Figure 2.7: Screen snapshot of PostScript output control panel

- **SunRasterfile** export the currently viewed image to Sun Rasterfile format

- **PPM Data Image** export the currently viewed image data to PPM format. Non-image data (i.e. axis labels and annotations) are not exported. The output is at the resolution of the image data

- **PPM Data Movie** export the currently viewed cube data to PPM format, similar to `PPM Data Image`. One PPM file is written for each image along the unseen axis

- **PPM Data Window** export the contents of the window to PPM format. Non-image data (i.e. axis labels and annotations) are also exported. The output is at the resolution of the window, and yields higher quality than using a separate screen-capture tool

- **Karma (whole dataset)** export the currently viewed dataset to Karma format

- **FITS (whole dataset)** export the currently viewed dataset to FITS format

- **Miriad (whole dataset)** export the currently viewed dataset to Miriad Image format

- **GIPSY (whole dataset)** export the currently viewed dataset to GIPSY format

- **Karma (subset)** export the currently viewed (sub)image to Karma format. If the subimage being viewed is a cube, the subcube will be saved

- **FITS (subset)** export the currently viewed (sub)image to FITS format. If the subimage being viewed is a cube, the subcube will be saved

- **Miriad Image (subset)** export the currently viewed (sub)image to Miriad Image format. If the subimage being viewed is a cube, the subcube will be saved

- **GIPSY (subset)** export the currently viewed (sub)image to GIPSY format. If the subimage being viewed is a cube, the subcube will be saved

2.6 Zooming

Most tools provide a generic 2-dimensional zooming interface. This interface is provided by the <**ImageDisplay**> widget (section 2.15).

2.7 Resizing/autoscaling

To resize a display window, grab the bottom right corner of it and drag to the size that you want. The image is updated automatically. Most programs have a menu **Zoom** which has a selection **Zoom Policy** which will show the <**ZoomPolicy**> widget which gives you control over how zooming is done.

This widget controls the policy to be used for displaying image data on a drawing canvas. The following controls are provided:

- **Close** this will close the window
- **Apply** apply control changes and refresh the window
- **Auto Refresh** if enabled, each control change is automatically applied and the window refreshed
- **Fix Aspect** if enabled, the image will be expanded/shrunk with the same factor in both dimensions to fill the window. If disabled, the factors are not necessarily the same (i.e. the pixels will not be square, but the image will fill the window better)
- **Allow Truncation** if enabled, and the image is too big for the window and integer zooming is enforced, a few rows or columns of data may be discarded to allow the image to be shrunk
- **Integer X Zoom** if enabled, force zooming in the horizontal dimension do be done in integer multiples. Turning this off allows you to fill the window, but can introduce artefacts
- **Integer Y Zoom** if enabled, force zooming in the vertical dimension do be done in integer multiples. Turning this off allows you to fill the window, but can introduce artefacts
- **Zoom In on Middle Mouse Click** if enabled, clicking the middle mouse button will zoom in 2x using the mouse click position as the new centre
- **Zoom Out on Right Mouse Click** if enabled, clicking the right mouse button will zoom out 2x

See Figure 2.8 for a screen snapshot.

2.8 Axis Labels

All the standard visualisation tools now support axis labelling. Most programs have an **Axis Labels** button which will show the <**DressingControl**> widget which gives you control over how labelling is done.

This widget controls the way axis labels are drawn onto a canvas. The following controls are available:

- **Close** this will close the window

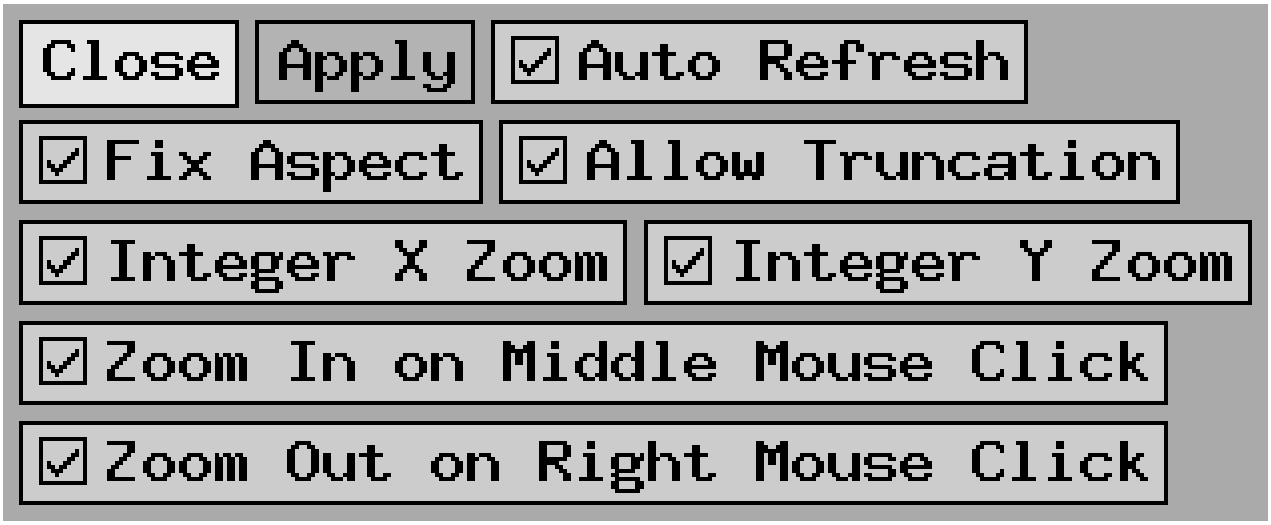

Figure 2.8: Screen snapshot of zoom policy control panel

- **Enable** if enabled, the axis labels are displayed
- **Apply** apply control changes and refresh the window
- **Auto Refresh** if enabled, each control change is automatically applied and the window refreshed
- **Show Labels** if enabled, the axis names are displayed
- **Show Scale** if enabled, the scale values corresponding to major tick marks are displayed
- **Show Top Tick Marks** if enabled, the tick marks at the top of the window are displayed
- **Show Bottom Tick Marks** if enabled, the tick marks at the bottom of the window are displayed
- **Show Left Tick Marks** if enabled, the tick marks at the left of the window are displayed
- **Show Right Tick Marks** if enabled, the tick marks at the right of the window are displayed
- **Internal Ticks** if enabled (the default), the tick marks are placed inside the bounding box, otherwise they are placed outside
- **Grid Lines** if enabled, the co-ordinate grid is shown
- **Screen Colours** pressing this will set the background colour of the window to black and the colour of the axis labels to green. If a colourmap is associated with the window, it's **Reverse** option is turned off

- **Paper Colours** pressing this will set the background colour of the window to white and the colour of the axis labels to black. If a colourmap is associated with the window, it's **Reverse** option is turned on

- **Change Colourmap** if enabled, the **Reverse** option for the window colourmap <**Cmapwinpopup**> widget (section 2.2) is set when you press either **Screen Colours** or **Paper Colours**

- **Show Colourbar** if enabled, a colourbar is displayed. This shows the mapping between data value and colour. Note that currently only linear intensity scales are supported

- **Font scale** this slider allows you to enlarge or shrink the font scale for the labels

A further set of controls is provided under the red line (most of the time you will not need to use these):

- **Background Colour** the background colour of the window. You need to press the **Apply** button for your changes to take effect

- **Label Colour** the colour the labels are drawn in. You need to press the **Apply** button for your changes to take effect

- **Grid Colour** the colour the grid is drawn in

Note that the axis labelling currently does not handle co-ordinate systems which have rotations close to 90 degrees or 270 degrees.

See Figure 2.9 for a screen snapshot.

2.9 Magnifying Glass

Many programmes have a **Zoom** menu with a **Magnifier** option which, when selected, will show a magnifying-glass window. As you move the mouse cursor over the main image window, a zoomed-in subsection around the cursor is displayed in the magnifying glass. This view is zoomed in four times the scale of the input image. If the image in the main window is zoomed in ten times, the view in the magnifying glass will actually be smaller!

This widget is a subclass of the <**SlaveImageDisplayPopup**> widget.

2.10 Panner

Many programmes have a **Zoom** menu with a **Panner** option which, when selected, will show a panner control window. By pressing the left mouse button in this window, the image in the main window is panned.

This widget is a subclass of the <**SlaveImageDisplayPopup**> widget.

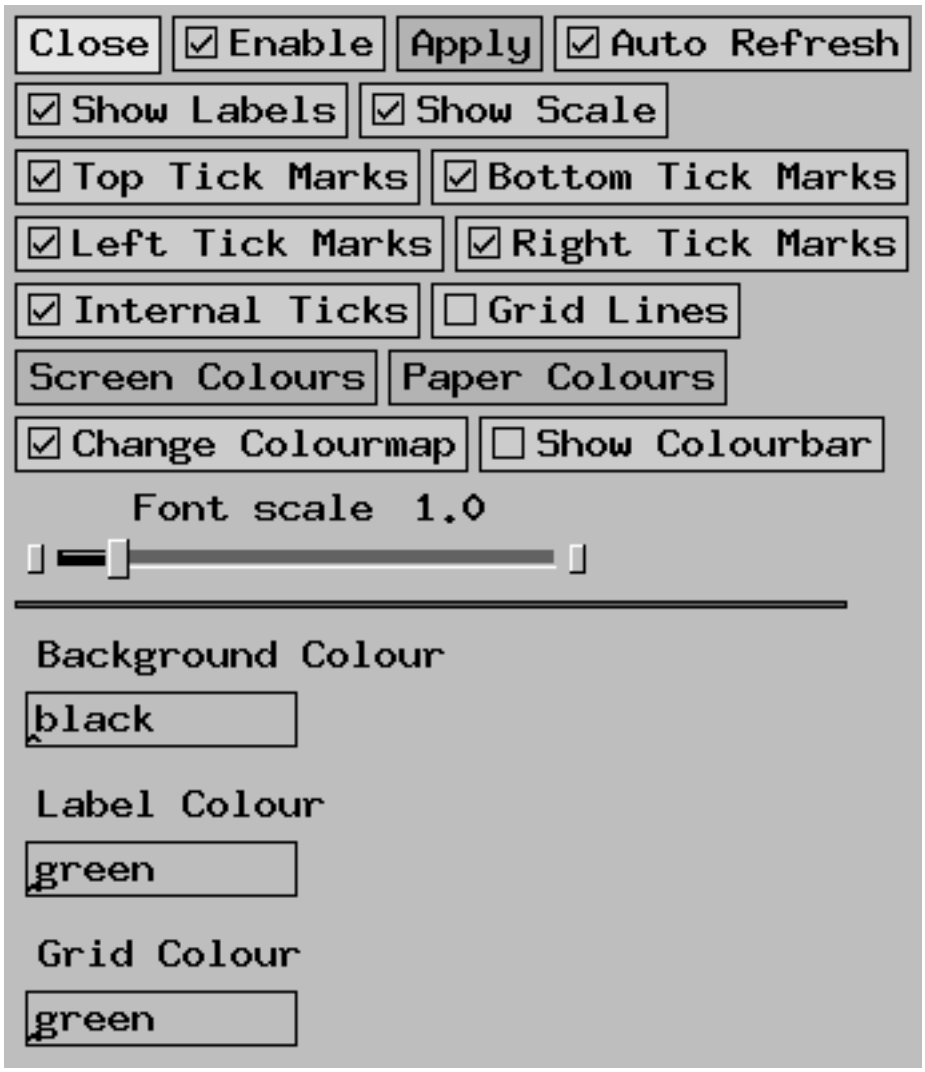

Figure 2.9: Screen snapshot of axis labelling control panel

2.11 Spectrum Profiles

Many programmes allow you to point at an image and obtain the corresponding profile along the third axis (usually a spectrum profile). Some programmes provide a **Profile** button which you need to press first before you will see the profile, while some others will automatically display the profile window when you have loaded a cube of floating-point data. You don't need to press any buttons to update the profile display, simply point the mouse at the desired spot. The ability to display spectrum profiles is provided by the <**TracePopup**> widget. This widget will display a line plot of a single trace of one-dimensional data. It is fast, allowing dynamic display of data, and provides zooming and smoothing controls, as well as having the ability to display axis labels. The controls available are:

- **Close** this will close the window
- **Auto Zoom** if enabled, the vertical scale is automatically zoomed to the range of the data in the profile
- **Unzoom** this will unzoom the window
- **Smooth** menu, to select what kind of smoothing to apply to the data, prior to display. Both hanning and uniform smoothing kernels are available, with a range of sizes
- **Print** this will pop up the <**Postscript**> widget (section 2.4)
- **Overlay** this allows you to popup various control panels to control overlay display, including axis labelling (section 2.8)
- **Save Data** this will pop up a dialog widget which allows you to save the data to a file. The data is saved in a simple ASCII format with a few comment lines followed by two columns of numbers. The first column is for the horizontal axis and the second column is for the vertical axis
- **Style** menu, to select the drawing style for the data points. The following choices are available:
  - **join** a straight line is drawn from each point to the next
  - **hist** a centred column is drawn for each point (aka. "histogram" style)
  - **cross** a small crosshair is drawn for each point

Zooming in TracePopup Widget

To zoom a profile, click (left) and drag the mouse (still keeping the left button down) across the window. You will see a "rubber-banded" box stretching between the first point and the current mouse cursor position. When you release the button the section of the profile within the box will be zoomed. To unzoom,

choose the **Unzoom** option in the **Zoom** menu, or press the "u" key while the cursor is in the profile window. This will show the entire image.

See Figure 2.10 for a screen snapshot.

2.12 Image Statistics

Many programmes have a facility to compute and display the statistics of the current (sub)image being viewed. This interface is provided by the <**ImageDisplay**> widget (section 2.15).

2.13 Viewing Header Information

Many programmes allow you to inspect the header of the dataset you are currently viewing as an image. This interface is provided by the <**ImageDisplay**> widget (section 2.15).

2.14 Freezing Events

All drawing operations in Karma tools are performed on a "canvas", which is usually a big plain black or white rectangle.

The canvas allows you to freeze "mouse move" operations (i.e. things which happen in response to your moving the mouse over the window, but not while pressing any mouse buttons or keyboard keys). This is useful when you have pointed at a certain place, and want to maintain a position or profile display while you move the mouse to some other window (perhaps to print the window). All you need to do is press the spacebar, and any further mouse moves are ignored. If you press the spacebar again the mouse moves once again do something. So that you don't forget that mouse moves are disabled, you will hear a quiet beep every time you move the mouse into the window.

Similar to above, except that when pressing any mouse buttons or keyboard keys the freeze position is used, rather than the real position of the mouse.

2.15 Common Image Display Features

Many of the visualisation tools have one or two very similar main windows. This is because they are all derived from the <**ImageDisplay**> widget, which provides most of the common functionality. Most of the tools which use this widget provide the following controls:

- **Set1** sometimes labelled as **Files**, or **Cubes** this will popup up the <**Filepopup**> widget (section 2.1)
- **Intensity** menu, this will popup widgets to control the colourmap (<**Cmapwinpopup**>), intensity policy (<**IntensityPolicy**>) and intensity scaling (section 2.2)

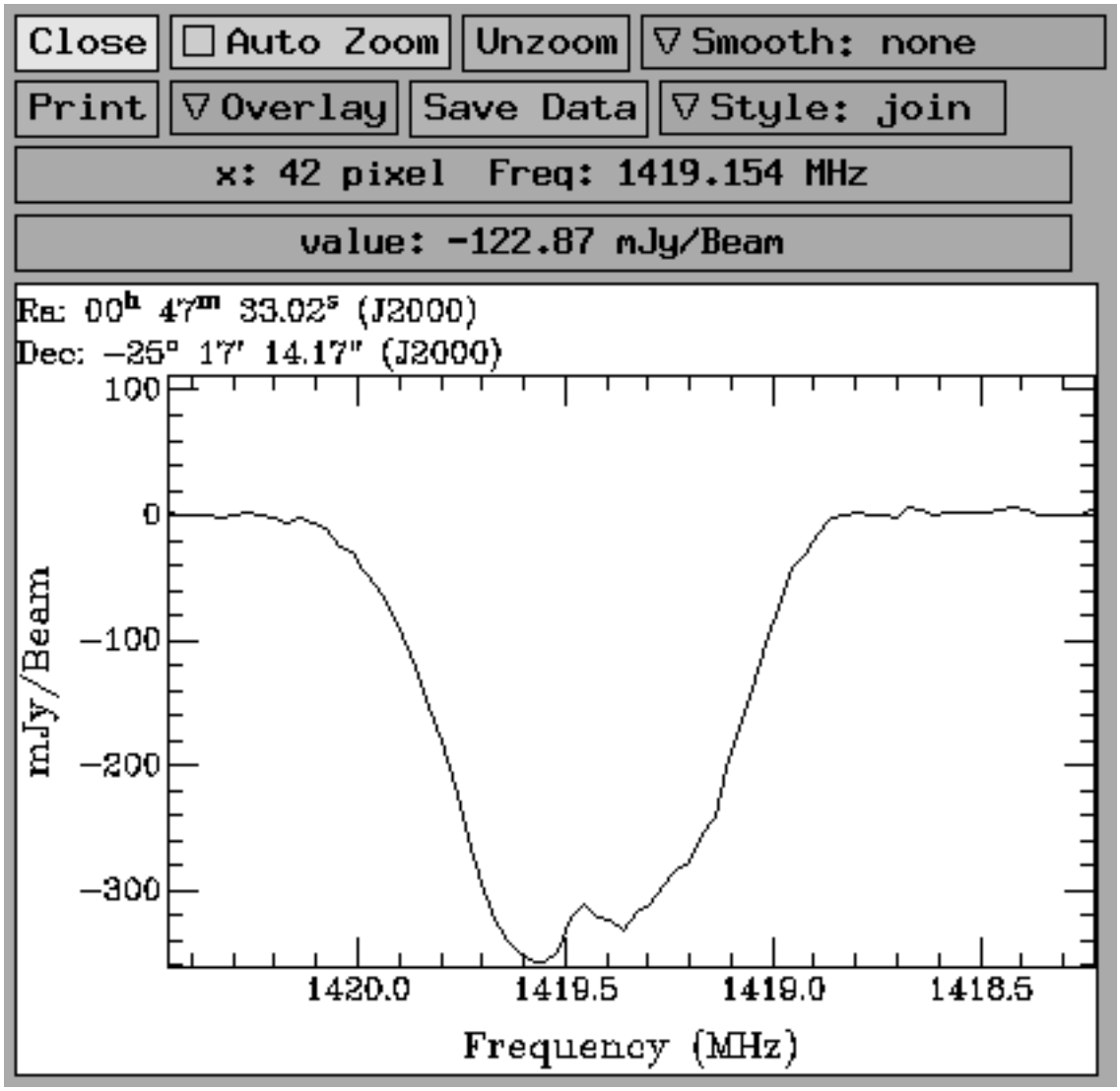

Figure 2.10: Screen snapshot of profile display window

- **Zoom** menu, this allows you to popup a magnifier or a panner, zoom the image in or out 2x or 4x, unzoom, or popup the <**ZoomPolicy**> widget (section 2.7)

- **Overlays** menu, this allows you to popup various control panels to control overlay display. The following choices are available:

  - **Axis Labels** this will popup controls for the axis labelling (section 2.8)
  - **Load Annotations** this will bring up a file browser to load a file of ASCII overlay commands. The file format is described in an appendix to the User Manual
  - **Editor** this will popup the <**OverlayEditorControl**> widget (section 2.19) which allows you to interactively draw overlays
  - **Remove Last (connection)** this will remove the last overlay object sent via a network connection
  - **Remove All (connection)** this will remove all overlay objects sent via a network connection
  - **Contours** this will pop up a file browser which allows you to load an image and display it as contours. You can load an unlimited number of images this way

- **Export** menu, you can export data in a variety of formats. See section 2.5

- **View** this will popup a control panel used for displaying two datasets (section 4). This is only available in tools like <**kvis**> and <**MultibeamView**>

- **Edit** this will popup the <**ImageEditorControl**> widget (section 2.21) which allows you to interactively edit image data

- **Quit** this will quit the application

- **Set2** sometimes labelled as **Images** this will popup up the <**Filepopup**> widget (section 2.1)

You will also see a few lines of display under the control buttons, which gives information about the data under the current cursor position.

See Figure 2.11 for a screen snapshot.

The Magnifier

Wherever you move the mouse over the main image, you will see a magnified portion of that part of the image (the magnification is 4x the data resolution). You can resize the magnifier window.

