Manual to aid the visualisation of Quantum Chromodynamics

Helen Ashton

October 7, 1994

1 Introduction

This manual is intended to aid the user in visualising the physical properties contained in data sets produced by United Kingdom Quantum Chromodynamics (UKQCD) on machines at the University of Edinburgh.

The manual covers the programmes available to extract the physical properties desired, as well as the two visualisation packages used. This is only intended to be an introduction to the facilities and what has already been set up in these packages. For further help with the visualisation packages mentioned please refer to the appropriate user manual for that package (more information about this later).

This manual was written as part of a Summer Scholarship Project (SSP) carried out at the Edinburgh Parallel Computing Centre (EPCC). It does not include an introduction to QCD. For a brief introduction to Quantum Chromodynamics (QCD) please refer to final report EPCC-SS94-09 [1].

2 Background

Four dimensional spacetime datasets are created by the UKQCD group using a Monte Carlo method. These sets of data describe how gluons interact with each other. From these datasets various physical properties can be calculated, some of which will be mentioned later. The fact that these data sets are four dimensional complicates the visualisation. Because of this most existing visualisation occurs in the form of one dimensional graphs representing the evolution of a physical property with time. However this does not take full advantage of the four dimensional nature of the simulations and important properties may be overlooked by the user. Visualising gives the UKQCD group a new aspect on their data simply because it provides a new way of viewing it.

Visualisation of QCD data is a relatively new field. This means that the work carried out was a trial and error procedure. It was not known if visualisation of the properties was possible, nor if it would make any sense physically. The work done does provide a basic structure and background to the visualisation techniques available at EPCC.

Hopefully this will lead to more properties being visualised, and give the physicist insight into previously hidden properties.

The first part of the project that this manual accompanies of involves combining the data to produce particles called mesons (this is covered in the first part of the report for this project). The data which result describe the motion of a particle that originated at a point in space (the centre of the data set). It is almost impossible to see what is happening to a three dimensional object by looking at two dimensional graph, so visualisation of the volume is important. Included in this manual are the details of how to go about creating and visualising mesons, starting with the original data taken from the QCD simulations.

The second part of the project looks at visualising the function used in the gauge fixing stage. This is a process that is carried out to make calculations easier (for a more detailed explanation of this please refer to report [1] that accompanies this project). Without visualisation of this property it is very hard to tell if this stage has been implemented correctly.

It is therefore an important part of both validating the simulations, and for further research, to be able to visualise the various physical properties in a variety of ways.

3 Facilities available

This section contains an explanation of the facilities that have been provided. There will be a description of the programmes that have been created to aid the visualisation, as well as an introduction to the visualisation packages.

The various facilities included in this manual are intended to take data from the UKQCD group in its initial form (IEEE floating point, 32-bit, single precision, binary representation. X should be the fastest varying axis, followed by Y, then Z). If the data is not in this form then it will be necessary to convert it for use of the programmes that have been provided. If only the visualisation packages are to be used there are various data import modules available which will be mentioned later, however the facilities covered in this manual assume the data type which results from use of the programmes mentioned in the following section.

3.1 Programmes

The following programmes have been written specifically for extracting physical properties from the raw data and preparing this for visualisation.

The programmes which relate to extracting physical properties from the raw data are all run from a program called `menu` which prompts the user to select the appropriate option

The following options are available:

- creating mesons from the raw matrix data — this programme takes the raw data created by the UKQCD simulations and creates the required meson. (see section 4.1)

- averaging over a number of configurations — to make the visualisation more meaningful. (see section 4.2)
- converting timeslice data into probabilities — to make the values visualised more meaningful (see section 4.3)

There are a couple of things the user should note before using the programmes provided. It is very likely that the user will use a different machine to create the data for visualisation than that used to visualise the data. This can cause problems. Two specific problems have been discovered. The first is concerning the size the machine assigns to reals. On UKQCD this size is one, but on the Suns and SGI's this is four. A parameter called *fsize* has been encorporated into the programmes to compensate for this. The programme, `menu`, (see section 4.7) has been set up with a size of one for use on UKQCD, however if the programme is to be used elsewhere then this size will need to be checked and altered as appropriate. The second thing to note is the way the machine stores floating point numbers. For further information on this see section 4.5. Finally the user should read section 4.4 for an explanation of how to enter in filenames when using these programmes.