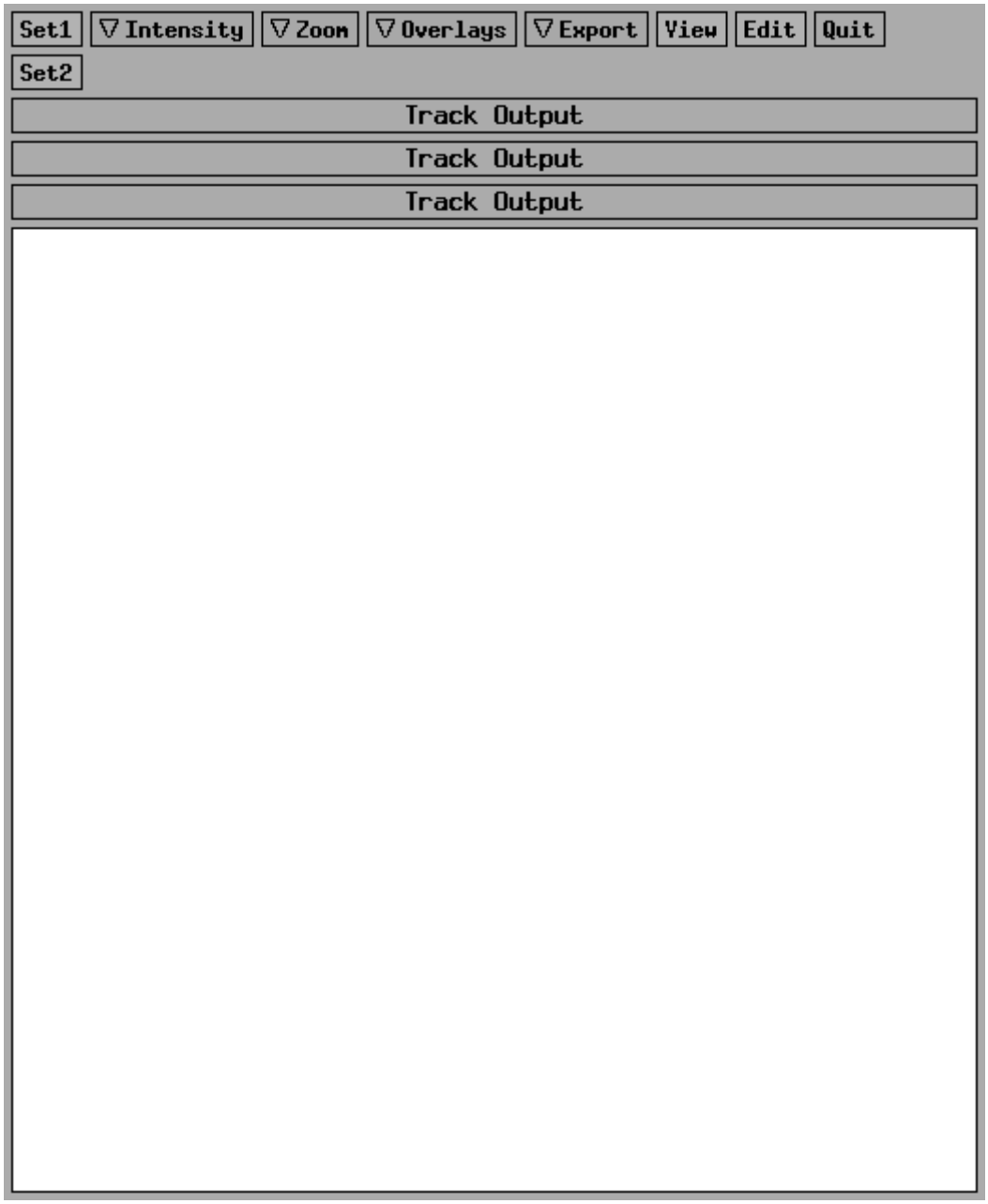

Figure 2.11: Screen snapshot of generic main window

The Panner

Simply click with the left mouse button in the panner window and the main window will pan to the desired location. The panner window will display a green rectangle showing you where you have panned to, relative to the entire image. You can resize the panner window. When you select the panner from the **Intensity** menu, the panner is popped up and the main window is set to panning mode.

Zooming in the ImageDisplay Widget

To zoom an image, click (left) and drag the mouse (still keeping the left button down) across the window. You will see a "rubber-banded" box stretching between the first point and the current mouse cursor position. When you release the button the section of the image within the box will be zoomed. To unzoom, choose the **Unzoom** option in the **Zoom** menu, or press the "u" key while the cursor is in the image window. This will show the entire image.

You can also press the "i" key (zoom in) and the image is zoomed 2x, with the new centre of the image being the position of the mouse when the "i" key was pressed. You can zoom in as many times as you like. To unzoom 2x, press the "o" key (zoom out). When unzooming it doesn't matter where the mouse is.

If you press and release the left mouse button without moving the mouse in between, the image will be panned so that what is under the mouse cursor is moved to the centre of the image. An extra facility in <**kvis**> and <**MultibeamView**> is the ability to zoom in and out using the middle and right mouse buttons, respectively, in a similar fashion to using the "i" and "o" keys.

Note that these zoom controls are not available when you place the main window in panning mode.

Another key you can press is "r" which simply refreshes the window, without changing the zoom area.

Image Statistics in the ImageDisplay Widget

Many programmes have a facility to compute and display the statistics of the current (sub)image being viewed. Simply press the "s" key in the display window and you will see a summary of the image statistics in the terminal window. This summary includes the number of (non-blank) values, the standard deviation, the mean, minimum, maximum and sum value. Alternatively, you can click (left) and drag the mouse to a new position (still holding the mouse down!) and press the "s" key. The statistics are computed over the boxed area. After you press "s" you may release the mouse button. This feature saves you from having to zoom in and unzoom every time you wish to compute statistics on a sub-image. Note that the old box remains on the image until something clears it (this is useful when you want to know which regions you have already computed statistics over).

The "v" key allows you to see the individual image pixel values, rather than their statistics. This works in a similar fashion to the "s" key. The "V" key is a variation which shows raw image pixel values. Note that if you press "v" or "V" without the mouse, all the pixels in the subimage being viewed would be printed. Since this can take a long time with a large number of pixels, a warning message is issued if you attempt this with many pixels. If you definately want to display a large number of pixels, press the control key at the same time.

Viewing Header Information in the ImageDisplay Widget

Many programmes allow you to inspect the header of the dataset you are currently viewing as an image. Simply press the "h" key and the header will be displayed in the terminal window. If you press the "H" key, you will get the header without the history.

Capturing Co-ordinate positions in the ImageDisplay Widget

Many programmes allow you to capture the current world co-ordinate position of the mouse, by pressing the "l" key in the image window. The output is set to the standard output, which you may redirect to a file prior to starting the programme.

2.16 Generating Moment Maps

Some of the programmes allow you to generate quick moment maps. This is usually available in tools where you need both a cube and an image of a similar section of sky. To save you the trouble of going into an astromonical reduction package to generate the moment maps, a menu is provided so that you can generate and view a moment map. This is usually the **Loaded Image Menu**, and has entries like **0th moment** and **1st moment**. Selecting either one of these will pop up the <**MomentGenerator**> widget. This widget allows the user to generate the 0th (total intensity) and 1st (velocity field) moment maps from a cube. The following controls are available:

- **Close** this will close the window
- **Apply Parameters** this will apply the parameters and compute the moment maps
- **1st Moment Algorithm Menu** this allows you to choose between a simple "weighted mean" algorithm or a more robust "median" algorithm
- **Lower Clip Level** values in the cube lower than this value are not used in the computation of the moments
- **Sum Clip Level** values in the computed 0th moment map lower than this value are not used in the computation of the 1st moment map
- **Start Channel** the first channel that will be used in the computation

Figure 2.12: Screen snapshot of moment map generator control panel

- **End Channel** the last channel that will be used in the computation

See Figure 2.12 for a screen snapshot.

2.17 Cursor Readout

Most data display windows have cursor readout facilities: simply move the mouse over the display canvas and you will see a few lines of numbers change above the canvas. In most cases positions are shown both in data pixel co-ordinates as well as real-world co-ordinates.

2.17.1 Warning for Astronomers

Most astronomical reduction packages count from 1, so the bottom-left corner of an image is at (1,1). This reflects the FORTRAN language from which these packages originated. The **Karma** software is written in the C language and reflects the C convention of counting from 0. This means that the bottom-left corner of an image is at (0,0) in **Karma**.

2.18 GUI Command-line Switches

Many of the Graphical User Interface tools have a number of command-line switches which may be used to modify their behaviour. These are listed below:

- **-private_cmap** This option will force the tool to use a private colourmap for its PseudoColour window. Otherwise, the tool trys to allocate colours from the default colourmap
- **-num_colours** This option specifies the number of colour cells that the tool will try to allocate for its PsuedoColour window upon startup. If less colours are available it allocates as many as possible (minimum 2)
- **-cmap_master host:port** This option will make the tool use the colourmap of another tool. The *host* portion of the second argument is the Internet hostname or address where another Karma tool is running. If this is "unix" or "localhost" the tool to connect to is running on the same machine. The *port* portion specifies which Karma programme to connect to. You may use the port number which is displayed in the other tools title bar, or you may use the name of the tool (such as "kvis"), and it will connect to the first "kvis" tool you started on that machine
- **-fullscreen** This option will make the image display window take up the entire screen. This is useful if you wish to make a video of the data. You will need to configure your window manager to cycle the window stacking order when a special key is pressed (e.g. the "Back" key with the Open Look window manager). Alternatively, pressing the right mouse button in the image window will place it underneath other windows. See also the appendix on making videos (appendix E).

In addition, the standard Xt command-line switches are supported. Read the manual page for **xterm** for more details. In particular, the following options are worth noting:

- **-display** control where the tool is displayed
- **-geometry** control the size and postion of the tool
- **-xrm** pass a resource string

2.18.1 Avoiding Colourmap Flashing

Standard workstations with 8bit "PseudoColour" displays only have a single colourmap active at one time. If you start two display tools, each wanting to use a large number of colours, then when you move the mouse between the windows you get that horrible colourmap flashing effect as the window manager updates the hardware colourmap. Also, since the colours used by one tool are likely to be quite different from those used in another tool, the image in one tool will look completely wrong when the mouse is in another tool. It is particularly annoying if colourmap flashing occurs when moving between an image window

and the desktop (i.e. normal terminal windows), because often you may want to read something in a terminal window while also looking at an image. There are a number of ways to get around this.

One of the most common problems is a WWW browser (like Netscape) or a display tool (like xv or ghostview) which is started before you start a display tool. You can try to limit the number of colours Netscape uses, but I haven't found that to be all that reliable. You can also use the **-install** command-line switch, which will force Netscape to use it's own colourmap, rather than stealing colours from the standard one. This will of course mean that as you move the mouse into the Netscape window the colours flash, but in practice this shouldn't matter so much because people tend to use Netscape less often than a visualisation tool.

Another more recent (and more insidious) problem is the "CDE" (the Common Desktop Environment), which steals not only too much screen space but also a large number of colours. Unfortunately, users don't get much choice about CDE, as the system administrator often blindly enables it when installing the operating system. If you start a display tool under CDE, the tool will usually have to create its own private colourmap in order to get enough colours. This would give you that colourmap flashing whenever you move between the image window and the desktop. You can try to get around this by using the **-num_colours** option to Karma tools (see above). Many tools try to allocate 200 colours, but you can use 100 instead and it is usually still workable. In the long run, though, it is probably better to get rid of CDE, or possibly configure it so it only takes a few colours (good luck).

Another problem can be when you use more than one Karma visualisation tool at the same time. The second tool is likely to create its own colourmap because the standard (desktop) colourmap has few colours left. You can either reduce the number of colours required by each tool, or you can share colourmaps using the **-cmap_master host:port** command-line switch (see above).

Finally, you may be wanting to use a Karma visualisation tool as well as some other display tool. To avoid colourmap flashing, you will need to reduce the number of colours each tool uses. You will need to read the manual for the other tool to see what option controls the colourmap size.

Summary

Examples of things you can do:

- `unix% netscape -install`
- If running within the CDE, or other display tools (eg SAOimage),
  `unix% kvis -num_colours 100`
  and reducing the number of colours in the other tools:
  `unix% saoimage -palette 100`
- When running multiple Karma tools at once, even on different machines, you can do this:
  `junior% kvis`

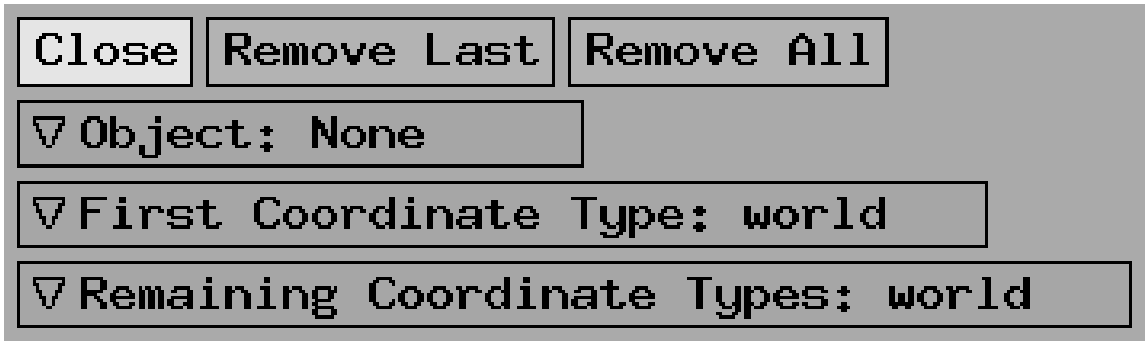

Figure 2.13: Screen snapshot of overlay editor control panel

```
Port allocated:  17545
jumbo% xray -cmap_master junior:17545
```

2.19 Drawing Overlays

Many tools provide a facility to draw overlays. This is provided by the <**OverlayEditorControl**> widget.

This widget allows you to interactively draw overlays on top of an image. It uses the <**overlay**> package which maintains a list of geometric figures which should be drawn.

The following controls are provided:

- **Close** this will close the window
- **Remove Last** will remove the last overlay that you drew from the list
- **Remove All** will remove all overlays from the list
- **Object Menu** allows you to choose between the different types of object that you can draw
- **First Coordinate Type Menu** sets the co-ordinate type of the first defining point for a geometric object
- **Remaining Coordinate Types Menu** sets the co-ordinate types of the remaining points which define a geometric object

See Figure 2.13 for a screen snapshot.

This widget uses the middle mouse button for its drawing functions. Some tools may already be using this button for other purposes, in which case the tool should provide a control which allows you to disable the normal use of the middle mouse button. For instance, the <**kvis**> programme provides such a control through the <**ZoomPolicy**> widget (section 2.7).

Drawing Rectangles

You click down the mouse button to define one corner of the rectangle and move the mouse (keeping the button down) until you have a rectangle to your liking. Once you are happy, release the button and the rectangle will be added to the overlay list.

Drawing Ellipses

Click the mouse button to define the centre of the ellipse and move the mouse (keeping the button down) to change the size and shape of the ellipse. If you press and release the 'r' key (still with the button down), this will toggle *rotation* mode, so that now as you move the mouse the ellipse will rotate. To leave *rotate* mode, just press the 'r' key again, and you will be back to changing the size and shape of the ellipse. Once you have an ellipse you are happy with, release the mouse button and it will be added to the overlay list.

Drawing Polygons

Click and release the mouse button to define the first and subsequent vertices of the polygon. To close (terminate) the polygon, click and release the button twice without moving the mouse. Once you have closed the polygon it is added to the overlay list.

2.20 Loading Overlays

Many tools provide a facility to load overlays (annotations) from an ASCII file. Please see appendix D for details.

2.21 Editing Images

Many tools provide a facility to edit image data. This is provided by the <**ImageEditorControl**> widget.

This widget allows you to interactively edit images. It uses the <**iedit**> package which maintains a list of geometric figures which can be drawn and then applied to data.

The following controls are provided:

- **Close** this will close the window
- **Undo Last** will undo the last edit object that you drew from the list
- **Undo All** will undo all edit objects from the list
- **Apply** apply the edit objects to the data (after this you can no longer undo previous edits)
- **Brush Width** the width of the paint brush in pixels

- **Paint Value** the data value to paint with
- **Minimum** set the data value to paint with to the current minimum (clip) value for the display canvas
- **Middle** set the data value to paint with to halfway between the current minimum and maximum (clip) values for the display canvas
- **Maximum** set the data value to paint with to the current maximum (clip) value for the display canvas

This widget uses the middle mouse button for its drawing functions. Some tools may already be using this button for other purposes, in which case the tool should provide a control which allows you to disable the normal use of the middle mouse button. For instance, the <**kvis**> programme provides such a control through the <**ZoomPolicy**> widget (section 2.7).

The drawing interface is the same as for the <**OverlayEditorControl**> widget (section 2.19).

Chapter 3

Viewing and Comparing multiple Images and Cubes

The <**kvis**> programme allows you to view multiple datasets.

This tool allows you to compare different datasets in a very easy fashion. You can display any dataset as an image, overlay contours of datasets, show profiles along any axis, and much more. The programme uses the <**ImageDisplay**> widget (section 2.15) to provide the usual interface.

3.1 Overview

The tool is centred around the concept of a *blink-state*, which defines how you want to display datasets (i.e. which dataset is shown as the image, which datasets are shown as contours, how to control movies and profiles and so on). A blink-state can control the display settings of all datasets that have been loaded into the application.

Multiple blink-states may be created, each with a different configuration of what datasets are to be shown. By pressing a key, the display is switched from one blink-state to another.

Blink-states are created and managed using the <**DataBrowser**> widget.

3.2 View Control

Pressing the **View** button will pop up the <**ViewDatasets**> widget.

This widget provides controls for a display window to manage the display of multiple datasets. Most of the controls for data display management are in the <**DataBrowser**> widget.

The controls provided are:

- **Close** this will close the window
- **Browsers** this will pop up the data browser window(s) for this display window

- **Movie** this will pop up the <**AnimateControl**> widget (section 2.3), which will allow you to play movies and step through frames. In addition, the following key bindings are defined for the main display window:
  - **PgDn** go to previous frame
  - **PgUp** go to next frame
  - **Home** go to start frame
  - **End** go to end frame

  On some keyboards, `PgDn` is marked as `Next` and `PgUp` is marked as `Prev`.

- **Profile Mode Menu** this allows you to pop up the <**TracePopup**> widget (section 2.11) and choose the profile mode. The following modes are available:
  - **None** will pop down the <**TracePopup**> widget
  - **Line** will show a single profile in the selected direction
  - **Box Sum** lets you draw a box over the image window (using the middle mouse button, in a similar way as using the left mouse button to zoom) and then the sum of all the profiles in that region is displayed. Box mode only works for 3-dimensional datasets, and will always display a profile along the unseen dimension
  - **Box Average** is like the **box sum** mode except that the average of all profiles is displayed rather than the sum
  - **Radial** will azimuthally average the image plane data and produce a plot of average value versus radius. A gaussian is then fitted to the radial profile and over-plotted in red and the FWHM (full width half max) of the gaussian is displayed over the profile. You must click the middle mouse button to define the centre of the object of interest, then, keeping the mouse button down, drag out to define the maximum radius of the profile and release the button to display. The algorithm automatically adjusts the centre position using a centroiding operation, so you don't have to be too careful about picking the centre of your object. More information on the **centroidingalgorithm** (`http://www.atnf.csiro.au/karma/user-manual/centroidingalgorithm.html#pubce` can be found in the Karma User Manual.

    The programme should also work with a photographic negative. The centroiding algorithm is automatically modified to look for a trough rather than a peak
  - **Boxed Horiz** will produce a horizontal profile which is averaged vertically. The box is drawn the same way as for the **box sum** mode. The horizontal centroid of the profile is computed and a gaussian is fitted and displayed

  - **Boxed Vert** will produce a vertical profile which is averaged horizontally. The box is drawn the same way as for the **box sum** mode. The vertical centroid of the profile is computed and a gaussian is fitted and displayed

- **Slice Direction Menu** this allows you to choose how you want to slice your cube. You can view XY, XZ or ZY planes
- **Profile Axis Menu** this allows you to choose along which axis you want the profile to be displayed (valid only for line profile mode)
- **Freeze Displayed Intensity Range** if unset (the default), the intensity range specified for a dataset is used when that dataset is displayed as an image. If set, switching from one dataset to the next will not change the displayed intensity range (however, changing the intensity range for the image dataset will change the displayed intensity range)
- **Track Cursor** if set, the corresponding image display window will "listen" for cursor moves which occur in other image display windows, and will draw a red circle at the same world co-ordinate position
- **Show Frame in Line Profile** if this is set the current frame displayed in the main image window is shown as a vertical red line in the profile window. This marker is only drawn if the profile axis is the unseen axis
- **Auto Title** if enabled, the axis labelling title is automatically generated. Whenever this is disabled, the string specified by **Axis Labelling Title** is used instead
- **Show Beam** if enabled, the dataset headers are searched for "BPA", "BMAJ" and "BMIN" FITS-style keywords. If these keywords are present, a representation of the telescope beam is overlaid on the image
- **Show Beam Name** if enabled, the name of the dataset is placed near the beam representation
- **Beam Xpos** controls the horizontal position of the the beam
- **Beam Ypos** controls the vertical position of the the beam

In addition, if you click the left mouse button in the display window without moving it in between the press and release, the image will pan across. If you click the middle mouse button, the image will zoom in 2x (the new centre of the image will be the place where you clicked). Click the right mouse button to zoom out 2x. If the profile mode is "box" then you can't use the middle mouse button to zoom.

Pressing the `c` key in the display window will compute and display a scatter plot of intensity values in the image dataset versus intensity values in the "alternate" dataset. This is useful for seeing if there is a correlation between the values in two images. You can use the left mouse button to define a sub-image (similarly to zooming in) from which the scatter plot is computed.

See Figure 3.1 for a screen snapshot.

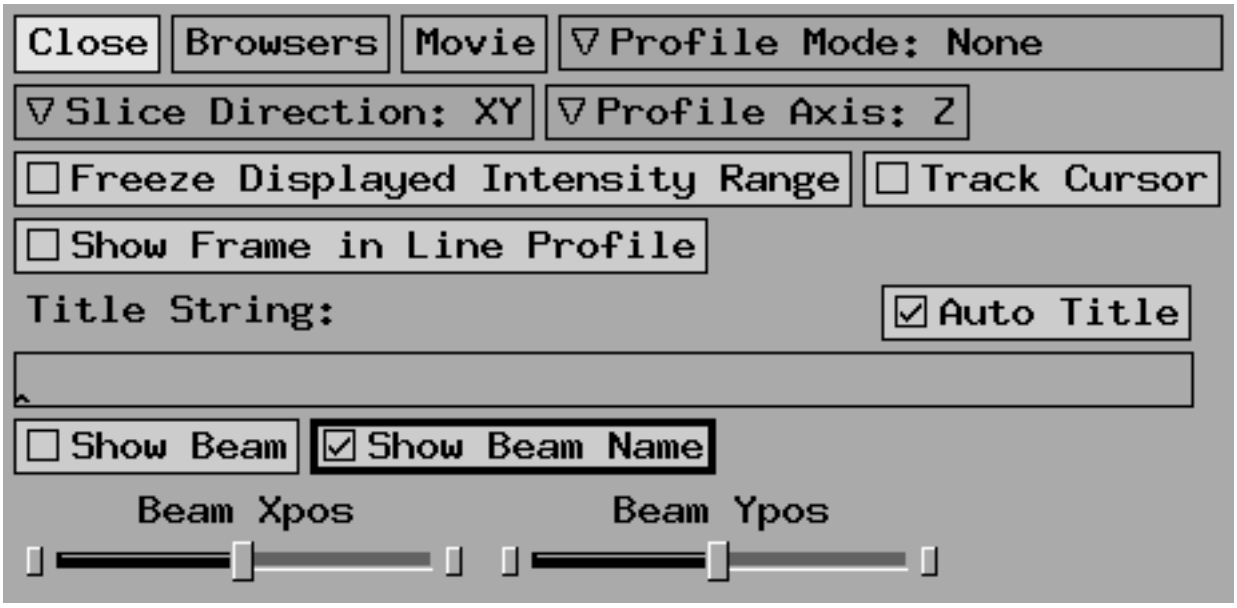

Figure 3.1: Screen snapshot of view control panel

3.3 Data Browser

This widget provides controls for managing the display of multiple datasets.
See Figure 3.2 for a screen snapshot.
The controls provided are:

Browser Controls

- **Close** this will close the window
- **Destroy** the blink-state that the browser is showing is destroyed. It is not possible to destroy the first blink-state for the display window
- **Copy** create a new blink-state (and browser) which has the same configuration as the currently viewed blink-state. In `all` mode, the browser for the new blink-state is popped up
- **Show** menu gives control over how the various blink-states are displayed. The following modes are available:
  - **all** each blink-state is viewed in its own browser
  - **some** each display window has a single browser, which can be switched to display any blink-state for the display window
- **Data** menu selects which kind of data is displayed in the browser. The choices are:
  - **Arrays** gridded array data (can be viewed as images or contours)

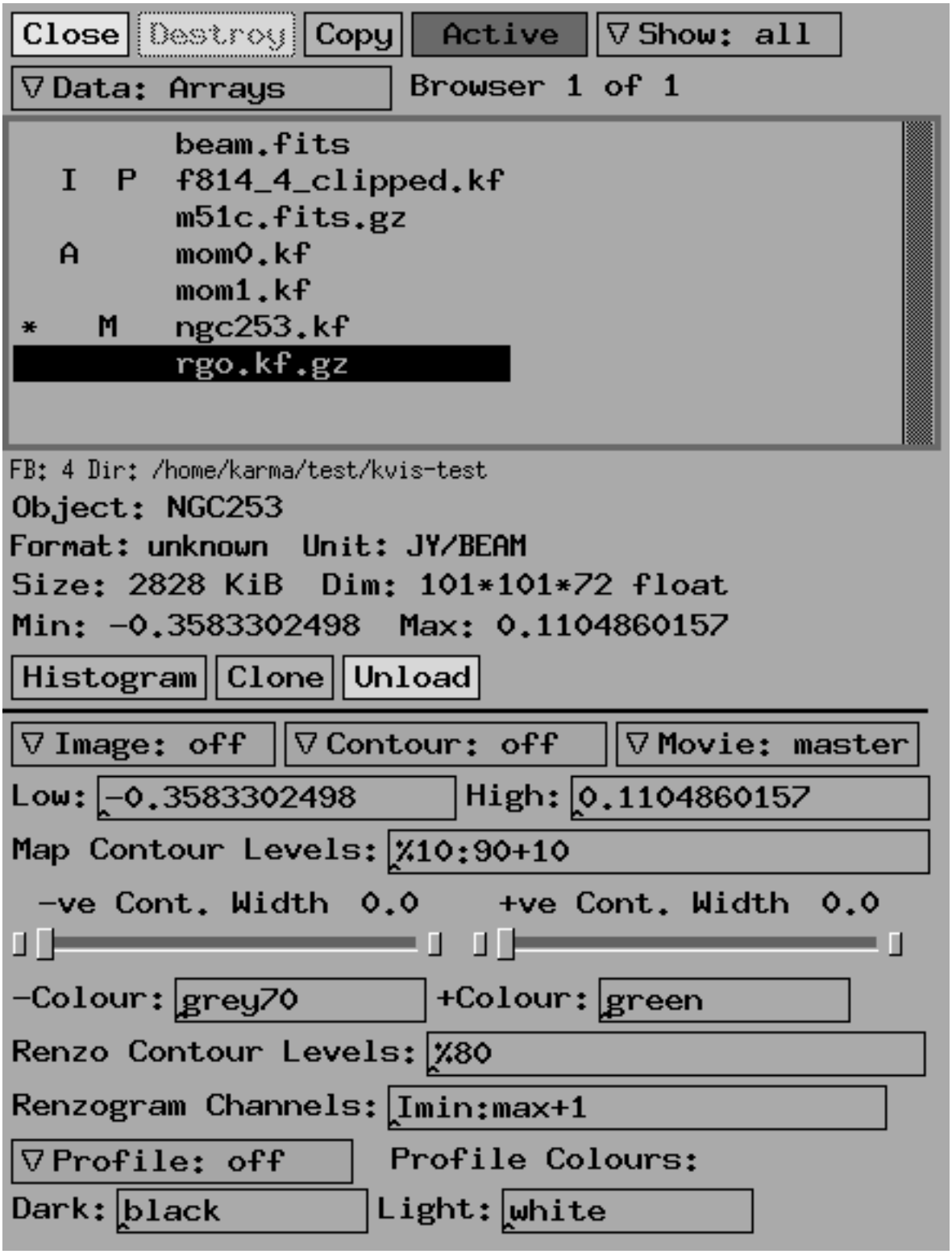

Figure 3.2: Screen snapshot of data browser

  - **Annotations** text and geometric figures which can be overlaid for annotation purposes

- **Advanced** this will raise the advanced control panel (section 3.3)
- **Make Data** this will raise the make data control panel (section 3.3)

Data List

Beneath these general controls, the list of loaded data appears. Each line shows the name and a summary of the configuration settings for that dataset. The list section is different for each data type. There are common operations that may be performed irrespective of data type, which are described here.

There is always a highlighted line, which shows which entry you can make changes to (using either the mouse buttons or the keyboard). The highlighted line is shown by a slightly different background colour. You can change the highlighted line by using the mouse or up/down arrow keys.

There is also a *selected* line, which shows which entry has more detail shown in the section(s) below. The list section only shows a summary of the current configuration for each dataset. The selected line is shown by using reverse video.

If the browser is in *active* mode (the default), then the *selected* line is locked to the highlighted line. If in *passive* mode, the *selected* line does not move until you explicitly select an entry.

If you select an entry while in *active* mode, the browser switches to *passive* mode. If in *passive* mode and you select the already selected entry, the browser switches to *active* mode.

As the mouse is moved over the data names, one of the names will be highlighted. The mouse and keyboard may be pressed to control the settings of the highlighted data. The following bindings are available:

- **Left Mouse** select the highlighted dataset for display in the sections below, and disable *active* mode. If the data is already selected and active mode is disabled, *active* mode is enabled
- **Spacebar** same effect as left mouse button
- **Up Arrow** highlight and select the previous data
- **Down Arrow** highlight and select the next data
- = (equals sign) selects this dataset for replacement (section 3.3), copying all data attributes
- $\sim$ (tilde sign) selects this dataset for replacement (section 3.3) without attribute copying

There are other key bindings which are independent of which dataset the mouse is highlighting:

- **Left Arrow** cycle backwards in the list of blink-states for the same window (in `Some` or `One` modes). Has no effect in `All` mode)

- **Right Arrow** as above, but cycle forwards
- **Shift Left Arrow** similar to `Left Arrow`, but also make the blink-state active
- **Shift Right Arrow** similar to `Right Arrow`, but also make the blink-state active
- **Control Left Arrow** cycle backwards to the last browser for the previous display window (`One` mode). Has no effect in other modes
- **Control Right Arrow** cycle forwards to the first browser for the next display window (`One` mode). Has no effect in other modes
- **Shift Control Left Arrow** similar to `Control Left Arrow`, but also make the blink-state active
- **Shift Control Right Arrow** similar to `Control Right Arrow`, but also make the blink-state active
- **b key** blink the display window to the next browser for this display window
- **B key** make the currently viewed blink-state active
- **>** **key** blink the display window to the next browser for this display window
- **<** **key** blink the display window to the next browser for this display window

The list section will show single character codes in a number of columns to the left of the data entry names. Different data types will show a different number of columns, and the character codes have different meanings as well. However following convention is followed:

- **=** (equals sign) this dataset has been selected for replacement (section 3.3) with data attribute copying
- **~** (tilde sign) this dataset has been selected for replacement (section 3.3) without attribute copying
- **uppercase** this is an exclusive option: only one entry may have this option set per blink-state
- **lowercase** non-inclusive option. Any number of entries may have this option set

Below the list of data, more detailed information and configuration settings are shown. The appearance depends on the kind of data being shown.

The configuration controls for the different data types are described below.

Data Replacement

As discussed above, a dataset may be selected for *replacement*. When a new dataset is loaded, the list of already loaded datasets is scanned for a dataset which has been selected for replacement, *and* was loaded from the same data source. A typical data source is a file browser, but may also include network connections. When a dataset is replaced with a new one, either all the old settings (such as intensity range, contour levels, etc.) are retained or none of the settings are retained, depending on which replacement mode was selected.

If more than one dataset is selected for replacement, and these datasets were loaded from the same data source, the selected dataset closest to the top of the list is replaced.

Note that if a data source is removed (e.g. a file browser is destroyed), then datasets loaded via that data source can no longer be replaced. These datasets can still be explicitly unloaded, however.

Array List

There are several columns used to display a summary of the configuration, using a simple legend. These are described below:

- **Image Column** shows which dataset(s) contribute to the displayed image. These are exclusive settings. The legend is:
  - **I** show this array as the image
  - **A** this array is the "alternate image" for intensity-intensity scatter plot computation hue-intensity
- **Contour Column** shows which datasets are displayed as contours. These are non-exclusive settings. The legend is:
  - **c** show this array as a normal contour map
  - **r** show this array as a Renzogram (only for cubes)
- **Movie Column** shows which datasets can be shown as a movie. The legend is:
  - **M** enable movie controls for this array
  - **m** slave this array to the movie controls
- **Profile Column** shows which datasets can be shown as profiles. The legend is:
  - **P** display line profiles of this array (this array controls the axis labelling and is drawn on top)
  - **p** display line profiles of this array

The following keyboard and mouse bindings are available to quickly control the configuration:

- **Middle Mouse** show as the image, the previous image becomes the "alternate image". If already displayed as the image, disable image display
- **Right Mouse** toggle between enabling and disabling contour display
- **i key** same effect as the middle mouse button
- **control-i** make the next dataset in the list the currently displayed image. This will cycle back to the top of the list when the end is reached
- **control-shift-i** make the previous dataset in the list the currently displayed image. This will cycle back to the end of the list when the top is reached
- **a key** make this image the "alternate image"
- **c key** same effect as the right mouse button
- **r key** toggle between enabling and disabling Renzogram display
- **control-c** enable contours and apply levels
- **control-r** enable Renzogram and apply level
- **M key** enable movie controls for this array
- **m key** slave this array to the movie controls
- **P key** display line profiles of this array (this array controls the axis labelling and is drawn on top)
- **p key** display line profiles of this array

Detailed Array Information

This shows information about the array, such as the directory from where the array was loaded, the format of the data (i.e. FITS, AIPS, Miriad, Gipsy and many more), the size and dimensionality and the range of data values. The following control buttons are available:

- **Histogram** pop up a histogram for this array. The histogram display can be used to control the intensity range for this array
- **Clone** create a virtual copy of the array, which will have independent configuration settings. Note that the data are not copied, instead a reference is made
- **Unload** unload the array

Note that these operations affect all browsers/blink-states and all display windows.

Array Configuration

How to display an array is controlled via this section. Some controls are per blink-state, while others are global and apply to all blink-states and all display windows. The following controls are per blink-state:

- **Image** menu. The choices are:
  - **off** the array is not selected for image display
  - **main** the array is selected for normal image display
  - **alt** the array is the "alternate" image for computing intensity-intensity scatter plots
- **Contour** menu. The choices are:
  - **off** the array is not displayed as contours
  - **map** the array is shown as normal contours
  - **renzo** the array is shown as a "Renzogram"
- **Movie** menu. The choices are:
  - **off** the array will not be shown as a movie
  - **master** the array is shown as a movie and controls the frames
  - **slave** the array is shown as a movie and is slaved to the master

The following controls are global to the application, and affect all blink-states and display windows:

- **Low** the low end of the intensity range
- **High** the high end of the intensity range
- **Map Contour Levels** the contour levels for normal contours. The syntax for the contour levels is as follows:
  - just type the levels you want e.g. `0.001 0.002 0.003 0.004`
  - levels with a constant increment: `0.001:0.004+0.001` means that the first level is `0.001`, that this level is incremented with `0.001` until `0.004` is reached. So this example gives the same levels as in the first case
  - levels incremented by a factor: `0.001:0.016*2`. This gives levels `0.001 0.002 0.004 0.008 0.016`
  - levels as percentage of the peak: first character should be %, numbers are taken as percentage, the rest of the syntax is as above, e.g. %10:40+10 gives 10, 20, 30 and 40 % of the maximum of the data
  - the special values `min` and `max` indicate the minimum and maximum value

These options can be mixed, i.e. `-0.03:-0.01+0.01 0.003:0.1+0.02 0.2 0.3` is interpreted correctly. You should press the enter key for your changes to take effect

- **-ve Cont. Width** the width of negative contour levels
- **+ve Cont. Width** the width of positive contour levels
- **-Colour** the colour of negative contour levels
- **+Colour** the colour of positive contour levels
- **Renzo Contour Levels** the contour levels for Renzograms. The syntax is the same as for the contour levels
- **Renzogram Channels** the channels (along the unseen axis) to display in the Renzogram. The syntax is similar as for the contour levels, except that a leading "I" or "W" must be given. Again, press the enter key for your changes to take effect. These leading characters have the following meaning:
  - **I** specify channel **I**ndices. Thus `Imin:max+1` selects channels between channel 0 (the first) and the last channel, in single channel increments (i.e. all channels). For example, if you wanted to display channels 10 to 20 inclusive, type: `I10:20+1`
  - **W** specify **W**orld co-ordinates (i.e. velocity in m/s or frequency in Hz). For example, if you wanted to show channels between 1418 and 1420 MHz, in 100 KHz steps, type: `W1418e6:1420e6+1e5`
- **Profile** menu. The choices are:
  - **off** the array will not be shown as a profile
  - **master** the array is shown as a profile, is drawn on top and controls the axis labels
  - **slave** the array is shown as a profile and is slaved to the master
- **Dark** the colour to use for profile display. This must be a dark colour, appropriate for display on light backgrounds (i.e. white paper)
- **Light** the colour to use for profile display. This must be a light colour, appropriate for display on dark backgrounds (i.e. a computer monitor)

Advanced Control Panel

This control panel allows you to control some extra aspects of the behaviour of all data browsers. The following controls are available:

- **Close** this will close the window
- **Clip Mode** menu controls how changes in the low and high clip levels affect display windows. The following modes are available:

  - **Image and Profiles** the image intensity range and profile vertical range are synchronised and clip level changes affect both
  - **Images only** only the image intensity range is affected by clip level changes
  - **Profiles only** only the profile vertical range is affected by clip level changes
  - **Both later** the image intensity range and profile vertical range are affected by subsequent clip level changes, but they are not immediately synchronised

- **Range changes affect all arrays** if this is turned on, an attempt to change the clip levels of any array dataset will change the clip levels for all array datasets
- **Make slaved movie on load** if this is turned on, an array dataset with three or more dimensions will be automatically marked as a movie slave when loaded
- **Activate annotations on load** if this is turned on, an annotation dataset is automatically made active (i.e. displayed) when loaded
- **Slave Profile H-range to image window** if this is turned on, the horizontal range of the profile window is set to the same as the horizontal range of the image window if the the axes correspond
- **Show new data as Image** if this is turned on, an array dataset is marked as the active image when loaded

Make Data Control Panel

This control panel allows you to create new array data from existing array data. This is useful if you want to combine data together. The following controls are available:

- **Close** this will close the window
- **Algorithm** menu controls how data arrays are combined. The following modes are available:
  - **RGB** combine one, two or three datasets into an RGB (red, green and blue) array. Each colour component is 8 bits, yielding a 24 bit array. The input values between the low and high clip are linearly scaled into the output. If the clip values on an input array are subsequently changed, the RGB image is rescaled. The following bindings are available:
    * **R** use this dataset as the red colour component
    * **G** use this dataset as the green colour component
    * **B** use this dataset as the blue colour component

        * **Left Mouse** use this dataset as the red colour component
        * **Middle Mouse** use this dataset as the green colour component
        * **Right Mouse** use this dataset as the blue colour component
        * # use this dataset as the grid template
    - **Arithmetic** combine multiple datasets using simple arithmetic (addition, subtraction, multiplication or division). The output dataset is computed by processing the list of datasets from top to bottom, applying the specified arithmetic operations for selected datasets. Before processing any datasets, the output dataset is initialised to either zeros (if the first operator is addition or subtraction) or ones (if the first operator is multiplication or division). The following bindings are available:
        * + add this dataset to the output dataset
        * **-** subtract this dataset from the output dataset
        * ***** multiply the output dataset by this dataset
        * / divide the output dataset by this dataset
    - **Hue Intensity** combine two datasets into a Hue-Intensity array. The two datasets may be directly mapped to intensity and hue, or they can represent a complex image. The real and imaginary components will be in separate datasets and they are converted to amplitude and phase values, which are then mapped to intensity and hue. The following bindings are available:
        * **B** use this dataset as the brightness (intensity) component
        * **H** use this dataset as the hue (colour) component
        * **R** use this dataset as the real component
        * **I** use this dataset as the imaginary component
        * **Left Mouse** use this dataset as the brightness component
        * **Middle Mouse** use this dataset as the hue component
        * **Right Mouse** use this dataset as the grid template
        * # use this dataset as the grid template

- the dataset list shows the available datasets. This is used to mark which datasets are used as inputs to the combining process. This list has a similar interface as the dataset list in the data browser. Note that in most cases, a dataset can only be marked once. If you need to mark a dataset for multiple purposes (such as to use it for both red and green inputs), you will need to clone the dataset first

- **Name** this allows you to specify the name for the new dataset. A default name is provided

- **Make** this will create the new dataset, based on the selections you have made. This could take some time, depending on how large the input datasets are

Annotation List

There is just one column used to display a summary of the configuration. The `a` character is used to indicate whether a annotation file is active (visible) or not. The `a` key may be used to switch between the two states. The middle mouse button has the same effect of switching between the two states.

Detailed Annotation Information

This shows where directory from which the annotation file was loaded. the `Unload` button may be used to unload the annotation file. This affects all blink-states and display windows.

Annotation Configuration

The `Active` toggle may be used to switch between active (visible) and inactive settings for this annotation file.

3.4 Command-line Options

You may pass the names of files to be loaded via the command-line. You may specify as many files as you desire, up to the limit imposed by the operating system.

3.5 Intensity-Intensity Scatter Plots

Earlier in this chapter (3.2) is a brief description of how you can use the `c` key to compute and display an intensity-intensity scatter plot of two images. You may also use this facility to display a scatter plot of two channels from the same cube. Simply *clone* the cube (i.e. using the `Clone` data browser button), and select the two channels, one from each cube. You will need to ensure that the original and cloned cube do not have a master/slave movie relationship.

If there is a master/slave movie relationship, then the scatter plot is computed using the two frames which have corresponding co-ordinates for the frame axes. This is usually the desired setting when generating scatter plots from frames of two different cubes.

Note that when computing a scatter plot from two images with a different co-ordinate grid, the pixels from the unseen image correspond to pixels with the same world co-ordinates in the visible image. Thus, there is no need to re-grid either of the images.

3.6 Renzograms, an alternative to velocity fields

A "renzogram" is a technique for displaying velocity information in a cube in a different (better) way than velocity fields. The technique was pioneered by Renzo Sancisi at the Kapteyn Institute in Groningen. A "renzogram" overlays a single-level contour map for every channel in a cube over an image (such

as an optical image or moment map). The contour map for each channel is displayed in a different colour (typically red-blue, although this can be changed interactively). This has an advantage over velocity fields because it shows multiple peaks in the velocity profile, which a velocity field cannot do.

To use this feature, simply load a cube and select it for renzogram display in the data browser. You can control the contour level used for the renzogram, and the range of channels displayed.

You may wish to load a separate image (perhaps an optical image) and display that as the image. This can be useful for comparing optical and radio data.

You can also view a profile (section 2.11) (spectrum) of your cube. Just select select line profile display and a profile window will appear. You will note that the profile is drawn in the same colours as the contours.

Chapter 4

Viewing Images and Cubes with the Multibeam

Note that this tool will be replaced with the more powerful <**kvis**> (chapter 3) programme.

The <**MultibeamView**> programme allows you to view two datasets. This allows you to compare different datasets in a very easy fashion. This programme uses the <**ImageDisplay**> widget (section 2.15) to provide the usual interface. Pressing the **View** button will pop up the <**View2Datasets**> widget. This widget provides an advanced control for two datasets (either two-dimensional or three-dimensional). It allows the user to display one dataset or the other, overlay contours of one over the other, show profiles along any axis, and much more. The controls provided are:

- **Close** this will close the window
- **Movie** this will pop up the <**AnimateControl**> widget (section 2.3), which will allow you to play movies and step through frames. In addition, the following key bindings are defined for the main display window:
  - **PgDn** go to previous frame
  - **PgUp** go to next frame
  - **Home** go to start frame
  - **End** go to end frame

  On some keyboards, `PgDn` is marked as `Next` and `PgUp` is marked as `Prev`.
- **Blink** if enabled, dataset 2 is displayed (if the display mode is set to blink), if disabled, dataset 1 is displayed. By repeatedly clicking this, the user can blink between the two datasets. You can also press the "b" key in the image window and get the same effect. Note that when blinking between two images with non-linear co-ordinate systems, the linear (data pixel) co-ordinates of the canvas boundary are preserved, *not the non-linear world co-ordinates*. This is especially noticable when blinking between two images with a different pixel scale

- **Profile Mode Menu** this allows you to pop up the <**TracePopup**> widget (section 2.11) and choose the profile mode. The following modes are available:
  - **None** will pop down the <**TracePopup**> widget
  - **Line** will show a single profile in the selected direction
  - **Box Sum** lets you draw a box over the image window (using the middle mouse button, in a similar way as using the left mouse button to zoom) and then the sum of all the profiles in that region is displayed. Box mode only works for 3-dimensional datasets, and will always display a profile along the unseen dimension
  - **Box Average** is like the **box sum** mode except that the average of all profiles is displayed rather than the sum
  - **Radial** will azimuthally average the image plane data and produce a plot of average value versus radius. A gaussian is then fitted to the radial profile and overplotted in red and the FWHM (full width half max) of the gaussian is displayed over the profile. You must click the middle mouse button to define the centre of the object of interest, then, keeping the mouse button down, drag out to define the maximum radius of the profile and release the button to display. The algorithm automatically adjusts the centre position using a centroiding operation, so you don't have to be too careful about picking the centre of your object. More information on the **centroidingalgorithm** (`http://www.atnf.csiro.au/karma/user-manual/centroidingalgorithm.html#pubc`e can be found in the Karma User Manual.