Shell scripts have also been used to carry out various tasks whilst working on this project. Shell scripts are a very useful way of carrying out tasks which involve entering a very similar command at the prompt many times, for example renaming a large number of files. For further information and examples see section 4.6.

3.2 Visualisation

Visualisation packages were created to convert data into meaningful images. In many cases this is the only way data will make any sense. For example, the numbers in a dataset that describes the image of a lamp will not help in working out what the image is. The best way to find out what is contained in the data set is to look at the two or three dimensional image. There are various types of visualisation packages. Some packages are designed for a specific type of visualisation and/or data. Others provide a base on which to build tools to create the required image. Both have advantages and disadvantages.

The three packages available for visualisation at EPCC are BOB, Explorer and AVS. BOB was chosen because of it's speed, the idea being to use BOB as an initial viewing tool. The other visualisation packages available at EPCC are AVS and Explorer. Explorer and AVS are very similar, however the choice of Explorer over AVS was mainly taken because one of my supervisors (Gordon Cameron) had experience with this package.

3.2.1 BOB

BOB stands for Brick Of Bytes which is the data format (see section 5.2) for this visualisation package. BOB was created for use in the American Army high performance Computing research center. The key feature was to allow images of three dimensional volumes to be viewed quickly and interactively. Although there is little that can be done apart from viewing all of, or sections of, a 3D volume and rotation, BOB is a very useful tool. It has stability and speed firmly on its side. It enables the user to browse through data files very quickly, and therefore choose those which require more detailed viewing. This is where Explorer comes in (see section 3.2.2).

3.2.2 Explorer

Explorer is another visualisation tool. However it is quite different from BOB. Explorer uses *modules* (see section 6.5) to create *maps* (see section 6.7) to view the data in the required way. This enables the user to view the data in many different ways. As for disadvantages Explorer is much slower than BOB, especially for large data sets. The stability of this package is not one of its strong points. It is possible to create maps which go round infinite loops, or *hang* Explorer by reading in data files of the wrong type, or which are too large. Often the only way to rectify errors like this is to *kill* the package and restart. It is therefore important to view any data in BOB initially to help avoid these problems.

4 Using the programmes

This section explains how to use all of the programmes provided and more details of what functions they perform. The programmes contained within `menu` will be covered first, followed by any remaining programmes and shell scripts.

`Menu` calls up a single menu which prompts the user for selection of one of the following programmes.

4.1 Option 1 - creating mesons

Selecting this programme allows the user to create mesons such as the pion, rho etc. The gamma matrix required by the programme is the resultant of the multiplication of the two relevant matrices for that particle in the matrix multiplication. For example for the particles visualised the gamma matrices used are as follows:

the pion

γ_5^2

the rho

(a) $\gamma_1\gamma_5$

(b) $\gamma_2\gamma_5$

(c) $\gamma_3\gamma_5$

Once selected from the menu this programme will ask for the dimensions of the data set. It requires all three dimensions, eg.

```
please enter dimensions of the data X Y Z

16 16 16
```

You will then be prompted to choose whether to type in the gamma matrix (4×4) element by element, or load it in from a file (default `gamma.dat`). Your matrix will now be stored in gamma.dat for further use. The gamma matrix is a complex matrix so each element has a real and imaginary part. If an element had a real part of 0.0 and an imaginary part of 1.5 then the user would enter:

```
( 0.0 , 1.5 ) Return
```

For example the relevant matrix for the pion (γ_5^2)

```
(1.0 , 0.0)   (0.0 , 0.0)   (0.0 , 0.0)   (0.0 , 0.0)
(0.0 , 0.0)   (1.0 , 0.0)   (0.0 , 0.0)   (0.0 , 0.0)
(0.0 , 0.0)   (0.0 , 0.0)   (1.0 , 0.0)   (0.0 , 0.0)
(0.0 , 0.0)   (0.0 , 0.0)   (0.0 , 0.0)   (1.0 , 0.0)
```

would be entered in as follows:

```
 GAMMA MATRIX
 ------------
 Please enter in the matrix by elements when prompted
 file (1) or enter (2)
USER> 2
element (           1,           1)
USER> (1.0,0.0)
 element (           1,           2)
USER> (0.0,0.0)
 element (           1,           3)
USER> (0.0,0.0)
 element (           1,           4)
USER> (0.0,0.0)
 element (           2,           1)
USER> (0.0,0.0)
```

etc...