    The programme should also work with a photographic negative. The centroiding algorithm is automatically modified to look for a trough rather than a peak
  - **Boxed Horiz** will produce a horizontal profile which is averaged vertically. The box is drawn the same way as for the **box sum** mode. The horizontal centroid of the profile is computed and a gaussian is fitted and displayed
  - **Boxed Vert** will produce a vertical profile which is averaged horizontally. The box is drawn the same way as for the **box sum** mode. The vertical centroid of the profile is computed and a gaussian is fitted and displayed

- **Slice Direction Menu** this allows you to choose how you want to slice your cube. You can view XY, XZ or ZY planes
- **Profile Axis Menu** this allows you to choose along which axis you want the profile to be displayed (valid only for line profile mode)
- **Display Mode Menu** this allows you to choose how you want to view your two datasets. The following choices are available:

  - **Set 1->image** view dataset 1 as an image
  - **Set 2->image** view dataset 2 as an image
  - **Set 1->image, Set 2->contour** view dataset 1 as an image and overlay dataset 2 as contours
  - **Set 1->contour, Set 2->image** view dataset 2 as an image and overlay dataset 1 as contours
  - **Blink between Set 1 and Set 2** enable blinking between the two datasets

- **Link Intensity Scales** if this is set then if the intensity scale for either dataset is modified using the <**Dataclip**> (section 2.2) widget then the intensity scaling for *both* datasets is changed. This can be useful when blinking between two images

- **Link Frames** if this is set then whenever the the current frame number for the image dataset is changed, the frame number for the other dataset is also updated to have the same value. If the two frame dimensions have the same co-ordinate type, then the co-ordinate values are matched, rather than co-ordinate indices. This feature is often used when displaying a movie of both datasets, one as images and one as contours

- **Show Frame in Line Profile** if this is set the current frame displayed in the main image window is shown as a vertical red line in the profile window. This marker is only drawn if the profile axis is the unseen axis

- **Contour Levels** this controls the contour levels. Be careful not to hit return after filling in the levels!!! The syntax for the contour levels is as follows:

  - just type the levels you want eg. `0.001 0.002 0.003 0.004`
  - levels with a constant increment: `0.001:0.004+0.001` means that the first level is `0.001`, that this level is incremented with `0.001` until `0.004` is reached. So this example gives the same levels as in the first case
  - levels incremented by a factor: `0.001:0.016*2`. This gives levels `0.001 0.002 0.004 0.008 0.016`
  - levels as percentage of the peak: first character should be %, numbers are taken as percentage, the rest of the syntax is as above, e.g. %10:40+10 gives 10, 20, 30 and 40 % of the maximum of the data

  These options can be mixed, i.e. `-0.03:-0.01+0.01 0.003:0.1+0.02 0.2 0.3` is interpreted correctly.

- **Contour Colour** specifies the colour the contours should be drawn in

- **Apply Levels** this will apply the contour levels and colour

- **Reset Levels** will reset the contour levels to the default

- **Auto Apply** if this is on, then if you load a new dataset over one that is currently being displayed as contours, the old contour levels are automatically applied. The reason this is off by default is because the contour levels for one dataset may be incorrect for another. An inappropriate set of contours can take a long time to display, so mistakes are costly
- **Axis Labelling Title** this allows you to change the title placed above the image when axis labelling is enabled. Normally, the title is automatically generated
- **-ve Contour Width** this controls the width of negative contours
- **+ve Contour Width** this controls the width of positive contours
- **Auto Title** if enabled, the axis labelling title is automatically generated. Whenever this is disabled, the string specified by **Axis Labelling Title** is used instead
- **Show Beam** if enabled, the dataset headers are searched for "BPA", "BMAJ" and "BMIN" FITS-style keywords. If these keywords are present, a representation of the telescope beam is overlaid on the image
- **Show Beam Name** if enabled, the name of the dataset for each beam is placed near the beam representation
- **Set 1 Beam Xpos** controls the horizontal position of the the beam for dataset 1
- **Set 1 Beam Ypos** controls the vertical position of the the beam for dataset 1
- **Set 2 Beam Xpos** controls the horizontal position of the the beam for dataset 2
- **Set 2 Beam Ypos** controls the vertical position of the the beam for dataset 2

In addition, if you click the left mouse button in the display window without moving it in between the press and release, the image will pan across. If you click the middle mouse button, the image will zoom in 2x (the new centre of the image will be the place where you clicked). Click the right mouse button to zoom out 2x. If the profile mode is "box" then you can't use the middle mouse button to zoom.

Pressing the `c` key in the display window will compute and display a scatter plot of intensity values in one image versus intensity values in the other image. This is useful for seeing if there is a correlation between the values in the two images. You can use the left mouse button to define a sub-image (similarly to zooming in) from which the scatter plot is computed.

See Figure 4.1 for a screen snapshot.

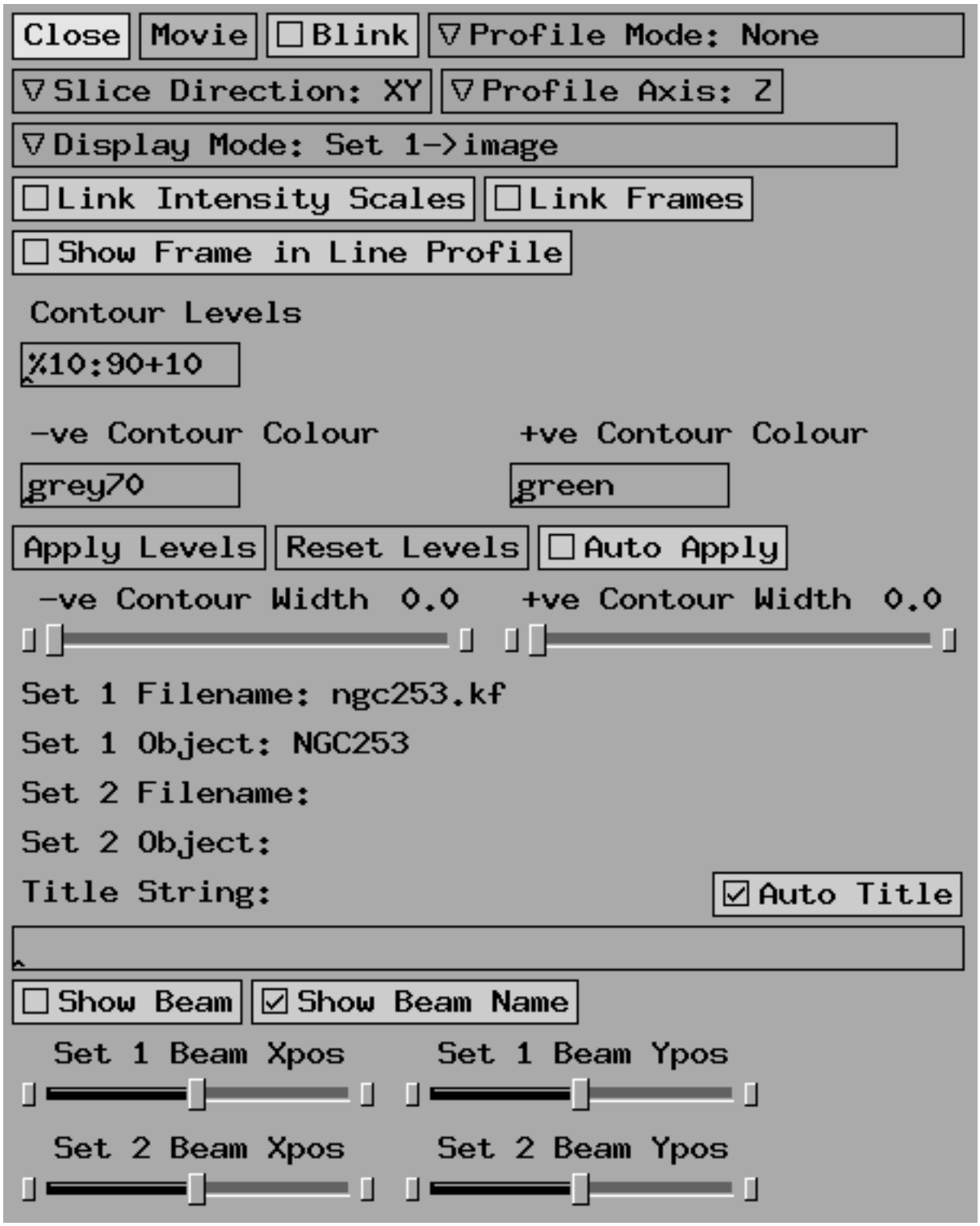

Figure 4.1: Screen snapshot of two datasets display control panel

4.1 Command-line Options

The following command-line options are supported by <**MultibeamView**>:

- `-set1 <filename>` this will load the dataset `filename` as set 1

4.2 Special Movies

If you want to play a movie that was made using the Hot Gas Substances (section 5.1.1) or the Hot Gas Continuous (section 5.1.1) shaders of the xray (chapter 5) volume rendering tool, you should compress this movie from 24-bit to 8-bit. Movies made with the other algorithms do not require compression and can be played directly. The compression of 24-bit movies is done with the command:

`conv_24to8 -com w -max <max num of colours> - <inmovie>`

The switch **-com w** tells the program that it should compress such that it has the same colourmap for all the images. In principle you could leave out this switch, but that is not recommended. Your movie will look awful.

The switch **-max <max num of colours>** gives the maximum number of colours the final movie can have. The recommended value to use is **-max 200**. Do not give a higher number than 220.

Please note the space between the name of the movie and the minus-sign.

You do not have control over the name of the output file. The string `conv_24to8_` is prepended to the name of the input file:

`conv_24to8 -c w -m 200 - ugc12441_movie`

produces a file `conv_24to8_ugc12441_movie.kf` (!) with the compressed movie.

4.3 Multiple Contour Images

The <**MultibeamView**> programme has special-purpose code to display two datasets, one as images and the other as contours. If you wish to display multiple images or cubes, you should use the <**kvis**> (chapter 3) programme.

4.4 Intensity-Intensity Scatter Plots

Earlier in this chapter (4) is a brief description of how you can use the `c` key to compute and display an intensity-intensity scatter plot of two images. You may also use this facility to display a scatter plot of two channels from the same cube. Simply load the same cube in twice (i.e. using the `Set1` and `Set2` file browser buttons), and select the two channels, one from each cube. You will need to ensure that the **Link Frames** toggle is disabled.

If the **Link Frames** toggle is enabled, then the scatter plot is computed using the two frames which have corresponding co-ordinates for the frame dimensions. This is usually the desired setting when generating scatter plots from frames of two different cubes.

Note that when computing a scatter plot from two images with a different co-ordinate grid, the pixels from the unseen image correspond to pixels with the same world co-ordinates in the visible image. Thus, there is no need to regrid either of the images.

Chapter 5

Volume Rendering a Cube

The <**xray**> programme is used to volume render cubes.

5.1 How the rendering is done

You can think of the way a data cube is rendered as lines of sight going from every pixel of the display through the data cube and the *voxels* (visualisation speak for volume element of the data cube) on a line of sight contribute in some way to the colour and intensity of the display pixel. To decide how every voxel contributes, a number of algorithms (called **shaders**) are available.

5.1.1 Shaders

Below are two lists with the shaders available. The first list contains simple shaders. The **Voxel sum** and the **Maximum voxel** shaders are useful for exploring the data cube, because they are relatively fast and give a reasonable display of the data.

- **Voxel sum** The value of the display pixel is the sum of all voxel values along the line of sight. This is a fast and quite useful way of representing the data. With this shader, one gets a good global view of the emission. It is less useful for looking at small details
- **Minimum voxel** The pixel value is the minimum voxel along the line of sight
- **Maximum voxel** The pixel value is the maximum voxel along the line of sight. This is a fast way of getting a reasonable image. It also allows, depending on your data, to have a view of the 'inside' of your data, the object looks more transparent than with the **Voxel sum** shader
- **Front voxel** The pixel gets the value of the first voxel along the line of sight above a specified threshold (not usually useful, the image is very blocky and has many holes). You get a control window where you can change the threshold value by clicking the left mouse button

The second kind of shaders uses an equation of radiative transfer to compute the images. These are called **hot gas** shaders.

The contribution of a voxel to the pixel on the display is calculated using a simple equation of radiative transfer, where a voxel is partly absorbed by voxels that are in front of it:

$$c_i = c_{i-1} \cdot (1 - o_i) + s_i \cdot o_i, \tag{5.1}$$

where c_i is the intensity along a line of sight after adding the i^{th} voxel in front, s_i the intensity of the voxel added, c_{i-1} the intensity along the line of sight that is behind the i^{th} voxel, and o_i the opacity of the voxel i. This calculation is done 'back-to-front', so voxels in the back of the cube are partly obscured by voxels in the front part of the cube.

If the opacity for a voxel is zero, that voxel becomes completely transparent *but also invisible*. This closely resembles the behaviour of a self-radiating cloud of hot gas, where gas particles both emit and absorb radiation. Regions with high gas particle density will have a higher surface brightness, but will also have a higher cross-sectional area of absorption.

The resulting rendered image appears like a cloud of glowing gas, which shows both internal and external structure.

You can use this to make features disappear (eg. noise). The disadvantage is that in order to make the data that you want to see transparent, you also make it barely visible. Consequently, the opacity of data you want to see has to be not too low and as a result one will tend to see only the surface of the emission regions.

There are a few versions of this implemented, differing in whether they do the rendering monochromatically or in colour.

- **Hot Gas: substances** This works as follows: the value range [–127, 127] of the voxels is divided into a number of subranges (typically eight). Each subrange has a colour and an opacity associated to it and is called a **substance**. Like every colour in computer displays, a colour consists of red, green and blue.

  The contribution of a voxel to the colour of the pixel on the display is calculated using the radiative transfer, but for each colour (red, green or blue) separately. This means that if a blue voxel is behind a red one, the blue is still visible. This makes the data appear more transparent. It also means that this shader produces images that are 24-bit deep (one byte for every colour) and to change colours on the display, you will have to manipulate the substances. It also means that to play movies made with this shader, you may want to convert the images to 8 bit using the conv24to8 programme (section **??**).

  A practical disadvantage of this shader is that it is difficult to set the colours and the opacities of the substances exactly as you want, there are many many parameters to set. It takes quite some time to produce reasonable images with this shader. You can save time by only defining

a small number of substances (say the default eight), but then the representation of the data is somewhat schematic. If you define the maximum number of substances, you can set up a substance table with a smoother colour transition, but of course it takes more time for you to set up the table.

A strong advantage of this shader is it can be useful to display both absorption and emission in a datacube.

- **Hot Gas Mono** This shader also uses opacities, but the control is much simpler. This shader takes

$$o \propto v^{\alpha} \tag{5.2}$$

where o is again the opacity, v the value of the voxel. By setting α you can control which parts of the data are visible/opaque. A small value of α makes low-level emission already quite opaque, while a large value will show only the brighter voxels. Also, the computation is done on the floating point data, not on colours (hence the **mono**). This increases the dynamic range of the image very much. Images produced by this shaders are floating point.

An option offered with this shader that the intensity can be transformed before rendering, according to a similar function as above:

$$s \propto v^{\beta} \tag{5.3}$$

where s is the value used for the voxel in the radiative transfer and v is the value of the voxel.

- **Hot Gas Continuous** this shader is a mix between the **Hot Gas Substances** and the **Hot Gas Mono** shaders. There are now 254 substances, but the opacity is defined in the same way as in **Hot Gas Mono**. The user interface of this shader is much better than that of **hot gas substances**. Also this shader produces 24bit images and to play movies made with this shader, you will have to convert the images to 8 bit using the conv24to8 programme (section **??**).

5.2 Starting up the Volume Rendering Tool

To start the rendering software on any workstation type

```
xray
```

5.2.1 Options

<**xray**> supports several command line options which are all documented in the manual page. The options of general interest are:

- **-private_cmap** This option will force <**xray**> to use a private colourmap for its PseudoColour window. Otherwise, <**xray**> trys to allocate colours from the default colourmap

- **-num_colours** This option specifies the number of colour cells that <**xray**> will try to allocate for its PsuedoColour window upon startup. If less colours are available it allocates as many as possible (minimum 2).

- **-fullscreen** This option will make the image display window take up the entire screen. This is useful if you wish to make a video of the data. You will need to configure your window manager to cycle the window stacking order when a special key is pressed (e.g. the "Back" key with the Open Look window manager). See also the appendix on making videos (appendix E).

- **-no_slice_win** This option will disable the 3D slicing window similar to what you get in kslice_3d (chapter 7)

After starting up, many windows will appear:

- the control window, with which of course you control the rendering program

- a slice window. This shows the data cube along the three principle planes: *XY, ZY* and *XZ*. By moving the mouse (press left), the windows will be updated. To make things run smoothly, click on **Precompute**. This is also available as a stand-alone program kslice_3d (chapter 7) and is quite useful for inspecting your data. This window will appear after you have loaded a cube (and possibly converted to bytes)

Another window that you will see (after the first time you render the cube) is the image display. This window is similar to the display window in many of the other tools (see section 2.15).

5.3 Loading Data into xray

<**xray**> supports many data formats like any other **Karma** programme. For reasons of memory use and speed, the preferred data format used by <**xray**> is that the data is stored as bytes, with values ranging from –127 to 127 (the value –128 is used for blank or missing data). The values in your data cube will have to be scaled into this range.

If you load a non–byte data cube, <**xray**> will display a window showing the histogram of the data values in the cube. If you want to use the full range of values in the cube (good for a first look at the data), click (left) on the **Full Range** button. If you want to select a sub–range of the data, click (left) in the histogram window to define the lower bound and click (right) to define the upper bound. You then click (left) on **Apply** to tell <**xray**> to compress the cube to a byte cube. You will then be able to click (left) on the **Save** button,

to save the compressed data cube. The default filename is "scaled", but you can change this prior to saving. Once you have saved a compressed cube, the next time you load it in <**xray**> it will load much faster.

The proper values for the scaling range of course depend on the data in the cube and on what you want to see in the data, but here are some hints. Normally, the histogram has a large peak around zero (The Noise). Unless you want to have a look at the noise in your data cube (eg. because you want to look at subtle calibration errors or errors in the continuum subtraction, errors that show up quite well with the rendering software), it may be a good idea to take the minimum such that most of this peak is excluded. Depending on which voxel algorithm you use in the rendering (see later), if you include data at the noise level, your rendered data cube will have a lot of noise 'in front' of your object of interest and it may just be in the way. There are however also other ways (see below) to hide the noise.

Consider that with the rendering software you may see faint structures in your data that you were not aware of. So you may not want the minimum to be too high either otherwise you may miss interesting features in your data. Of course, when there is absorption in the data, the noise will have to be included.

Also consider that the because the data is scaled into [–127, 127], the dynamic range is limited, so in some cases it may be a good idea to choose the maximum not too high. You will be able to see more detail in the fainter emission.

5.4 The main control window

See Figure 5.1 for a screen snapshot.

The following controls are available:

- **Files** this pops up a standard <**Filepopup**> widget (section 2.1) with which you can load your cube
- **Show Orientation** this pops up a window which shows a wireframe of the cube. As you change the cube rotations this is updated immediately. This should give you a feel for which way the cube will rotate before you press **Compute**
- **Make Movie** this pops up a movie control panel. This is descibed in section 5.5
- **Rescale** if you loaded a non-byte cube then you can bring up the histogram display using this button and then you can set the data scaling again
- **Filter** this pops up a filtering control panel described in section 5.6
- **Quit** this will quit the application
- **Hide Wireframe** if disabled, a wireframe showing the visible edges of the cube is displayed over the rendered data, to show how the data cube is

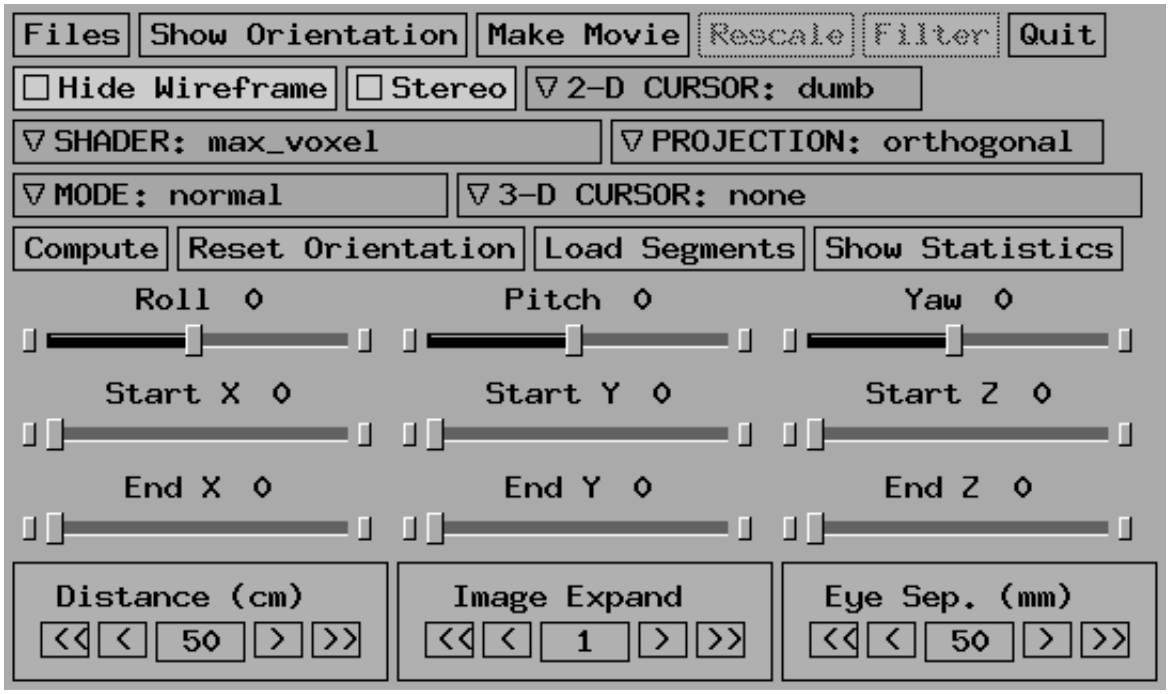

Figure 5.1: Screen snapshot of main control panel

rotated. It also helps the brain to perceive the data as three dimensional. If you don't like the wireframe, you should enable this toggle

- **Stereo** if your are sitting in front of **phoenix** and you started the window system in stereo mode, clicking on **Stereo** will enable stereo display. The way this works is that slightly different images are displayed in your left and right eyes, mimicking the way you see depth normally. You will hopefully see the rendered cube as a three-dimensional object with depth. To make use of this you need to wear a pair of "Crystal Eyes" glasses (these fit over most prescription glasses). If you want to use this feature please ask the system administrator who will lend you the glasses. These things are rather expensive so we don't just leave them lying around.

  Because of a problem with the vendors graphics library it is best to use the stereo option when rendering to 24bit images (i.e. when using the "Hot Gas Substances" or "Hot Gas Continuous" shaders)

- **2-D Cursor Menu**

  This menu allows you to switch from the default cursor to an experimental 2-dimensional "active" cursor. The default (or "dumb") cursor will allow you to read image values and positions in the display window in the conventional way. The "active" cursor allows you to point at a feature (say a blob) on your volume rendered cube and obtain the 3-dimensional position of that feature. The horizontal and vertical position information is accurate, however the depth position is computed using a heuristic and

is less accurate. In another window you will see a plot of the cube values versus the position along the line of sight, drawn in white. A yellow vertical line shows the depth of the feature you are pointing at (really the depth the algorithm thinks the feature is at). Both the 3-dimensional position information and the plot window are dynamically updated as you move the cursor in the image display window.

- **Shader Menu** this allows you to select which shader to use. See section 5.1.1

- **Projection Menu** this allows you to select a prjection. Two projections are available: **Orthogonal** and **Perspective**. If you select the perspective projection, you can control the distance of the cube (i.e. the amount of perspective) with the **Distance** slider in the control window. The main reason that the **Perspective** projection is there is that this is quite useful for stereo display. Even if you are not using stereo mode, you may want to try it anyway. It helps for the 3D perception of the images

- **Mode Menu** this allows you to choose the rendering mode. The process of projecting rays into the data volume and collecting the data along each ray can be quite involved, and there are speed/quality tradeoffs. The **Mode Menu** allows you to choose between these tradeoffs.

- **3-D Cursor Menu** this allows you to choose the kind of 3-dimensional cursor available. This code is currently in development, so things may change rapidly. An overlay cursor is available (the cursor is drawn over the rendered volume), so the cursor will not dissappear behind a feature in the data. A proper translucent cursor is also available. You can move the position of the 3-D cursor by pressing the left mouse button in any of the three slice windows (the ones with the red crosshairs) and dragging the mouse

- **Compute** this will cause the cube to be rendered using all the current parameters

- **Reset Orientation** this will reset the view and then renderer the cube

- **Load Segments** this will bring up a file browser which allows you to load files with extension `.seg`. These files contain pairs of three-dimensional co-ordinates, each pair defining a line in the world co-ordinate system of the volume. These 3D lines are drawn through the volume and are obscured by opaque structure in the volume data. The files are simple ASCII format, where each line defines one pair of 3D points. Note that the endpoints are projected from 3D to screen co-ordinates, and then the points are joined by a straight line on the screen. For best results your lines in 3D space should be short

- **Roll Slider** this controls the roll angle. Think of an airplane rotating about it's long axis

- **Pitch Slider** this controls the pitch angle. The nose of the airplane dips up and down
- **Yaw Slider** this controls the yaw angle. The nose of the airplane swings left and right
- **Start X Slider** this controls the starting X position of the subcube to render
- **Start Y Slider** this controls the starting Y position of the subcube to render
- **Start Z Slider** this controls the starting Z position of the subcube to render
- **End X Slider** this controls the end X position of the subcube to render
- **End Y Slider** this controls the end Y position of the subcube to render
- **End Z Slider** this controls the end Z position of the subcube to render
- **Distance** this changes the distance your viewpoint is from the cube. This only has effect when using the **Perspective** projection
- **Image Expansion** Often, the image generated by the volume redering software is rather small (this is because the original cube was small). The software automatically expands the image so that it fills as much as possible the display window, using simple pixel replication. This results in blocky images, which may be undesirable. Setting the expansion factor to a value greater than 1 uses an anti-aliasing image expansion algorithm which maintains the original smoothness of the image. (See note on anti-alising for more information). Setting this to too large a value will result in very large movies, which is undesirable. Try a vlaue of 2 to start with
- **Eye Separation** When viewing the cube in stereo mode, it is useful to enhance or diminish the stereo separation. The lower the value, the closer the left and right eye viewpoints, and hence the lesser the stereo effect.

5.4.1 Rendering Modes

The following rendering modes are defined:

- **normal** this is the default mode. Because rays through the volume will not fall exactly on grid points, some approximation of the value has to be made. In this mode the ray is snapped to a grid point. This can result in aliasing problems, but is the fastest mode. In most cases the aliasing is not a problem. In addition, this mode computes a rotated version of the cube in a cache in the background while you are not doing anything (the background processing does not slow down normal rendering operations). The cube is rotated using the same grid snapping technique, so has the same aliasing problems. However, provided you do not keep changing the

view (or the cube), subsequent render operations (such as changing the shader) are four to ten times faster. Whenever you rotate the cube (or change it) the cache is automatically built up again. <**xray**> always uses as much of the cache as it can

- **smooth cache** will build up the cache using a better algorithm when a ray falls between grid points. The value computed uses a distance weighted contribution from each of the nearest surrounding values; in effect an interpolation. This algorithm does not suffer from aliasing. The cache takes about four times longer to build up than in **normal** mode, but once it's done rendering is as fast as using the cache that **normal** mode produces. In fact, it doesn't matter how complex the cache building process is, once it's built rendering times are the same (for the same shader, of course). If the cache is not completed, part of the rendering uses the **normal** mode when you request an update. This can lead to the interesting side-effect that the lower part of the rendered volume is smoother whilst the upper part is rougher (and possibly shows aliasing artefacts)

- **always smooth** will always render using the interpolation algorithm. It does this by waiting for the cache to be built using the interpolation algorithm and then rendering. If you don't change your view or cube, subsequent rendering operations are still at maximum speed. If you find that you are having aliasing problems, you should use this option when making a movie (otherwise the movie generation process will not wait for the cache to be built up)

- **smooth delayed** will first render using the **normal** mode and will then build up the cache using the interpolation algorithm in the background. Provided you don't change the view or cube parameters, once the cache is computed the cube will be rendered again and you will see a refined image. Note that if you render the cube (with **normal** mode) and then idly change the view or cube parameters and sit back, a little while later the cube will suddenly rotate, even though you haven't clicked on **Apply Orientation**

5.4.2 Rotating the cube

With the sliders **Roll**, **Pitch** and **Yaw** you can rotate the data cube by hand. The coordinate axes are defined such that the X-axis is horizontal, the Y-axis vertical and the Z-axis pointing into the display. The **Roll** rotation is around the X-axis, **Pitch** around the Y-axis and **Yaw** around the Z-axis. Since rotations do not commute, it is important to know that the order of the rotations is **Roll** first, then **Pitch** and then **Yaw**.

To see the cube in the orientation you specify, click (left) on **Compute**. **Reset Orientation** resets the orientation to face-on.

5.5 Making a Movie

The main problem with volume rendering data cubes is that one does not really get a three dimensional perception of the rendered cube. One can display a scene from daily life in a very crude way and still perceive it as a 3D scene. But this apparently is not true for data cubes. The brain does not recognize the object in the image and therefore it does not make a three dimensional perception. One way to get a three dimensional perception of the data is to make a movie in which the data cube is rotated while it is rendered.

For making such a movie, click (left) on **Make Movie**, and the movie control window will pop up. This window allows you to make a series of images where the data cube is rotated in steps. Use the sliders to set the increment in **Roll, Pitch** and **Yaw**, and the number of images you want. **Start Movie** then starts computing this sequence. To save a movie, type in a filename and click **Save Movie**. Once the movie has been generated an animation control window will appear, allowing you to view the movie.

One consideration is the size of the movie. If this is too large (i.e. larger than about 40 Mbyte on **phoenix**), playing the movie will be slow. An estimate of the size of a movie is

$$(x^2 + y^2 + z^2) \cdot N_{\text{frames}} \cdot N_{\text{zoom}}^2$$

where x, y and z are the size of the axes of the data cube in pixels and N_{zoom} the zoom factor. My experience is that rotating the cube in steps of 10 degrees (so the movie is 36 frames) is in general sufficient to get a smooth movie.

It may not be a good idea to make a movie of images with a large **Image Expand Factor**, depending on the sizes of the axes of the cube. You will find that when you play such a movie, it will be slow because it has to swap images in and out. You can always zoom afterwards when you play the movie by resizing the window, although the result may not be as good.

5.6 Filtering your data

Often, there will be problems with the data (too noisy, confusing or bright continuum sources) which make it difficult to see your data. To help, there are a few filtering alogorithms implemented which should help. Click on **Filter** in the main control window and a filtering control panel will pop up.

The following filtering algorithms are available:

- **simple** one of the adaptive filtering algorithms
- **slow median mask** one of the adaptive filtering algorithms
- **fast median mask** one of the adaptive filtering algorithms
- **3D** one of the adaptive filtering algorithms
- **subtract continuum** the continuum-subtraction algorithm

5.6.1 Adaptive Filtering

One problem with volume rendering is that in order to see faint structures in the data cube, one has to set the opacities, intensity transformation and clips such that also the noise becomes bright. As a consequence, the emission is visible only through a thick fog of noise and this noise hides a lot of information.

One solution is to apply adaptive filtering to the data cube: the data is smoothed where the emission is faint and extended (or absent), while the data is left intact if it is stronger or more pointlike. There are several techniques to do this. At the moment two algorithms are implemented, but we are working on other filters.

The filters are based on work of J.-L. Starck, F. Murtagh and A. Bijaoui with a few extentions of our own. They consists of making a wavelet transform of each channel in the cube. This allows to consider the data locally at different resolutions and modify it such that the signal-to-noise is improved.

Note that the wavelet transforms can be 3D and 2D (channel by channel).

The adaptive filtering algorithms available are:

- **Simple** this is a very simple filter. It is a bit crude, but it has the advantage that it is fast, the other filters are quite a bit slower. It gets rid of most of the noise. This filter makes a wavelet transform and in each wavelet plane, it clips the wavelet coefficients that are below the noise level (defined by the parameter filter clip (section 5.6.1)), to zero. This filter is only slighty more complicated than simple clipping the data below a certain level. The fact that it looks at the data with different resolutions makes it more efficient

- **Median mask (fast) & (slow)** the first step of this filter is identical to the **simple** filter. This filter also makes a wavelet transform of the data, and makes a logical mask for each wavelet plane defining where there is significant signal for every resolution, and inverse transforms only the significant wavelet coefficients. Now an iteration cycle begins. It transforms the difference between the original data and the filtered data, retains only the coefficients of the transform of the differnece for those positions and resolutions where there is significant signal in the data using the logical mask. It inverse transforms this masked difference and adds this filtered difference to the filtered data. For the second iteration cycle, it again defines a mask, takes the difference between the original data and this second filtered data, masks the wavelet coefficients and adds the masked difference to the filtered data. etc. etc. The iteration cycle ensures that the wavelet coefficients of the filtered data are identical to those of the original data for those resolutions and positions where there is significant signal.

  An important element of this filter is that it does a median filtering on the logical mask. This makes isolated noise peaks disappear and makes the data look more 'consistent'.

  The difference between the **fast** and **slow** version of this filter is that in the **fast** version 2 iterations are done, compared to 6 iterations in the

slow version. The **fast** filter still leaves some noise blobs in the data, but fewer than the **simple** filter. The **slow** filter gets rid of almost all of them

There are two parameters to set for this filtering:

- **levels** this specifies the number of wavelet planes to be computed. The best values to are either 1 or 2. Using more than 2 planes produces artifacts for the **simple** filter. Usually it is also not necessary to consider more than 2 planes.

  Specifying **Levels** to 0 the original unfiltered data cube is rendered, but with noise clipping applied

- **filter clip** this specifies the clip level to be applied to the wavelet planes, in units of the noise level. Recommended values are 3 for the **simple** filter. For the **median mask** filter use 2 or 3.

5.6.2 Contintuum Subtraction

Often when you observe a spectral-line source you may have a confusing continuum source in the field. Your data reduction package should have a programme to subtract the continuum (e.g. "uvlin" in **Miriad**). The problem with some of these programmes is that in order to compute a fit to the spectrum of the continuum source, you first have to tell the programme which channels contain the spectral-line emission so that it can ignore those channels when computing the fit. The problem is how to find out which channels contain the line emission?

An effective technique is to invert your UV data into a cube *before* subtracting the continuum and then using <**xray**> to render it. Of course, if the continuum source is brighter than your spectral-line source (it usually is), the spectral-line source will be washed out. This is where you can use the **subtract continuum** filter. This will subtract the average flux in a spectral profile, for every point on the sky image. This is a good continuum-subtraction algorithm, to first order: it should filter out more than 90% of the continuum source.

Once you have your first order continuum-subtracted cube, you can use <**xray**> to find the spectral-line emission. In particular, you should find the 3D slicing window very handy, since it will give you the 3D co-ordinates of a point. Be warned that **Karma** programmes count from 0, whereas most astronomical reduction packages count from 1. If you take a channel index from **Karma** and put it into **Miriad**, remember to add the value 1.

Now that you have identified the spectral-line emission, you can run a programme like "uvlin" to do a better job of subtracting the continuum.

5.6.3 Other Controls

To apply the filtering. click on **Do Filter**. The cube is rendered automatically.

To get back the original data without any filtering, click on **Undo Filter**.

You may also **Save** the filtered cube for later use.

5.7 Hot Gas Substances control panel

See Figure 5.2 for a screen snapshot.

This controls the settings for the Hot Gas Substances (section 5.1.1). You should see a window with a histogram of the data values in the cube (actually the logarithm of the distribution), and in the bottom part you will see space for the colours of the substances. Each colour has a horizontal line on it. This is an indicator for the opacity of the substance.

To change the colour of a substances you have to click (left) on the space on the bottom of the window that is reserved for the substance. This selects the substance. To change the colour, move the mouse in the upper part of the window and drag (middle, not left!!) the mouse around until you have the colour you want. The saturation of the colour can be changed by moving the mouse around while you drag with the right button. To change the opacity, put the mouse on the substance and drag with the middle mouse button vertically.

To change the value ranges for the substances, drag (left) the mouse in the upper part of the window. In the upper part, there are a few symbols that help to orient yourself. The + is the setting for the value ranges, the **O** the setting for the substance you have selected, and the horizontal line on the left of the window is the saturation of that substance. With this window, one is only able to change the value ranges in a way similar to a normal colour table control.

To use the settings you have made, click (left) on **Apply** and a new image will be calculated.

The button **Save** brings up a window to save substances.

To load a substance table click on **Load** and select your file.

5.8 Hot Gas Mono control panel

See Figure 5.3 for a screen snapshot.

The control window for this allows you to set the opacity law of the Hot Gas Mono (section 5.1.1) shader, as well as the exponent for the intensity transformation. One can also set the range of voxel values that are set to blank.

The following controls are available:

- **Close** this will close the window
- **Compute** this will render the cube
- **Intensity Alpha** this controls the value of β in equation 5.3
- **Opacity Alpha** this controls the value of α in equation 5.2
- **Blank Start Slider** this controls the start of a region of data values which will be blanked
- **Blank End Slider** this controls the end of a region of data values which will be blanked. If this value is greater than that of **Blank Start Slider**

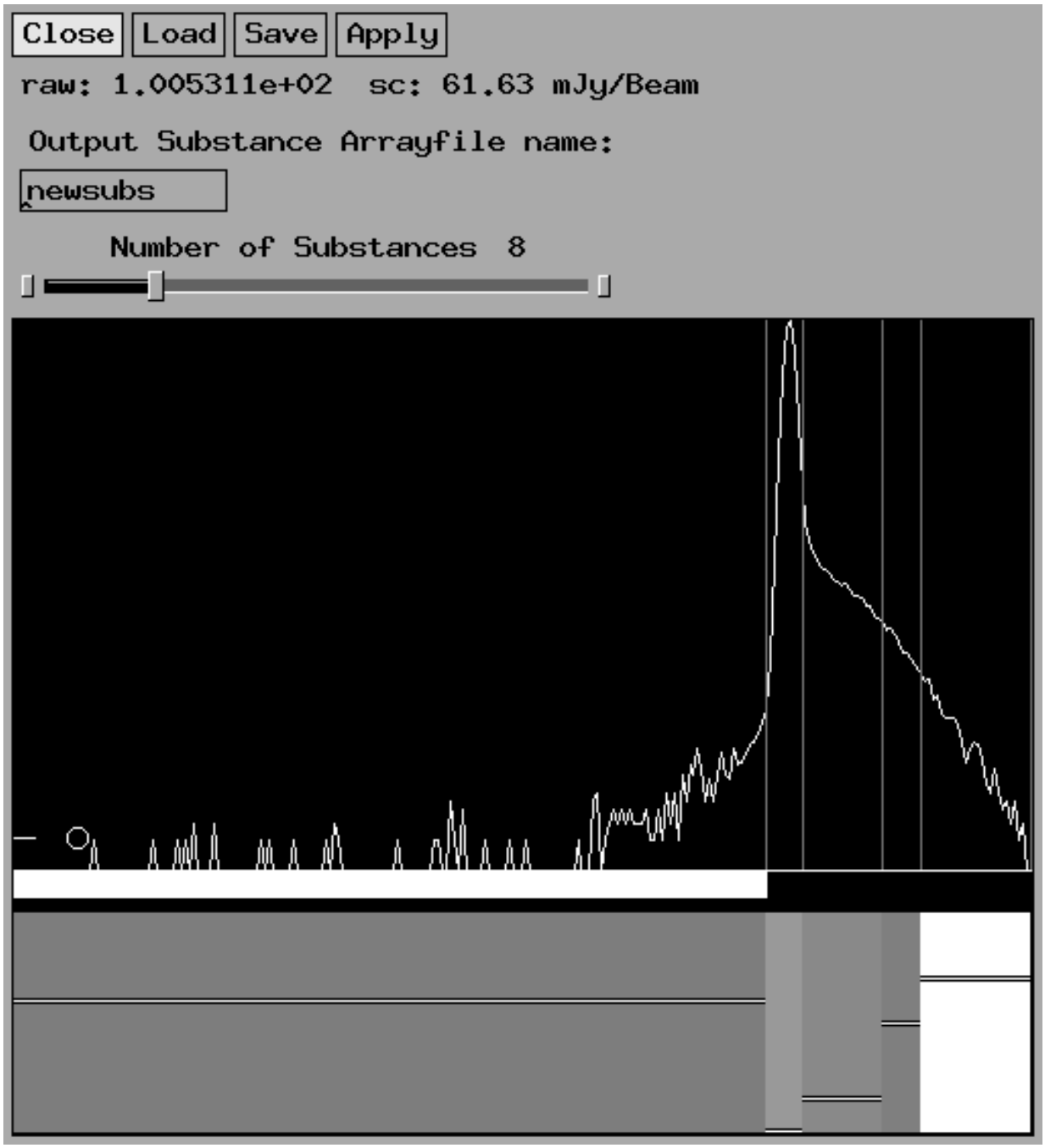

Figure 5.2: Screen snapshot of hot gas substances control panel

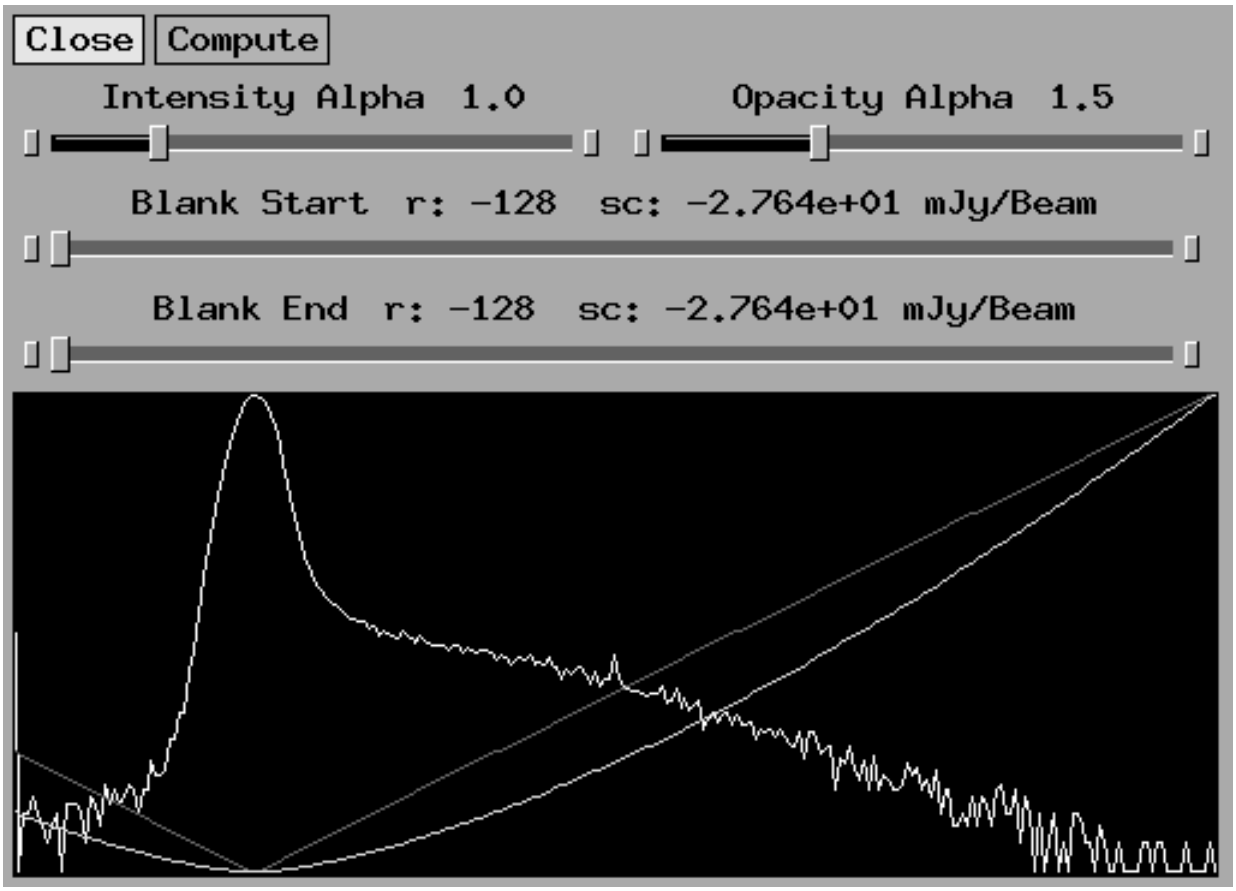

Figure 5.3: Screen snapshot of hot gas mono control panel

then some range of data will be blanked. The blanking is implemented by setting the opacities for that range to 0

In the display canvas under the controls you will see a log-histogram of your data (in white) as well as a red curve showing the transfer function between data value and intensity (see equation 5.3) and a yellow curve showing the transfer function between data value and intensity (see equation 5.2. The opacity curve also reflects the blanked range of values.