The gamma matrix will be displayed for verification. Next the programme requires the filenames of the data matrices (see section 4.4 for more information on entering filenames).

The resulting output data file contains data, in the form of floating point numbers, X varying fastest then Y, then Z, which could now be visualised. However use of the next two programmes is recommended for more meaningful visualisation.

4.2 Option 2 - Averaging data

This programme will average data over a number of files. The purpose of this is to create data sets that have greater physical meaning.

Once the user has selected this option from the menu they will be required to enter in the filenames in the form explained in section 4.4. The programme will then average the data contained in these files, writing the result to the output file specified by the user.

4.3 Option 3 - Translating into probabilities

To aid the user in locating the data in the visualisation stage and to give physical meaning to the data values, it is necessary to convert the data into probabilities. In otherwords scaling the data so that the total sum of all the data values is one, one being 100% probable.

Once the user has selected this programme s/he will be asked for the names of the files to be converted into probabilities. Filenames should be in the form specified in section 4.4.

4.4 Filename styles

To aid the user and to minimise the amount of time spent entering information, part of the design of the programme includes *styles* of filenames. This section will contain a description of what is meant by a filename style and how to make best use of this facility.

This facility was designed around the way the filenames are structured in the raw data files from the UKQCD simulations. These files tend to be of the form

```
C60L4320L00    C62L4320L00    C64L4320L00
C60L4320L01    C62L4320L01    C64L4320L01
C60L4320L02    C62L4320L02    C64L4320L02     etc...
```

where `L4320L` relates to the pion mass.

There is an obvious underlying structure to this of

C <*configuration*> L4320L <*timeslice*>

When using the programmes in `menu` to extract the physical properties, the base and extension tend to stay constant throughout a number of executions, with either the first or second number altering. It would therefore make sense to combine this into the programme to free the user from entering filenames manually and being tied up during the execution of the programme.

For example, if the files are named as those mentioned above, the dialogue would be as follows:

```
 You will be asked to enter in the input
    filenames in the following form:
     base   number1   ext   number2

 If you do not wish anything in base or
  ext then simply press return and ignore it
 If it is a number category you do not
  require then type -1

 the base
C
 the first number of number1 (integer please)
```

```
60
 the last number of number1
64
 the step between the numbers (integer)
2
 the extension
L4320L
 the first number of number2
0
 the last number of number2
2
 the step between the numbers
1
```

This will produce the required filenames. For verification the filenames will be displayed as they are opened.

As mentioned in the above dialogue it is possible to vary the style of filenames by pressing `return` when prompted for a string and `-1` when prompted for an integer.

The best way to use this facility is always to have filenames in the form of

<*Base*> configuration number <*extension*> timeslice number

to speed up the use of programmes and to avoid renaming files. For an easy way of renaming files see section 4.6 on shell scripts.

4.5 Converting data

This programme provides two facilities. It swaps the bytes of the floating point numbers and translates the data into *brick of bytes* form for use with BOB.

Swapping bytes is only relevant when using the same data on different machines. When using all the facilities described in this manual, the user will most likely be using the UKQCD machine and an SGI machine. This is when problems occur and data transferred from one machine to the other may have made perfect sense on one machine, but will be incompatible on the next. In other words one machine is big endian and the other little endian.

Therefore it is necessary is take account of this when changing from one machine to another. A programme has been provided called `convert` (written by Stephen Booth) which enables these bytes to be swapped (output file extension .swap) and to convert the file into character form (output file extension .char) for use in BOB. The swap stage can be done on either machine, but the character stage must be done on the machine where the data makes sense, ie. the meaningful floating point numbers must be changed into the character file.