5.9 Hot Gas Continuous control panel

This shader is a cross between **Hot Gas Substances** and **Hot Gas Mono**. It has a similar control as the **Hot Gas Mono** shader in <**xray**>. Also here the exponent of the opacity law can be set, as well as the values range for valid data. The difference is that one can set the connection between voxel value and colour using a colourtable-like interface.

5.10 Hot Gas Three Colour control panel

This shader is an extension of the **Hot Gas Continuous** shader. It allows you to divide the data range into three regions, with independant opacity and colour transfer functions for each region. This shader is a little esoteric, and is designed for rendering data with two or three distinct populations of data values. This kind of data does not occur in radio astronomy. You may generate this data by combining two datacubes together. This shader would allow you to render one dataset in one colour and another dataset in another colour.

Chapter 6

kpvslice: interactive position-velocity slicing

The <**kpvslice**> programme allows you to use a two-dimensional image to define a slice through a three-dimensional data cube. The image and cube must have two dimensions in common, and the coordinate range of those dimensions must (partially) overlap, but the axes need not have the same reference point, orientation or scale.

A typical use is to examine the position-velocity diagram for a slice drawn along the major axis of a rotating galaxy. The slice is drawn on the total column density or velocity field map, and the corresponding position-velocity diagram will appear in the other window. The slice can be moved around interactively for quick comparison of different regions.

The fancy colour front cover of this document contains a screen snapshot of this programme.

6.1 How to use it

You need a cube (naturally) and a sky map image. The image is usually taken to be the total-intensity map of the cube, but it could just as easily be the velocity-field or an optical image. As long as the cube and the image have a proper co-ordinate system for RA and DEC, you're in business. To load the cube, press the **Cubes** button in the left window and select a cube file. Press the **Images** button to load the image file. The image is displayed in the left window. If you then click (middle) in the image and drag the mouse (still keeping the button pressed), you will see the position-velicity slice in the right window being updated before your very eyes. Back in the image window a line is drawn over the image to show where the slice is. The horizontal axis in the right window shows position and the vertical axis shows velocity. If the machine you are running this on is rather old and slow, you can disable **Auto Update** and the position-velocity image will only be computed once you release the middle mouse button. The cursor position (in RA and DEC) is printed to the command shell window whenever the middle button is pressed or released. This tells you the endpoints of the slice. Also, the centre of the slice and it's

position angle are also printed.

If you click with the right mouse button near the centre of the slice, you can then move the slice centre around whilst the length and position angle is kept constant. If instead you clicked the right button near either end of the slice, you can change the position angle while the length and centre are kept fixed.

When moving the cursor in the slice window, the coordinates of the cursor are displayed in several ways. The "natural" coordinates of offset (from the slice centre) and third-axis coordinate are shown in the second line. The image coordinates corresponding to the offset are shown in the third line. The position of the cursor along the slice is also shown by a small circle in the image window.

If you compute the moments of the cube with <**kpvslice**>, both are held in memory, so you may switch between them easily. The values in the first moment image along the slice are plotted as a red trace in the slice window. Note that no smoothing is done before the clipping level is applied, so they do not do as good a job of representing weak signals as the standard MOMNT task in AIPS. This may be fixed in later versions of the program.

If the values in the two-dimensional image have the same units as the cube's third coordinate axis, the values are extracted and displayed as a green trace in the slice window. In practice this usually happens when one draws a slice on a velocity field image; the velocities from the image are drawn as a green line on the position-velocity diagram. The position angle is defined as the angle in the corner of a spherical triangle.

6.1.1 Viewing Profiles

You can also view a profile (spectrum) of your cube profiles (section 2.11). Just press the **Z-Profile** button and move the mouse over the image window.

If you wish to see a profile *along the slice across the image*, press the **I-Profile** button and another profile window will appear. Just don't confuse the two different profile windows.

6.2 Extracting Slices From An Image

<**kpvslice**> can also extract a (one dimensional) slice from an image. In this case, you do not have to load a cube, only the image is needed. Click on the **I-Profile** button and draw your slice. The profile window will show the values in the image which lie along the slice locus.

6.3 Command-line Options

The following command-line options are supported by <**kpvslice**>:

- `-trace_internal_colour <colour>` the colour specified by `colour` is used when displaying the trace taken from the internally computed 1st moment map

- `-trace_external_colour <colour>` the colour specified by `colour` is used when displaying the trace taken from the externally computed (loaded) 1st moment map

6.4 Algorithmic Details

6.4.1 How the Slice is Computed

The slices are interpolated using a weighted average of the nearest four pixels. The weighting is inversely proportional to the distance of the point in the slice from the centre of the neighbour pixel.

So the "thickness" is basically one spatial pixel, but the width of the region that the interpolation is made over varies depending on the angle of the slice with respect to the pixel grid. It is least (3 pixels wide) when the slice is along the x or y axis of the pixel grid, and greatest (sqrt(2)*3) when the slice is at 45° to the pixel grid.

The interpolation is made only with respect to the pixel grid, so the above numbers are only strictly true if the x and y directions have the same spatial scales. If the arcsec/pixel is different in the x and y directions, the width of the slice in arcsec will change from differently as the angle of the slice w.r.t the pixel grid changes.

```
++++.+++++++
+++++.++++++
++++++x+++++   'x' is the current pixel, '*' are the neighbours
+++++++.++++   '.' is the other pixels in the slice.
++++++++.+++
+++++++++.++
```

Chapter 7

kslice_3d: slicing data cubes

The programme <**kslice_3d**> is the **Karma** version of an idea of François Viallefond (a program called **3view**). Also **GIPSY** has an implementation of something very similar. Because **Karma** is much more designed for this kind of work, <**kslice_3d**> is much faster than the other two versions.

The program shows 3 images in a window, being the data cube sliced along the principle planes. The top left image is a XY image (so for a radio cube this would be a channel), the bottom image is XZ (so normally α, v) and the top right ZY (v, δ). With the mouse one can interactively change the planes that are shown. If the mouse is in the XY image, the other two images are the XZ and the ZY image corresponding the the *(x,y)* position of the cursor. Alternatively, if the mouse is in one of the two other images, the XY image shown is the image corresponding to the Z-coordinate of the mouse.

<**kslice_3d**> is fast enough (contrary to **3view** and **GIPSY**) that if you move the mouse while keeping the left button pressed, the images are updated real time.

Note that the volume rendering programme xray (chapter 5) will automatically display a <**kslice_3d**> window so that you can slice through the cube as well as volume render it.

There are only a few buttons in <**kslice_3d**>:

7.1 Precompute

If a cube is loaded, the images to display are not computed automatically. If next you move the mouse, <**kslice_3d**> will have to compute the image that is has to display in real time. This makes things a bit slow sometimes. Clicking **Precompute** will make <**kslice_3d**> compute all the images it can display. This takes a few seconds, but it makes the update of the images very fast.

Note that to run smoothly, <**kslice_3d**> requires quite a bit of memory, certainly if you expand the images. If this gives problems, try running it sitting in front of **phoenix**. Note that the memory requirement is on the machine you are sitting in front of, not the machine that you are running the programme on (although for best performance they should be the same).

7.2 Magnifications

There are three sliders that control the size of the three images. If you expand too much, <**kslice_3d**> will not display anything (i.e. you will see a blank window). Just reduce your expansion factor appropriately until the display re-appears.

If you want to make the images larger than the window would allow, you can resize the window (drag on the bottom right corner) and after this increase the magnifications.

Chapter 8

kshell: looking for expanding shells

This tool allows you to browse your datacube searching for expanding shells. Since most shells are mostly radially symmetric, it is reasonable to azimuthally average concentric annuli around a designated centre. What this means is that the programme allows you to define a centre position on the sky and a maximum radius. It will then step through a range of radii (from zero to the maximum), and at each radius will average the values in one velocity map around the circumference of a circle (hence "azimuthal average"). This process is repeated for each velocity map, and the result is an image with one axis corresponding to radius and the other velocity.

If you are looking at an expanding shell, for each velocity map the average radius of the emission from the centre is approximately constant. As you progress from one velocity extreme to the other, the radius will change from zero, pass through a maximum at the central velocity and go back down to zero. So, in the computed radius-velocity map, you should see half an ellipse (the "signature"). If you have slightly missed the centre of the expansion, you will see a double ellipse, so this is a good way of finding the centre. If you are not looking at an expanding shell at all, you just see a mess, so this is also a good way for speculatively searching for shells. The lower the expansion velocity, the more squashed the ellipse becomes (vertically, of course). In fact, if you had a static shell, you would just see a straight line. The power of this tool is that it is both a fast way of browsing for expanding shells (because of their charactistic signature) and that it is insensitive to fragmentation of shells.

To drive this programme, you need a cube file (load with the **Cubes** button) and an image file (load with the **Images** button). Actually, the image file is optional: you can compute an image on the fly from the cube by selecting say the **0th moment** option from the **Image Mode** menu, and adjusting the parameters as needed and pressing **Apply**. The reason you need an image is so that you can search for a possible centre of an expanding shell. The image is displayed in the left window. The image and cube files do not have to lie on the same co-ordinate grid.

Once you have loaded your files, you simply click with the middle mouse

button (don't let go yet!) to define the centre of the shell, "drag" the mouse so that you see the expanding circle, and release the middle button when you think the radius is sufficiently large such that the entire shell is enclosed. Don't make it larger than is really necessary, since it will take longer for the programme to compute the radius-velocity image. After a short time (fraction of a second to a few seconds for typical cubes), you will see the computed radius-velocity image. Do you see a nice ellipse in the right window? If you move the mouse into the right window, you should see the current position (in sensible units) as well as the averaged intensity at that point. You can use the **Export** menu above the image to write the image (with co-ordinates) in a variety of formats (choose one of the "(whole dataset)" options).

A small tip for people looking at data with very high dynamic range (such as OHIR masers): set intensity scale for the right window to logarithmic (see the zooming (section 2.6) section), otherwise you will only see a few bright blobs.

Chapter 9

koords: interactive co-ordinate fitting

This tool allows you to quickly place a co-ordinate system (compute a plate solution) onto an image which does not have one. This is very useful if you have, say, a CCD image which does not have a co-ordinate system on it. All you need is an image of a similar piece of sky which does have a co-ordinate system (such as an image from the Digital Sky Survey). The tool handles flipping and rotation correctly. This tool is much faster to use than the various command-line driven tools currently available (usually less than a minute).

After you start the tool, you will get two windows. The left window should be loaded with the reference image (the image which has a co-ordinate system), by pressing **Ref. Images**. The right window should be loaded with your target image (the image without the co-ordinate system), by pressing **Target Images**.

The programme should also work with a photographic negative. The picking algorithm is automatically modified to look for a trough rather than a peak.

Once you have loaded your images, you are now ready to define matching pairs of points (stars). To do this, you must first click (middle) in the left window, near a star, and then click (middle) in the right window near the corresponding star. You then go back to the left window and define another star, and click on it's mate in the right window. You must define at least three pairs of stars.

As you select a star, a green circle is drawn around the fitted centre of the star. If at any time you miss the star, you can just click again (only before going to the other window). If you press **Undo Pair**, a pair of stars will be removed from the list. As long as some visible part of the star is within the circle cursor (also known as the "capture region") when you click (middle), the tool will find the centroid of the star.

You might find it helpful to click the **Magnifier** button to see the magnifier window. You will see a crosshair which is centred on the current mouse position in the main window. You will also see a green circle which shows the "capture region": if any part of the star lies within that circle, when you click (middle) the centroid of the star will be picked. You can adjust the size of the capture region by clicking with the left mouse button in the magnifier window. You

will also see a dashed green circle which is slightly larger than the solid green circle. The area between the two circles is called the background annulus. The magnifier will also show small red crosshairs at the exact centroid positions of selected stars.

Once you have selected all your star pairs, you can click on **Compute Co-ordinate System** and the tool solves for a co-ordinate system. In your command-line window a report of the solution is displayed. If you move your mouse around the right window, you will see the proper world co-ordinates of the pixels. Once you are happy with the co-ordinate solution, you can press the **Apply Co-ordinate System** button, and then save the image (with the new co-ordinate system) using the **Export** menu. The coordinate system is written in the header using the standard (CTYPEn, CRVALn, CDELTn, CRPIXn) notation.

The coordinate system is computed under the assumption that the image of the sky has been projected by the telescope optics in a particular way. The projection geometry is assumed to be the Rectangular projection (denoted by the standard projection name "ARC"), which is used by Schmidt telescopes. This may not be always be the appropriate projection to use, but the errors should only become noticeable for fields larger than 0.5 degree.

9.1 Centring Algorithm

The following steps are taken by the picking algorithm:

- find the brightest pixel within the circle cursor ("capture region")
- proceed from that point by "walking up the hill", searching for a neighbouring pixel which is of a greater value. It continues searching until it finds a pixel with no neighbours with a greater value (i.e. a local maximum). This is usually the peak pixel
- find the half-intensity size of the star. To do this it needs an estimate of the background value. The background is computed by finding the median value inside the background annulus
- computes the centroid position by using a two-dimensional weighted mean algorithm. The centroid is computed out to the half-intensity size.

Note that the centring algorithm will not work with heavily clipped images. This is because there is no real peak to work with, rather there is only a plateau. In these cases, the centring algorithm is effectively bypassed and the position you selected will be used for the star position. This allows you to position by eye. Please note that heavily clipped images have lost information, and hence the accuracy of star positions will be limited.

Chapter 10

kpolar: regridding to polar co-ordinates

This tool allows you to regrid an image using a rectangular to polar transformation. It allows you to see the azimuthal variation of intensity around a given centre. This can be useful for looking at the (optical) structure of galaxies, or doing close inspection of a dirty beam (quite useful when doing radio astronomy).

To use this programme, all you need is an image file, which you load with the Files button. Once your image is loaded, you need to define an ellipse:

- Press the right mouse button to define the centre, then drag the mouse while keeping the button pressed, to define the size and shape of the ellipse
- If you press and release the 'r' key (still keeping the mouse button down) this will toggle on 'rotation mode' so that now when you drag the mouse the ellipse will rotate. Press 'r' again to turn off rotation mode so you can continue changing the size and shape of the ellipse
- Once you are happy with your ellipse, release the mouse button and the polar image will be computed and displayed in the Polar Window. The vertical axis is azimuth (angle) and the horizontal axis is radius

Please note: this is an experimental programme that (at present) does this transformation in a simpleminded way. It works by binning the input image in radius and azimuth directions using a set of concentric elliptical annuli. Each annulus has a width of one pixel. Some points that you may wish to consider when using this task are listed below:

- The programme does simple nearest-neighbour interpolation to produce the output intensities; the absolute flux scale will probably not be perfectly preserved
- No 'world' (i.e. angle in degrees, radius in arcseconds) co-ordinate system is attached to the image. No allowance is made for images which have different scales on their x and y axes

- The centre position of the ellipses is not fitted in any way, it is exactly where you place it

Feedback on improvements to this task (including code!) is welcome.

Chapter 11

Superimposing Images

The <**kvis**> (chapter 3) programme allows you to display different images on top of each other, while each image has a different colour coding. This can be used to display amplitude and phase together in one image, or compare different channels from a data cube by superimposing them.

These visualisation techniques require a 24-bit display.

11.1 Image and Contours

Note that there is another way to superimpose images which involves drawing one image as a greyscale or false-colour image and the other is drawn over the top as contours. This can be done with <**kvis**> (chapter 3), and does not require a 24-bit display.

You may also use the <**send-contours**> (section 12.5.10) programme to draw contours over any image. This programme is a command-line tool, however, and is not normally used since the the GUI tools are much easier to use.

11.2 Hue and Intensity

With the program <**kvis**> you can superimpose two images, where one image is taken to control the intensity and the other the colour. This can be used to display amplitude and phase together in one image, or line integral and a velocity field.

To display the images run

`kvis`

First load **File1** and **File2**. You should now see the combined images. The sliders **Start Hue** and **Stop Hue** manipulate the colour of the image, the sliders **Minimum Intensity** and **Saturation** the intensity.

By default, **File1** will control the intensity, and **File2** the hue. You can swap this by choosing "Phase, Amplitude" from the **RAW DATA FORM** menu. If you select "Real, Imaginary" then a rectangular to polar transformation of the data is performed, where **File1** is real image and **File2** is the imaginary image. You can swap the sense of this by choosing "Imaginary, Real". After

the rectangular to polar transformation, the amplitude value will control the intensity and the phase value will control the hue.

11.3 Intensity and Intensity

Another possibility is to overlay images so that all images are coded as intensity but in a different colour. This you can use to overlay two or three different channels of a data cube and inspect subtle differences. The <**kvis**> (chapter 3) programme can be used to do this.

Chapter 12

Command-line Tools

Karma has a number of command-line tools to manipulate data files. There are tools for converting from foreign data formats to Karma format, tools to convert from Karma to foreign formats as well as tools to manipulate data files in various ways (i.e. regridding).

12.1 Common Interface

All of the command-line tools use the <**panel**> package in the Karma library to implement a common "feel". This package defines a set of *panel items* which are simply parameters you can set or actions you can take. The tools may be started in interactive mode by giving no arguments to the tool, or they may be started in batch mode, giving a set of arguments to the tool. Once the tool has completed processing, it exits.

12.2 Interactive Mode

To start a tool in interactive mode, just type in its name at the shell prompt. You should then see a prompt with the name of the tool followed by a "> ". In this discussion it is assumed that you will press the return key after typing something in.

Probably the first thing you will want to do is press `?` to see a list of parameters and actions. On the left hand side you will see all the parameter and action names. For parameters, you will then see zero or more values after the parameter name. For actions, you will not see any values. Finally, you will see a "#" character usually followed by a short description of the parameter or action.

To specify a parameter or action, type in its name, followed by any arguments required. You can type shorter, non-ambiguous name if you wish. You can only specify a single parameter or action on one line.

12.2.1 Actions

Actions are things that happen when you type their name. Some actions require no further arguments (such as the `exit` action), while others may require any number of arguments. The brief description for the action should indicate what arguments are needed.

12.2.2 Scalar Parameters

These are parameters which take as argument a single scalar value or string. After typing in the name of the parameter, followed by a space, you must type in the value or string.

Strings do not need to be enclosed in quotes *unless they contain a space*. You may use either single or double quotes to enclose a string. Naturally, you must use the same type of quote character to close the string as you used to open it. To encase a quote character inside a string, just type it if you used the other type of quote character to enclose the string, or type the quote character twice if the string is enclosed by the same type of quote.

Scalar values may be integer or floating point, and may be real or complex depending on the type of the parameter. If it is a complex value, the real and imaginary components must be separated by a space.

12.2.3 Vector Parameters

A vector parameter is an array of scalar parameters. The parameter is displayed with a trailing `[n..m]` after its name. The value n is the minimum number of elements in the array and the value m is the maximum number of elements in the array.

After typing in the name of the parameter, you must type in the array of scalar elements. Elements in the array must be separated by a space. You must type in at least the minimum number of elements, otherwise your input is rejected. If you type in more than the maximum number of elements allowed, the excess elements are discarded.

If you wish to append elements to an existing array, simply type `+` followed by the parameter name and then followed by the scalar elements.

If you want to empty out the contents of an array, just type the name of the parameter without any elements after it. This does not work if the minimum number of elements is greater than 1.

12.2.4 Choice Parameters

A choice parameter allows you to choose from a selection of possibilities. If you wish to see what all the possibilities are, type `??` and you will see a list of choices for each choice parameter.

After typing in the name of the parameter, type in your choice. You can type the minimum non-ambiguous choice string if you wish.

12.2.5 "Eternal" Actions

There are a few actions which are common to all tools, which allow you to exit the "panel" (exit the tool), with or without saving parameters, and others which manipulate the communications infrastructure.