4.6 Shell scripts

Shell scripts can be written to carry out multiple tasks, for example using `convert` on many files. For example the following shell script will display *converting filename* ie. *converting* `data00.dat`, run this file through `convert` producing an output files of `converted00.swap` and `converted00.char` (the file extensions are added by `convert` not the shell script) and finally display *finished*. This will be carried out for all 48 numbers.

```
#!/bin/sh

for number in 00 01 02 03 04 05 06 07 08 09 10
11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
42 43 44 45 46 47
do
echo converting data$number.dat
./convert data$number.dat converted$number
echo finished
done
```

Another example of a shell script that may be useful to the user is the following which adds in the dimensions of the file (contained in `dims.dat`) to the data file (eg `pion00.swap` and outputs it to another file (eg `pion00.dat`:

```
#!/bin/sh

for number in 00 01 02 03 04 05 06 07 08 09 10
11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
43 44 45 46 47
do
echo pion$number.swap
cat dims.dat pion$number.swap > pion$number.dat
echo pion$number.dat
done
```

These shell scripts will need to be altered to use the required input and output filenames.

4.7 Where to find the programs

The code, maps, modules and other files needed to use the facilities mentioned in this report have been compressed and packaged using

```
tar cvf QCDvis_?????.tar directories
```

then

```
gzip QCDvis_?????.tar
```

These will need to be uncompressed before use as follows:

```
gunzip QCDvis_?????.tar.gz
```

followed by

```
tar xvf QCDvis_?????.tar
```

The following `tar` files exsist:

QCDvis_code.tar.gz contains all code written. Following decompression contains `.../Fortran` directory which contains all the code.

QCDvis_doc.tar.gz contains all documentation. Following decompression contains `.../REPORT` directory, which contains the files to create the report and manual for this project, and a `.../POSTER` directory which contains the files to create the poster for this project.

QCDvis_images.tar.gz contains some of the images created during this project. After decompression this will contain directories `.../IMAGES` and `.../GIF` which contain the various images.

QCDvis_packages contains facilities for Explorer and BOB. Following decompression consists of `.../Maps` and `.../Modules` directories which contain the Maps and Modules for Explorer, and `.../colour` and `.../opaque` directories which contain colour and opacity maps used in BOB. This file also contains `...DOTexplorerrc` file needed for Explorer.

All directories contain `README` files describing their contents.

5 About BOB

As mentioned earlier (see section 3.2.1) BOB is a visualisation package that creates fast, interactive images of three dimensional volumes of data (see section 5.2). Bob has many benefits...

- BOB is extremely fast
- There are no problems with locating the data range of the image
- BOB is extremely stable
- Many files can be loaded in at once and browsed through rapidly

... and drawbacks...

- BOB runs only on an SGI terminal — no remote logins!
- BOB offers only one viewing style — volume rendering

- BOB data must be in a particular form (see section 5.2)

Due to the limited facilities available in BOB this proved to be the easier of the two packages to become familiar with. To use BOB it is necessary for the user to learn how to load in files, use the colour and alpha maps and the dimensions window.

5.1 How to use BOB

To use BOB you must be at an SGI graphics terminal. To start using BOB with a block of data (see section 5.2) of dimensions X $\times$ Y $\times$ Z:

```
bob -s X×Y×Z filename
```

For example a file called `junkdata.char` with dimensions of 12 x 10 x 14 would be loaded into BOB by typing:

```
bob -s 12×10×14 junkdata.char
```

When BOB has loaded up the user will be presented with two windows, the main window (titled *bob* and the *Bob Data Dimensions* window. The left hand side of the main window contains the image and the right hand side contains the following

- pull down menus including the colour and alpha maps (see section 5.4)
- the files that have been loaded into BOB

The dimension window is the control window. Initially the box of data will have dimensions of no greater than 32$\times$32$\times$32. The user can then zoom in on sections of the cube, rotate the cube, and pan out. The dimension window echos the movement of the image, although the user cannot control movement from this window. However what can be done from this window includes resizing the image, and moving the location of the centre of the image. The sliders contained in this window can be used for that purpose.

5.2 Data types

As mentioned earlier BOB's data type is bricks of bytes. In other words data must be in the form of one value per site, with this value being between 0 and 255. A program called `Convert` (see section 4.5) has been written, by Stephen Booth, to translate data into this format.

5.3 Viewing your data

When BOB starts up it views the image in the data file at the top of the list of filenames. To view another file simply click over the filename with the left mouse button and BOB will display this file. There is also a scroll bar on the right of the list of files that can be used in the normal way.

When the mouse is in the section of the main window that contains the image the mouse buttons have various functions

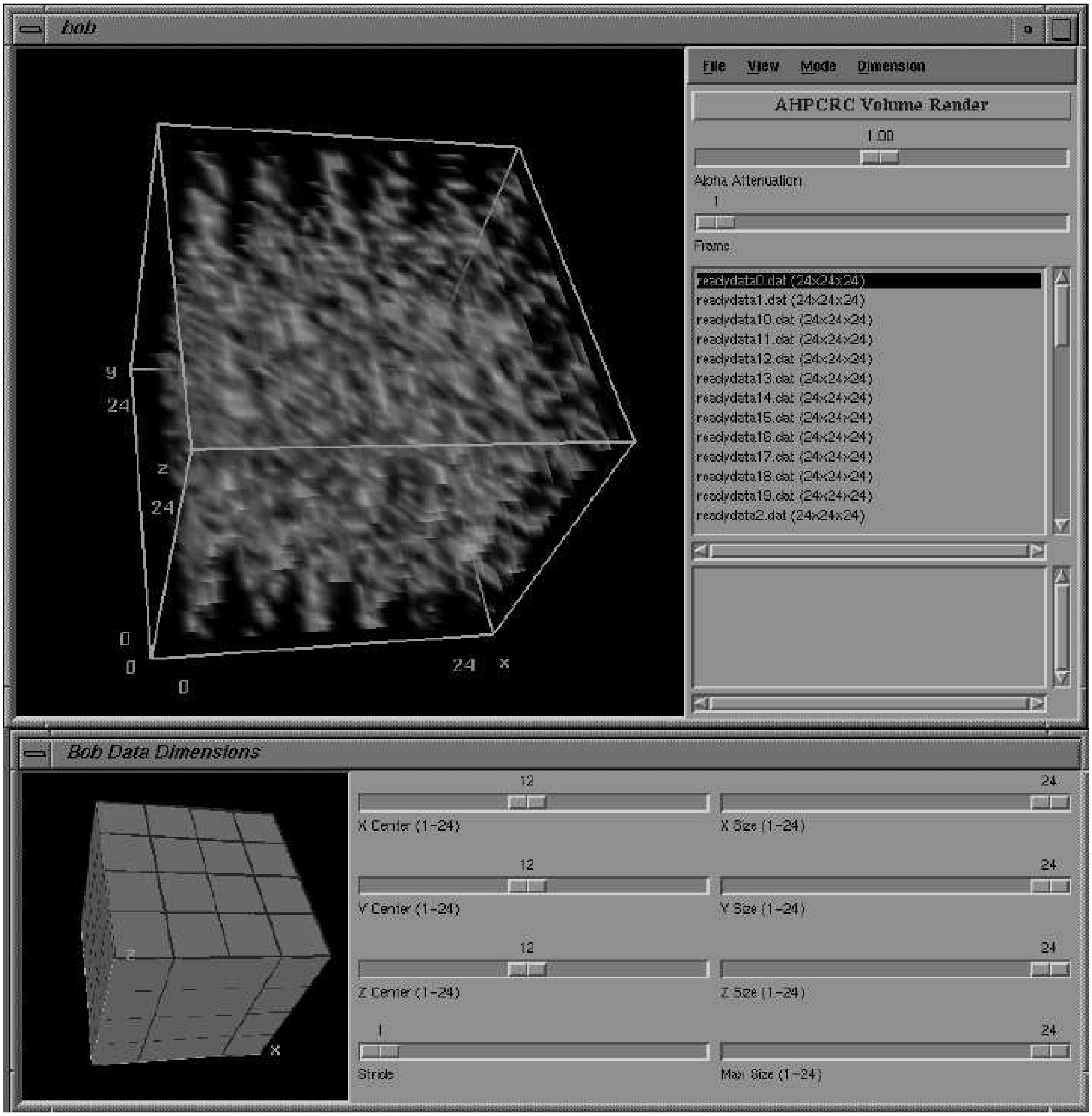
bob
File
View
Mode
Dimension
AHPCRC Volume Render
1.00
Alpha Attenuation
1
Frame
readydata0.dat (24x24x24)
readydata1.dat (24x24x24)
readydata10.dat (24x24x24)
readydata11.dat (24x24x24)
readydata12.dat (24x24x24)
readydata13.dat (24x24x24)
readydata14.dat (24x24x24)
readydata15.dat (24x24x24)
readydata16.dat (24x24x24)
readydata17.dat (24x24x24)
readydata18.dat (24x24x24)
readydata19.dat (24x24x24)
readydata2.dat (24x24x24)
Bob Data Dimensions
X Center (1-24)
X Size (1-24)
Y Center (1-24)
Y Size (1-24)
Z Center (1-24)
Z Size (1-24)
Stride
Max Size (1-24)