12.2.6 Other Arguments

Most tools define some other arguments which are not parameters or actions. They are usually filenames to process, although they can be other things too (like dimension to filter along). An unlimited number of other arguments may be typed in one line.

Some tools will show the other arguments by printing "General usage: " followed by the arguments.

12.2.7 Exiting Interactive Mode

The `abort` action will abort the tool without saving any parameters. The `exit` and `quit` actions will exit the tool after saving parameters. Pressing control-D has the same effect as typing `exit`.

12.3 Batch Mode

Another term for "batch mode" is "shell mode", which just means that the tool is started from the shell, with arguments, and exits when it has completed. From the shell, you can pass in any number of parameters and actions. The limitation in interactive mode of only typing a single parameter or action is removed.

12.3.1 Passing in Parameters and Actions

You simply need to place the - character in front of the name of the parameter or action (with no intervening space). This marks the start of a new parameter or action. All subsequent shell arguments are treated arguments for the action or parameter, until the the list is broken. A new parameter or action name with a preceeding - will break the list, as will a solitary - character. The solitary - character marks the start of remaining "other arguments", after which there can be no further action or parameters.

12.3.2 Passing in Strings

If a string contains spaces, you should enclose it in quotes on the shell command line. If you are familiar with how most shells work, you will know that it removes enclosing quotes before passing them onto the programme. Karma deals with this by checking if a shell argument has a space in it, and if it does, it puts a pair of double quotes around the argument again. This should be transparent to you unless you start doing esoteric things with embedded quotes.

12.3.3 Passing in Negative Values

Passing in a negative number is not a problem, because this case is trapped (note all parameter or action names must begin with an alphabetic character). If you wish to pass in a string with a preceeding - then you should type in -- at the shell command line. The double minus will be reduced to a single minus before being used.

12.3.4 Passing in Other Arguments

If you do not pass in any parameters or actions, then you can just type in the other arguments as you would in interactive mode. However, if you preceed the other arguments with parameters or actions, you need to type a solitary - between the last argument to the last parameter or action and the first "other argument". This solitary - should be surrounded by at least one space on either side.

12.4 Saving and Restoring Parameters

Each of the command-line tools will look for a file called ".**name**.defaults", where *name* is the name of the programme. This file (the "defaults" file) contains saved parameters which will be read by the tool as it starts. First the current working directory is searched, and then each parent directory in turn, until either a defaults file is found, or the root directory is reached. If still a defaults file is not found, then the tool will look in the `.karma/module-defaults` directory under the users home directory, and then finally it looks in the `$KARMABASE/defaults` directory. If no defaults file can be found then internally hard-coded defaults are used.

Upon normal exit, each tool will write its defaults file in the current working directory.

12.5 List of Tasks

12.5.1 chlen

This tool will change the length of a dimension in a Karma file. You can increase the length of a dimension by zero-padding or resampling and decrease the length by truncating, windowing or resampling. The `option` parameter controls whether a dimension is resampled, truncated or padded. The resampling option is the same as a one-dimensional regridding.

If you have chosen to resample a dimension, then the `resample_option` parameter controls whether a nearest neighbour copy or linear interpolation is performed.

The general usage for the programme is:

`chlen infile dimension_name`

where `infile` is the input data file and `dimension_name` is the name of the dimension to change. Note that the input file may be of any format.

Examples

If you had a spectral-line datacube and you wanted to resample the velocity axis to, say, 600 data pixels, the following options would be needed:

```
resample_option     linear_interpolation
option              resample
num_coordinates     600
```

and then you would type something like:

```
mycube VELO-HEL
```

12.5.2 images2karma

This tool will convert multiple images of any format into a Karma cube. This tool is not implemented with the <**panel**> and hence does not have settable parameters or an interactive mode. The usage for this programme is:

```
images2karma infiles [...]  outfile
```

The input images must be of the same shape, size and type, but do not need to be of the same file format.

12.5.3 karma2ppm

This tool will convert a Karma image or cube file into one or more **PPM** (Portable Pixel Map) files. The **PPM** format is a 24bit TrueColour image format. A **Karma** image or cube (movie) may be either TrueColour or PseudoColour with an associated colourmap in the file.

If the input image or cube is greyscale (i.e. no colour information is available), then it is still possible to write out a **PPM** file if you supply a colourmap file, previously saved with a <**Cmapwinpopup**> widget (section 2.2). You specify the colourmap file using the `cmapfile` parameter.

The general usage for this programme is:

```
karma2ppm infile outfile
```

where `infile` is the input **Karma** file and `outfile` is the output **PPM** file. If the input file is a 3-dimensional array, a sequence of numbered output files will be written.

Writing other Image Formats

You can also use <**karma2ppm**> to write out images in other popular formats, such as **GIF**, **TIFF**, **Targa TrueVision** and so on. To do this requires that you have the NetPBM tools installed. These tools are freely available and very popular, as they allow conversion between many different image formats. To use this facility, you need to set the `converter` and `extension` parameters. The `converter` is the name of the programme which converts PPM/PNM images to the image format you desire. The `extension` is the conventional filename extension for that image format.

For example, if you wanted to convert a **Karma** image or cube to a **TIFF** image or sequence of images, you would need the following settings:

```
converter pnmtotiff
extension tiff
```

On the other hand, if you wanted to generate **GIF** images, you would need the following settings:

```
converter ppmtogif
extension gif
```

Notice how the the **TIFF** converter seems to require **PNM** files whereas the **GIF** converter seems to require **PPM** files. This is not a problem, since **PNM** (Portable aNy Map) is the generic format and **PPM** (Portable Pixel Map) is a specific subset of **PNM**.

Changing the Output Image Size

If you wish to write out images of a different size from your input image/cube, then you will need to use the `out_width` and `out_height` parameters. If your input is smaller than this, the image will be blank-filled and centred onto the desired output image size. If the input is larger than this, it will be centred and truncated. No pixel averaging or replication is performed.

12.5.4 kdecimate

This tool will decimate a cube, averaging a specified number of values along an axis to produce a single value. This decimation is performed along all 3 axes. This tool provides a simplified version of the <**LoadAndDecimate**> widget (section 2.1.1). The only reason to use it is if you are developing batch mode scripts to process large number of datacubes.

The general usage for this programme is:

```
kdecimate infile outfile
```

12.5.5 kminmax

This tool will show the minimum and maximum value in a datafile. It will also display the number of blank values (TOOBIGs) and the number of illegal values. Note that illegal values should never occur, and are the result of an incorrectly written file. Illegal values will cause problems for Karma programmes (such as a segmentation fault or other crash). Hence this programme is an excellent diagnostic to determine if the data you are giving to a programme is legal.

This programme supports many data formats. The general usage for this programme is:

```
kminmax infile dataname
```

where `infile` is the file to load and `dataname` is the name of the data elements to process (this is usually the same as the unit name, for example "JY/BEAM" or "HZ").

12.5.6 kprinthead

This tool will show the header in a datafile. It supports many data formats. The general usage for this programme is:

`kprinthead infile`
where `infile` is the file to print the header for.

12.5.7 kregrid

This tool will regrid one or more input images or a cube onto an output image or cube. It is possible to use this tool to mosaic several images or cubes together onto a new grid. The general usage for this programme is:

`kregrid infile`

where `infile` is the input file to regrid or is a file containing a list of files to regrid and mosaic. The output filename will be "kregrid_`infile`.kf"

12.5.8 krotate

This tool will rotate (by regridding) an image or cube such that the longitude axis (Right Ascension or Galactic Longitude) is aligned with the horizontal axis. This is often used to rotate images to a more convenient or familiar orientation (e.g. images from the Hubble Space Telescope). The general usage for this programme is:

`krotate infile outfile`

where `infile` is the input file to rotate and `outfile` is the name of the output (rotated) file you wish to create. If you specify an extension of `.fits` then a FITS file will be written, otherwise a Karma file will be written. You may use the `projection` option to control the output projection (the default is to use the same projection as the input).

12.5.9 ksend

This tool allows you to send commands to various GUI tools. The commands include loading a file, writing PostScript to a file, setting display ranges and so on. This can be useful for batch processing many different datasets using the same control settings.

If you wanted to load the files `mom0` and `mom1` into the <**kvis**> (chapter 3) programme, displaying `mom0` as an image and `mom1` as contours and then write PostScript to the `ps` file, you would first need to start <**kvis**> and set up the controls appropriately to display contours. Then you can use the command:

```
ksend -add_connection unix kvis multi_array -load ARRAY:FILE0 mom0
-load ARRAY:FILE0 mom1 -print ps
```

A variation of this would be to generate a Portable Pixel Map (**PPM**) file with name `fred.ppm`:

```
ksend -add_connection unix kvis multi_array -load ARRAY:FILE0 mom0
-load ARRAY:FILE0 mom1 -ppm fred.ppm
```

You can use the above command many times, each time specifying different files, and hence obtain batch mode operation.

If you wanted to set the intensity display range to 1.0 and 1.5 for dataset 1 you would do something like this:

```
ksend -add_connection unix kvis multi_array
-rclip "Dataset 1" 1.0 1.5
```

Alternatively, if you wanted to set the intensity display range to 95% of the data, you would do something like this:

```
ksend -add_connection unix kvis multi_array
-hclip "Dataset 1" 0.95
```

12.5.10 send-contours

This tool allows you to load an image and compute contour overlays which may then be sent to any GUI programme for display. This facility is only useful when you want to add contours to a tool which doesn't already have built-in support.

For each extra dataset you wish to display as contours, you need to run a copy of the <**send-contours**> programme. To start the programme, type:

```
send-contours
```

Once in the tool, you will need to specify the desired contours levels. This interface is quite primitive, requiring you to enter the actual values for each level. For example, if you have a 1st moment map with interesting contours at 1.419 GHz and 1.420 GHz, you would type the following:

```
contour_levels 1.419e9 1.420e9
```

You then need to specify the input file containing the image data:

```
in_file infile
```

where `infile` is the input filename. Next you need to make a connection from the tool to a GUI tool. Typically, this will be the <**kvis**> programme which is running on the same machine. For example, type:

```
add_connection unix 7605 2D_overlay
```

The number `7605` is the Karma port number of the GUI tool, and is displayed in the title bar of the GUI tool. Other GUI tools will have different port numbers. Once you have done all this setup, type `go` and you should see your new contours. The normal GUI functions for zooming are supported. You can use the `empty_list` command to remove the contours and clean up the display.

Appendix A

Chromatic Aberration

Radio telescope interferometers exhibit chromatic aberration effects. This is because the resolution of the instrument changes with wavelength. For low-bandwidth spectral-line observations, these effects are minor and can usually be ignored. However, for high-bandwidth observations (say 128 MHz) these effects can pose a problem.

Reduction packages such as AIPS and AIPS++ will produce a beam pattern image for each frequency channel. This allows these packages to compute a spectral-line cube which has a pixel size (cellsize) independent of frequency. The chromatic aberration manifests itself in the dirty cubes as a changing antenna pattern. In other words, if you play a movie of the channel maps, you will see the grating ring pattern change. Once the cube has been properly cleaned, you should no longer see grating rings and hence the cromatic aberration has been effectively corrected.

Reduction packages such as Miriad will only produce a single beam pattern for a nominal reference frequency channel (this approach has the advantage of saving disc space). Because of this, Miriad produces cubes where the pixel size changes with frequency. You can see this when you play a movie of the channel maps: an object distant from the phase centre will appear to move, although the grating ring pattern will not change: it only shifts with the source object. This shifting of an object with frequency is of itself not a serious problem, however it is important that the co-ordinate system takes the variable pixel size into account, so that it tracks the moving object correctly.

The scaling of the cell size with frequency is accounted for in the coordinate handling within Miriad. The correction is quite small for narrow bandwidths (e.g. 4 MHz), but quite noticable for cubes with 128 MHz bandwidths at 20cm (a 7 pixel shift is not unheard of).

The FITS convention assumes that pixel sizes are constant. The visualisation software also makes this assumption.

Since June 1997 **Karma** has taken account of this chromatic aberration by adding the new "CELLSCAL" keyword as it reads in Miriad cubes. This keyword is used in the astronomical world co-ordinate package to correctly track variable pixel sizes. At the same time Miriad was modified to add the "CELLSCAL" keyword when it writes FITS files. Prior to this, old FITS cubes

written by Miriad did not have this keyword, and hence violated the FITS convention. If you have such old cubes, you should either regenerate the FITS from the original Miriad cube or else manually edit the FITS header by adding a line such as:

```
CELLSCAL = '1/F'
```

If you use a current version of Miriad and Karma then the co-ordinate system will correctly handle the chromatic aberration effects. Note that the source object will *still* appear to move as you play a movie of channel maps. I have been asked to change this, but this would require the visualisation software to regrid the cube. Apart from the computational cost involved, it is also not the correct approach, as subtle artefacts can be introduced. The correct approach is to produce a cube in Miriad which does not have a variable pixel size. This is possible to do (although a little awkward). It involves computing each channel image independently. This will mean that Miriad computes a beam pattern image for each channel. If you manually specify the pixel size, each computed image will have the same pixel size. After this you clean each image and collect them into a single cube. You can make this procedure easier by writing a script. Once you have produced such a cube, check the header to ensure that you don't have a "CELLSCAL" keyword with value "1/F" (Miriad should only do this when a 3-dimensional "invert" is performed).

Appendix B

Setting up

B.1 Environments needed

The visualisation software is written using the **Karma** package (a library for networked data processing and visualisation). In order to be able to use this library, a number of environment variables have to be set. There are two ways of doing this:

1) is to type:

```
source /usr/local/karma/.login
```

You can also include this in your *.login* file.

2) if you are using the standard .login file, select **Karma**

B.2 Operating Systems

All the programs here are available for Unix environments. I have ready access to the following platforms:

- sparc_SunOS
- sparc_Solaris
- i386_Linux
- mips2_IRIX6
- mips4_IRIX6
- alpha_OSF1

so I make binary distributions for these platforms. **Karma** has also been ported to the following platforms:

- rs6000_AIX
- mips1_ULTRIX
- hp9000_HPUX
- crayPVP_UNICOS
- c2_ConvexOS
- mips1_IRIX5
- mips2_IRIX5

but since I don't have ready access, binary distributions may not be forthcoming.

If you want to run **Karma** under another operating system, I will try to make the executables for it if I can get access to a machine.

Appendix C

Resources

All the GUI visualisation tools define a number of *resources* which control the appearance and functionality of the tools. Many of these resources are common to other non-visualisation programmes for the X Window system, such as the <**xterm**> programme. Listed below are those resources which are specific to Karma visualisation tools. These resources may be set in your ∼`/.Xdefaults` file, which is the standard file into which to put your personal resources. Some systems use the non-standard ∼`/.Xresources` file instead, although the format is the same. On SGI systems you may need to put your Karma resources in a separate file and run:
`xrdb -merge <filename>`
in your window system startup script. You may also/instead (on SGI systems) need to set resources in your ∼`/.Sgiresources` file.

In some cases you will need to prepend the application classname (i.e. `Kvis`) to the resource name. First try specifying the particular resource as shown and if that doesn't work prepend the application classname.

- ***ZoomPolicy.fixAspect** this controls whether the aspect ratio will be fixed (**True**) or not (**False**)
- ***ZoomPolicy*integerXZoomToggle.state** this controls whether horizontal zooming will be forced to integer multiples (**True**) or not (**False**)
- ***ZoomPolicy*integerYZoomToggle.state** this controls whether vertical zooming will be forced to integer multiples (**True**) or not (**False**)
- ***IntensityPolicy*resetIntensityToggle.state** this controls whether the intensity range will be reset upon loading of a new file (**True**) or not (**False**)
- ***iscaleMenu*setChoice** this controls the default intensity transformation. The folowing are supported:
  - `linear` the screen intensity is proportional to the input data values
  - `log(val-min)` the screen intensity is proportional to the logarithm of the input data values

  - `log(max-val)` the screen intensity is proportional to the logarithm of the negative input data values
  - `sqrt(val-min)` the screen intensity is proportional to the square root of the input data values

- ***MagnifierPopup.fastPanner** this controls whether the magnifier window will use a fast, large-memory mode (**True**) or not (**False**) panner. The default is to use the slower, small-memory panner. In most cases the default mode is quite fast enough, unless you enlarge the magnifier window so it nearly fills your screen. The reason the default is to use the small-memory mode is that if you have a large image (say 2k*2k) then the fast mode would require 64 MBytes of RAM just to store one copy of the image! Unless you have lots of RAM in your computer, you will be forced to wait a long time while your machine swaps to disc, hence the default

- ***ImageDisplay.fastPanner** as above, but applies to the main image window. The default is to use the fast, large-memory panner. This is only of significance when you use panning mode (i.e. select "Panner" from the "Zoom" menu)

- ***MagnifierPopup.showCrosshair** this controls whether magnifier windows should show a crosshair (**True**) or not (**False**)

- ***displayAxisLabelsToggle.state** if this is **True** then most windows which support axis labels will have enabled them by default. You also need to set the ***OverlayMenu.createAtInit** resource to **True** (see below)

- ***OverlayMenu.createAtInit** if this is **True** then creation of certain control panels is performed at programme startup, rather than being delayed until if (and when) required. Normally, delayed creation is preferred, since it speeds application startup, but this prevents some resource specifications from taking effect until sometime later. Some programmes provide the shorthand "`-create_at_init`" option

- ***transient** if **True** all popup windows will be iconified if the main window is iconified

- ***AnimateControl*pinToggle.state** this controls whether the animation control will be stay popped up when an image is displayed (**True**) or will be popped down (**False**)

- ***AnimateControl.transient** as above but specifically only for the <**AnimateControl**> widget (section 2.3)

- ***Cmapwinpopup.transient** as above for the <**Cmapwinpopup**> widget (section 2.2)

- ***Dataclip.transient** as above for the <**Dataclip**> widget (section 2.2)

- ***Dataclip*subsetToggle.state** this controls whether a subset of the input data will be used to compute a histogram for the <**Dataclip**> widget (section 2.2)
- ***Dialogpopup.transient** as above for the <**Dialogpopup**> widget
- ***DressingControl.transient** as above for the <**DressingControl**> widget (section 2.8)
- ***Filepopup.transient** as above for the <**Filepopup**> widget (section 2.1)
- ***Filepopup*dotsToggle.state** control whether (**True**) or not (**False**) hidden files and directories are shown by default in the <**Filepopup**> widget (section 2.1)
- ***Filewin.numLines** the desired size of the directory browser (in lines). The default is that the browser size is automatically configured
- ***Filewin*list.translations** may be used to change the file browser mouse behaviour. The default is to highlight the entry under the current mouse pointer position. If you specify `<Btn1Down>:Set()\n<Btn1Up>:Notify()` then highlighting only occurs when an entry has been selected. The only advantage of this behaviour is that it leaves the last selected entry highlighted (i.e. moving the mouse won't highlight a different entry)
- ***IntensityPolicy.transient** as above for the <**IntensityPolicy**> widget (section 2.2)
- ***LoadAndDecimate.transient** as above for the <**LoadAndDecimate**> widget (section 2.1.1)
- ***MagnifierPopup.transient** as above for the <**MagnifierPopup**> widget (section 2.9)
- ***MomentGenerator.transient** as above for the <**MomentGenerator**> widget (section 2.16)
- ***OverlayEditorControl.transient** as above for the <**OverlayEditorControl**> widget (section 2.19)
- ***PannerPopup.transient** as above for the <**PannerPopup**> widget (section 2.10)
- ***Postscript.transient** as above for the <**Postscript**> widget (section 2.4)
- ***SlaveImageDisplayPopup.transient** as above for the <**SlaveImageDisplayPopup**> widget
- ***TracePopup.transient** as above for the <**TracePopup**> widget (section 2.11)

- ***View2Datasets.transient** as above for the <**View2Datasets**> widget (section 4)
- ***ZoomPolicy.transient** as above for the <**ZoomPolicy**> widget (section 2.7)
- **Kvis.geometry** the size and placement of the main display window for <**kvis**>. The default is `+0+0` (top left)
- **Kvis*firstDataBrowserShell.geometry** the size and placement of the first data browser window for <**kvis**>. The default is `+540-0` (to the right of the main display window and at the bottom of the screen)
- **Kvis*imageDisplayShell1.geometry** the size and placement of the second display window for <**kvis**>. The default is `-0+0` (top right)
- ***Postscript*nameDialog*font** the name of the font to use in the text-entry dialog for the output filename in the <**Postscript**> widget (section 2.4)

Appendix D

Annotation File Format

D.1 Usage

Karma has a facility to annotate 2-D images with lines, boxes, and other simple figures, as well as strings of text. Commands to draw these various things are stored in text files which are read in by selecting the "Load Annotations" option from the "Overlays" button menu (see §2.20). The names of annotation files must have the extension ".ann" or ".ANN" for Karma to recognize them. Annotations read from a file can be removed from the image with the "Remove" buttons in the Annotation File Loader. Often the best way to produce a good set of annotations for a particular image is to see what an annotation file produces on the screen, then modify the file contents, remove the annotations, and reload the file, until everything appears satisfactory.

Note the annotations descibed here only work with 2-D images, or 2-D slices of 3-D images. For information on marking rendered 3-D images with line segments in <**xray**>, see §5.4.

D.2 Fundamental Syntax

Annotation file syntax is very simple. Commands are listed in the order they are to be performed, separated by carriage returns. Each command is followed by one or more arguments, which may be separated with commas, spaces, or tabs. Comments may be placed in the annotation file by using a comment character (#). All text occuring after a comment character on the same line is ignored. Blank lines are also ignored. The syntax is case-insensitive: UPPER, lower or MiXeD cases are all valid. Commands can be extended over more than one line of text by placing a continuation character (\) at the end of each line to be continued. Any text appearing after the continuation character on the same line is ignored. Continuation characters do not continue comments, only command strings; therefore, any (\) following a (#) is ignored. If the line following a continued line is blank or a comment line, the continuation is terminated.

Note: The parser ignores any spaces, tabs, or commas preceding "useful" text on a command line. Consequently, a space should be placed prior to the

continuation character if the text beginning on the following line is not to be considered part of the last argument on the current (continued) line. The reason for this is to allow a long argument (*e.g.*, a really long font name) to be broken up if necessary without being treated as two separate arguments.

D.3 Annotation Commands

There are three types of commands: those which draw objects, those which change attributes of objects, and non-graphical commands. A complete list of currently implemented commands is given below, with syntaxes and brief descriptions. In the syntax lists, the command is given in `CAPS` (with occasional synonyms in parentheses). Required arguments follow in <`angle brackets`>, and optional arguments or argument groups in `[square brackets]`. The brackets are for illustration only and should not be typed.

D.3.1 Non-Graphical Commands

At present there is only one of these, but it is quite useful.

- `INCLUDE` <`name of file`>
  Includes the contents of another annotation file in the current one. The parser acts on the instructions it finds in the included file before moving to the next command in the present one. Thus one can include the same file's worth of annotations in several different files without having to maintain multiple copies. Furthermore, by specifying attributes such as color in the parent file instead of the included file, the attributes can be changed from one inclusion to the next.

  The filename can include the "~" notation, as well as other simple Borne Shell-like expansions of environment variables using the `$variable`, `${variable}` and `${variable:-word}` notations.

  Note: Filenames are case-sensitive on many operating systems (*e.g.*, Unix). Also, when using multi-level inclusion (including files which include other files, etc.), be careful with relative pathnames. The parser's working directory remains that of the original file; thus, if `file1.ann` has `INCLUDE subdir/file2.ann` and that file has `INCLUDE file3.ann`, Karma will look for `file3.ann` in the directory containing `file1.ann`, not in `subdir/`. Unless all the files are in the same directory, it's probably safer to use absolute pathnames or environment variables.

D.3.2 Attribute Commands

Attributes are general properties of objects. An attribute command sets one of these properties for all objects drawn after, until the attribute is changed by another attribute command. Each has an initial value it is assigned when the application first starts.

- `COLOR (or COLOUR) <name of color>`
  Sets the color of subsequent objects. The color name is a standard system color, a list of which can be found for X-Windows systems in the file /usr/lib/X11/rgb.txt. Basic color names include "red", "green", "yellow", "white", etc. The default color is green.

- `FONT <name of font>`
  Sets the font of subsequent text objects. The font name is a standard system font, a list of which can be found for X-Windows systems with the command "xlsfonts". The name of the default font is "hershey14". *Note: font names are case-sensitive!*

- `COORD <default coordinate type (W, R, or P)>`
  Sets the default coordinate type of subsequent objects. This will apply to any objects which do not have a coordinate type given in their command string. Any coordinate type in an object command string takes precedence over the default coordinate type, but only applies to that particular object. Some objects allow multiple coordinate type arguments; in this case, each coordinate type given applies only to vertices following it in the argument list, until superceded by another coordinate type specification. See below for descriptions of the different coordinate types. The initial setting of `COORD` is `W`.

- `PA <position angle convention (STANDARD or SKY)>`
  Sets the position angle convention for subsequent objects. Unlike coordinate type, position angle convention cannot be specified inside an object command, but must be set with this attribute command. See below for descriptions of the two position angle conventions. The initial setting of `PA` is `STANDARD`.

D.3.3 Object Commands

Object commands draw specific objects over a displayed image. These include text strings as well as figures made up of dots and lines.

The locations of drawn objects can be specified in three types of coordinates: world (`W`), relative (`R`), and pixel (`P`). World coordinates are those indicated by axis labeling, e.g., right ascension and declination (or velocity, in a position-velocity map) — but be careful about coordinate units, since annotations use the units specified by the image header (e.g., the FITS standard default units for right ascension are degrees, not hours, and for velocity are m/s, not km/s!). Relative coordinates range from 0 to 1 inside region bounded by the image axes, increasing from the lower left corner of the image. Pixel coordinates count screen pixels explicitly in the entire display region (including outside any labeled axes), beginning in the *upper* left corner; note these are *not* image pixels, but rather the physical pixels of the computer monitor screen. Image pixel coordinates are not a usable coordinate type here, unless no world coordinates are defined for the image, in which case Karma will use the image pixel coordinates in place of world coordinates. For most purposes

involving the marking of image features, world coordinates are the best choice, though relative coordinates are occasionally good for marking a particular part of the display region regardless of image projection. Pixel coordinates, by their inflexible nature, are as a rule not very useful. The three coordinate types are specified in an argument list with the letters `W`, `R`, and `P`.

In several of the object commands below (`DOTS, LINE, CLINES, DLINES, VECTOR`), different geometric parameters of the object can be specified in different coordinate types, and so a coordinate type argument may precede more than one coordinate pair in the argument list. However all coordinate type arguments are optional; they are only necessary to override the current value — either one specified earlier in the argument list, or the default coordinate type selected with the `COORD` attribute command.

- `DOT (or POINT) [coord_type] <x_coord> <y_coord>`
  Draw a single point.
- `DOTS (or POINTS) [coord_type] <x1> <y1> [[coord_type] <x2> <y2>] ...`
  Draw a collection of points.
- `LINE [coord_type] <x1> <y1> [coord_type] <x2> <y2>`
  Draw a line between two points.
- `CLINES [coord_type] <x1> <y1> [coord_type] <x2> <y2> [[coord_type] <x3> <y3>] ...`
  Draw a series of connected lines ("connect the dots").
- `DLINES [coord_type] <x1> <y1> [coord_type] <x2> <y2> [[coord_type] <x3> <y3> [coord_type] <x4> <y4>] ...`
  Draw a collection of disconnected lines.
- `VECTOR [coord_type] <x_base> <y_base> [coord_type] <x_comp> <y_comp>`
  Draw a vector with specified components at a given location. At present this is just a line, with no arrowhead drawn. However, unlike the `LINE` command, which specifies coordinates of both endpoints, the arguments for `VECTOR` are the position of the base and the components of the vector which get added to the base to determine the other endpoint.
- `BOX [coord_type] <x1> <y1> <x2> <y2>`
  Draw a rectangular box, given coordinates of two opposed corners.
- `CBOX [coord_type] <x_cent> <y_cent> <x_width> <y_height> [angle]`
  Draw a rotatable, rectangular box, given center coordinates, width, and height, and an optional position angle in degrees.
- `CROSS [coord_type] <x_cent> <y_cent> <x_width> <y_height> [angle]`
  Draw a rotatable cross, given center coordinates, width, and height, and an optional position angle in degrees.

- `CIRCLE [coord_type] <x_cent> <y_cent> <radius>`
  Draw a circle, given center coordinates and radius.

- `ELLIPSE [coord_type] <x_cent> <y_cent> <a> <b> [angle]`
  Draw a rotatable ellipse, given center coords, semimajor and semiminor axes, and an optional position angle in degrees. The semimajor axis must be given before the semiminor axis if adherence to the standard definition of ellipse position angle is desired. In order to maintain consistency with the `CBOX` and `CROSS` commands, the first axis is the one oriented by the position angle, regardless of whether it is the largest.

- `TEXT [coord_type] <x_left> <y_left> <string>`
  Draw a text string left-justified at the given coordinates (more precisely, these specify the lower left corner of text). The string to be used is whatever follows the third argument, e.g., several words separated by spaces, and ends with the line of text, unless continued onto another line. Comment (#) and continuation (\) characters cannot be included in a text string, but are acted upon in the usual fashion if found — thus text commands can also be extended to multiple lines.

WARNING: At present, rotatable figures (boxes, crosses, and ellipses) have a tendency to appear distorted from their "real" appearance in curved world coordinates, e.g., maps of the sky. Proper projection of geometric objects into world coordinates is a nontrivial exercise and has been deferred.

D.3.4 Position Angle Conventions

Standard Postition Angles are generally defined in the Cartesian sense where $PA = 0$ specifies $\{X > 0, Y = 0\}$, $PA = 90^\circ$ specifies $\{X = 0, Y > 0\}$, and so forth. In a coordinate system where X increases from left to right, and Y increases from bottom to top, PA will increase in a counter-clockwise direction. However this will be reversed if the X increases from right to left, or Y from top to bottom (though not both). So be careful! To summarize: PA is measured CCW from right in `R` coords, and CW from right in `P` coords. How it is measured in `W` coords depends on the coordinate system: in many cases, it will be the same as `R` coords, but this is not always true! For example, most astronomical maps of the sky have the X coordinate increasing toward the left, in which case, PA will be measured CW from left!

Sky Position Angles are for the astronomers. Usage of this convention in annotations assumes two things: (1) a map of the sky is being displayed, with the X coordinate increasing from right to left, and the Y coordinate increasing from bottom to top, i.e., north = up, south = down, east = left, west = right; and (2) position angle is measured from north through east (up through left). The convention is followed regardless of whether the coordinate type is `W`, `R`, or `P`. So unlike `STANDARD` position angles, `SKY` PAs retain their appearance regardless of coordinate type. This same system can be used with Ecliptic and Galactic coordinates as well as Equatorial, though the $PA = 0$ referent points up in all cases, toward the grid's north pole. *However* unless the image projection is flat,

$PA = 0$ will only point exactly north at the image projection center (but if the image covers a small piece of the sky and is not close to a coordinate pole, the error will be small). Generalizing this to be correct in all parts of the image is a future project.

D.4 Example Annotation File

Two sample annotation files, `m42.ann` and `m42_include.ann`, have been provided to illustrate the functions and concepts outlined above. To use these example files properly in Karma, it is necessary to load the FITS image m42_ha.fits to set the world coordinates. For best image appearance, select a square root intensity scale from the Intensity Policy popup, and a 99.99% intensity range from the IScale popup.

These example files and some image data may be found at:
`ftp://ftp.atnf.csiro.au/software/karma/data/annotations/`

```
######################################################################
#
#          SAMPLE KARMA ANNOTATION FILE M42.ANN
#
#        *** to be used with M42_*.FITS images ***
#
######################################################################

# Comments are preceded by a ``#'' character
#
# All of the commands names below are given in CAPS for readability,
# but they could just as easily be lower case.
#
# World coordinates here are decimal degrees of Right Ascension and
# Declination (in accordance with the FITS standard).

COORD W           # These are the attribute settings at startup time.
PA STANDARD       # If no annotation files loaded prior to this one
COLOR GREEN       # have changed any attributes, then these commands
FONT hershey14    # are reduntant.

# Label the image near the top.  Use relative coordinates to fix
# the position regardless of image zoom.
#
COLOR WHITE
TEXT R 0.1 0.9 THE ORION NEBULA

# Change font to something a little smaller for other labels.
FONT hershey12

# Mark the four Trapezium stars with blue circles, and connect them.
```

```
# Use world coordinates for these markers and name label.
# Since COORD is currently set to W, specifying W coords in the
# argument lists of the objects below is not necessary.
#
# Note W coords are in degrees for consistency with coordinate
# information in the FITS image header, even though RA is labeled
# in hours by Karma.  Note also the TEXT string is continued onto
# another line.  The leading space is ignored, but a trailing space
# after "Trap" would not be.  The commas in the CLINES command
# string are not necessary, but make it easier for a human to read.
#
COLOR BLUE
DOT 83.81613 -5.38706             # Theta 1A Orionis; HD 37020
DOT 83.81738 -5.38494             # Theta 1B Orionis; HD 37021
DOT 83.81875 -5.38972             # Theta 1C Orionis; HD 37022
DOT 83.82208 -5.38778             # Theta 1D Orionis; HD 37023
CIRCLE 83.81613 -5.38706 0.0005   # Theta 1A Orionis; HD 37020
CIRCLE 83.81738 -5.38494 0.0005   # Theta 1B Orionis; HD 37021
CIRCLE 83.81875 -5.38972 0.0005   # Theta 1C Orionis; HD 37022
CIRCLE 83.82208 -5.38778 0.0005   # Theta 1D Orionis; HD 37023
CLINES 83.81613 -5.38706, 83.81738 -5.38494, 83.82208 -5.38778, \
                83.81875 -5.38972, 83.81613 -5.38706
TEXT 83.84 -5.38 ``Trap\
        ezium''

# Indicate the star Theta 1C Orionis, source of much of the UV flux
# ionizing the gas in the nebula.  Use a mixture of world and
# relative coordinates to fix the label on the screen when zooming.
#
COLOR YELLOW
LINE W 83.81875 -5.38972 R 0.6 0.1
TEXT R 0.6 0.10 Theta 1C

# Mark the region imaged by the HST mosaic released in 1995
# (http://oposite.stsci.edu/pubinfo/pr/95/45.html).
#
COLOR RED
BOX  83.86867 -5.44977 83.74646 -5.31922
TEXT 83.86867 -5.31922 HST

# One transverse light year at a distance of 1500 ly subtends 0.038
# degrees.  At a declination of -5.39 degrees, this is 0.0382 Right
# Ascension degrees.
#
COLOR GREEN
CBOX  83.91196 -5.49598 0.0382 0.038
CROSS 83.91196 -5.49598 0.0382 0.038
```

```
TEXT  83.88 -5.49598 1 ly^2

# Here is a series of disconnected lines, for no very good
# reason.  The last continuation character is ignored because
# the following line is empty.
#
COLOR LIGHT GRAY
DLINES R \
  0.96 0.05 0.99 0.05, 0.96 0.10 0.99 0.10, 0.96 0.15 0.99 0.15, \
  0.96 0.20 0.99 0.20, 0.96 0.25 0.99 0.25, 0.96 0.30 0.99 0.30, \
  0.96 0.35 0.99 0.35, 0.96 0.40 0.99 0.40, 0.96 0.45 0.99 0.45, \
  0.96 0.50 0.99 0.50, 0.96 0.55 0.99 0.55, 0.96 0.60 0.99 0.60, \
  0.96 0.65 0.99 0.65, 0.96 0.70 0.99 0.70, 0.96 0.75 0.99 0.75, \
  0.96 0.80 0.99 0.80, 0.96 0.85 0.99 0.85, 0.96 0.90 0.99 0.90, \
  0.96 0.95 0.99 0.95, \

# Change default coordinates to relative.  In the DLINES command
# above, the default was world coordinates, so the R was necessary
# to specify something else.
#
COORD R

# Here are a of couple vectors.
#
COLOR CYAN
VECTOR 0.1 0.25, 0.05 0.05
VECTOR 0.1 0.30, 0.05 0.05

COORD W     # Change back to world coordinates as the default.

# Here are three plain, adjacent ellipses.  No position angle
# parameter is given, so PA=0.  However this applies to the first
# semiaxis given, not necessarily the larger of the two.  With the
# semiminor axis listed first, these ellipses have an effective
# position angle of 90 degrees (in the ``standard'' system -- see
# below).
#
COLOR ORANGE
ELLIPSE 83.9151 -5.3978 0.002 0.005
ELLIPSE 83.9111 -5.3978 0.002 0.005
ELLIPSE 83.9071 -5.3978 0.002 0.005

# Put a dot in the center of all three orange ellipses, using a
# single command.
#
DOTS 83.9151 -5.3978, 83.9111 -5.3978, 83.9071 -5.3978
```

```
# Include another annotation file, which has examples of some
# rotatable objects.
#
INCLUDE m42_include.ann

######################################################################
#
#          END OF SAMPLE KARMA ANNOTATION FILE M42.ANN
#
######################################################################

######################################################################
#
#        SAMPLE KARMA ANNOTATION FILE M42_INCLUDE.ANN
#
#         *** to be used with M42_*.FITS images ***
#
######################################################################

# Draw rotated crosses, cboxes, and ellipses with standard PAs.

PA STANDARD

# (The W's aren't actually necessary since COORD is still set to W.)

COLOR AQUAMARINE
CBOX    W 83.72204 -5.45408 0.02 0.008 10
CROSS   W 83.72204 -5.45408 0.02 0.008 10
ELLIPSE W 83.72204 -5.45408 0.01 0.004 10

COLOR MAGENTA
CBOX    R 0.8 0.15 0.05 0.03 10.0
CROSS   R 0.8 0.15 0.05 0.03 10.0
CBOX    R 0.8 0.15 0.10 0.06 10.0
ELLIPSE R 0.8 0.15 0.05 0.03 10.0

COLOR VIOLET
CBOX    P  75 200 50 20 10
CROSS   P  75 200 50 20 10
ELLIPSE P  75 200 25 10 10

# Draw rotated crosses, cboxes, and ellipses with ``sky'' PAs.

PA SKY

COLOR TAN
CBOX    W 83.7147 -5.3007 0.02 0.01  10
```

```
CROSS   W 83.7147 -5.3007 0.02 0.01  10
ELLIPSE W 83.7147 -5.3007 0.01 0.005 10

COLOR KHAKI
CBOX    R 0.88 0.70 0.08 0.04 10
CROSS   R 0.88 0.70 0.08 0.04 10
ELLIPSE R 0.88 0.70 0.04 0.02 10

COLOR GOLD
CBOX    P 450 220 44 22 10
CROSS   P 450 220 44 22 10
ELLIPSE P 450 220 22 11 10

#########################################################################
#
#       END OF SAMPLE KARMA ANNOTATION FILE M42_INCLUDE.ANN
#
#########################################################################
```

Appendix E

Making Videos

Sometimes you will want to prepare a video for presentation. The procedure to do this depends on the equipment you have available and the desired quality of the production. This appendex discusses some options.

E.1 Basic Facilities at the ATNF

The ATNF has video recording equipment connected to the workstation **phoenix**. This equipment is not capable of producing high-quality (i.e. broadcast standard) videos, but it has the advantage of being located at the ATNF, and so is convient to use.

Once you have produced your movie, you need to purchase a VHS video cassette and use the video recorders supplied. When playing the movies, it is advisable to use the `-fullscreen` command-line option for the kvis (chapter 3) and xray (chapter 5) tools. This will make the image display window take up the entire screen. You should see a button labelled **Raise** somewhere which allows you to place the displayed image above all other windows. By clicking the right mouse button you can lower this window again, and get back your control panels.

E.2 Using Sydney University VisLab Facilities

The University of Sydney has created a commercial visualisation laboratory. They are able to make high-quality broadcast standard VFS and Betacam videos. All you need to do is provide them with a sequence of numbered images in a supported format.

VisLab also have facilities to convert from many popular image formats to a format their machinery requires, but this will cost real money. Instead, you should produce files in "Targa TrueVision" format, which they will be able to use directly. The karma2ppm (section 12.5.3) command-line utility may be used to produce this format.

The image size required by VisLab is 720 pixels horizontal and 576 pixels vertical. Note that you need to leave a border of 10% inside this otherwise some television screens will chop the outer parts of your movie.

You would need to use the following parameters for <**karma2ppm**>:

```
converter           "ppmtotga"
extension           "tga"
out_height          576
out_width           720
```

You would then type something like:

```
karma2ppm mymovie image
```

and you would get a sequence of numbered images starting from 1, such as:

```
image_01.tga
image_02.tga
image_03.tga
image_04.tga
image_05.tga
image_06.tga
image_07.tga
image_08.tga
image_09.tga
image_10.tga
image_11.tga
image_12.tga
image_13.tga
image_14.tga
```

Try to keep the filenames to 8.3 lengths (i.e. 8 characters for the first part and 3 characters for the extension). This avoids any potential problems with some (broken) operating systems.

These files are already compressed, so further compression is probably not worthwhile. Write these images onto a tape or CD-ROM and take it to VisLab. It is recommended that you attend the video recording session in person, in case there are problems. You may find that the colour balance, intensity or contrast is different on your workstation screen than on the television screen. Aborting the process early if there are problems will save you money.

Rememember: it's *your* video. Nobody knows exactly how you want it to appear except you.

Appendix F

Acknowledgements

Nearly all of the Karma software and documentation has been written by Richard Gooch, formerly of the CSIRO Australia Telescope National Facility, and later at the Department of Physics and Astronomy, University of Calgary. Richard Gooch may be reached by email at
`rgooch@safe-mbox.com` or on the WWW at
`http://www.safe-mbox.com/~rgooch/`

Some of the early widgets (such as the file browser and colourtable controller) were written by Patrick Jordan, then with the CSIRO Division of Radiophysics. In addition, Patrick contributed through lengthy architectural discussions. Patrick Jordan may be reached by email at
`patrick@ariel.com.au` or on the WWW at
`http://www.ariel.com.au/patrick/`

The anti-aliasing image expansion code and the wavelet-based filtering code was written by Tom Oosterloo, then with the CSIRO Australia Telescope National Facility.

The first versions of the programme <**khuei**> (section **??**) were written by Simone Magri of the CSIRO Australia Telescope National Facility. Simone Magri may be reached by email at
`Simone.Magri@atnf.csiro.au` or on the WWW at
`http://www.atnf.csiro.au/~smagri/`

The <**kpvslice**> (chapter 6) and <**koords**> (chapter 9) programmes were a collaborative effort between myself and Vincent McIntyre (then with the University of Sydney). Vincent McIntyre may be reached by email at
`Vince.McIntyre@atnf.csiro.au` or on the WWW at
`http://www.atnf.csiro.au/~vmcintyr/`

The Starlink to Karma conversion programme <**sdf2karma**> was written by Karl Glazebrook of the Anglo Australian Observatory. Karl Glazebrook may be reached by email at
`kgb@pha.jhu.edu` or on the WWW at
`http://www.aao.gov.au/local/www/kgb/`

The annotation file reading code was designed and implemented by Steven Gibson, then with the University of Calgary and currently with Arecibo Observatory. The appendix describing the data format was also written by Steve.

He may be reached by email at
gibson@naic.edu or on the WWW at
http://www.naic.edu/~gibson/

Appendix G

Quirks and Unexpected Behaviour

There are some operations that may not behave in the way you may expect, although they are operating according to design. These are "quirks" rather than "bugs". Below is a list of quirks.

- When loading a file and "Reset Intensity Upon File Load" is disabled, the intensity range of the previous image will be used. However, the intensity range that is used is the *raw* range, not the scaled range. Thus, with images which have different scaling parameters, the scaled intensity range will change.
- When moving between two images (such as when blinking, or when loading a new image), the display system attempts to retain the old image parameters (i.e. zoom box). What is retained are the corners of the zoom box measured in image pixel co-ordinates, not world co-ordinates.
- Co-ordinate positions start from 0, not 1.

Appendix H

Undocumented Features

Index