Figure 1: BOB

- Left mouse button - holding this down and moving will rotate the image
- Middle mouse button - holding this button down and moving will move the image in the plane of the screen
- Right mouse button - holding this button down will cause the image to zoom towards or away from the user depending on whether it is moved up or down.

The image can also be continuously rotated by letting go of the left mouse button whilst still moving the mouse. The image will keep moving in that direction until the user clicks over it.

5.4 Modifying the image

The image you first get in Bob will need some modification for optimum viewing. When BOB initially visualises a file it does so using a greyscale map with no specific opacity map. Almost certainly this arrangement will not emphasise the most important parts of the image and so it will be necessary to alter this.

There are options in the `Mode` menu which will alter the image (*auto alpha, maximum value* etc.), however the most useful way of getting the most from the image is to alter the colour and alpha maps.

These maps are used to alter the colour and opacity (alpha) display. They can be found in the `File` menu at the top of the main window. Choose the `Colour map editor` or `Alpha map editor` option.

These maps are very simple to use. Both maps contain a grid of boxes. Each box in the grid corresponds to a data value between 0 and 255, 0 being the top left box, and 255 the bottom right box. By clicking on these you can then alter the properties of the value that corresponds to that box. The colour map has two options for altering colours (contained in the `Mode` menu)

- RGB — Red, green and blue
- HSV — Hue, saturation and value
- YUV (video only)
- Alpha only

At the bottom of the map a slider will appear corresponding to the above styles, for example in RGB red, green and blue sliders will appear. These can be altered in the usual manner. The alpha map is similar but this time an Alpha slider appears. Changes in these maps are automatically passed onto BOB. In the menus at the top of the icol display there are various other options, *undo, snap* etc. It is suggested that the user explores these options to make best use of the maps.

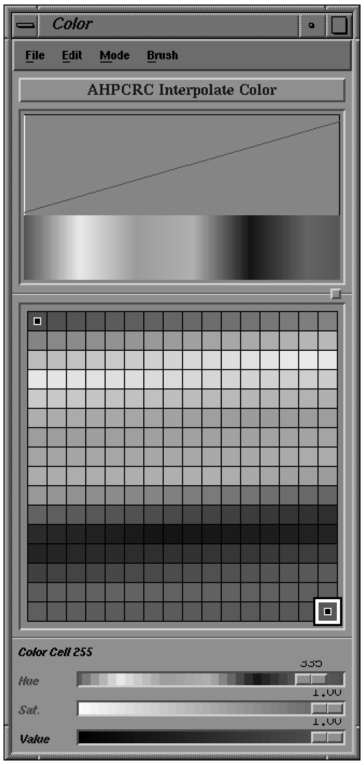

Figure 2: Colour map

5.5 Where to go for more information

There is a user manual provided for BOB called "All About Bob - A tool for browsing 3D data sets" [2]. This can be found in the following directory:

`nevis/Tools/Evaluation/GVL/vis_tools/doc/allbob.show.PS.Z`

and must be copied into the users directory and uncompressed to view as follows:

`uncompress allbob.show.PS.Z`

and can be viewed using ghostview:

`ghostview allbob.show.PS &`

6 Getting started with Explorer

To use Explorer you must be at an SGI terminal. Start up explorer by typing:

`explorer &`

at the prompt. If using the `.explorerrc` file provided (see section 6.4) the user will also need to type:

`export EXPLORER_VERSION=V2`

The user will be presented with a startup window, and then three other windows:

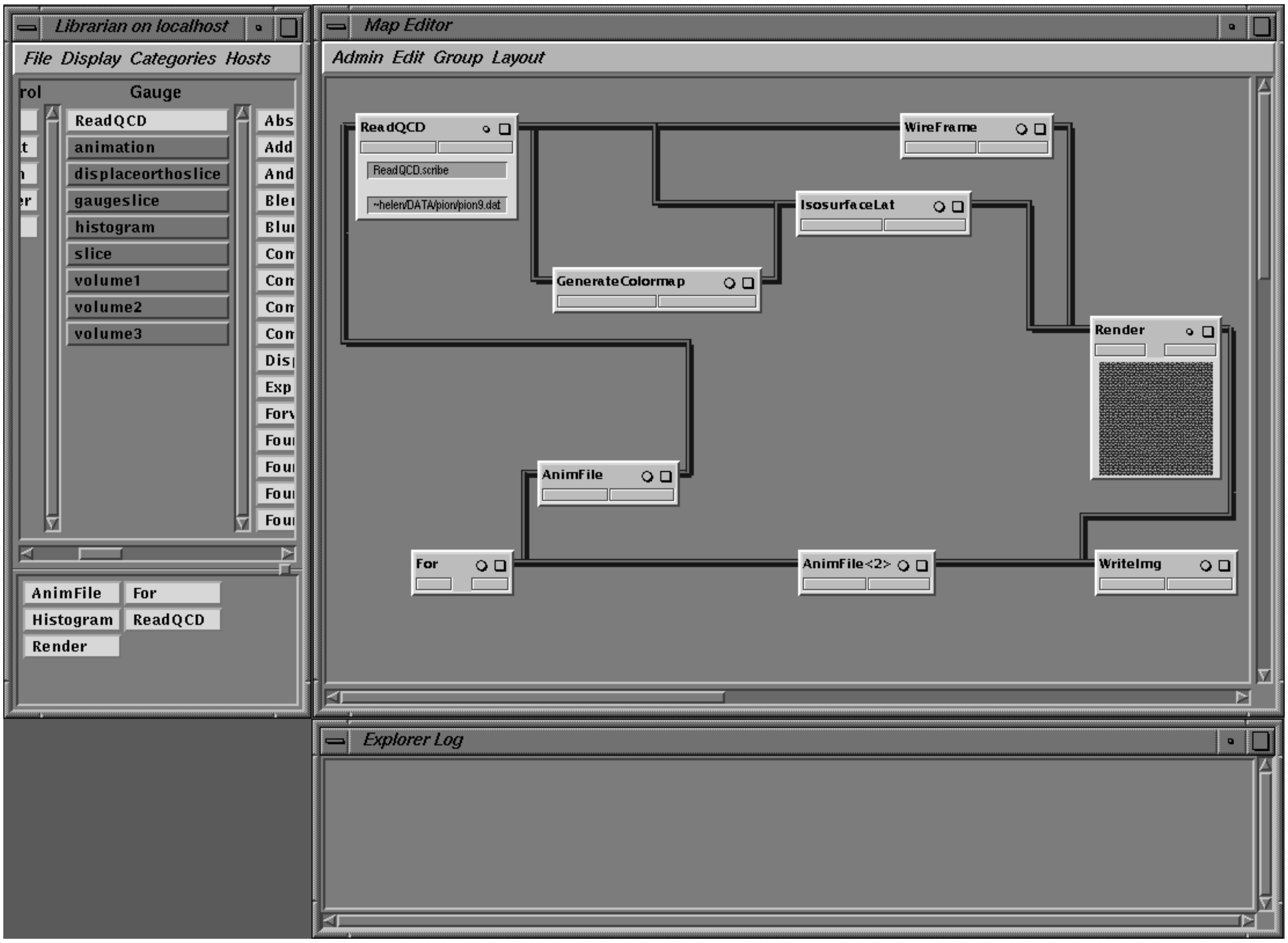

Figure 3: Explorer

- the Librarian (see section 6.1)
- the Explorer Log (see section 6.2)
- the Map Editor (see section 6.3)

6.1 The Librarian

The Librarian contains all the maps and modules available for you to use. Modules appear in beige and maps in blue.

There are some maps specifically for your use in visualising QCD data in lists called *Meson* and *Gauge*. To find these in the librarian use the horizontal slider to move about. The lists are headed *Meson* and *Gauge*.

To locate a map quickly place the mouse in the required list and type the first letter of the module. The list will change to begin at that letter (all lists are in alphabetical order). To start a map move the mouse pointer on top of the map, and drag it onto the Map Editor using the left mouse button. The map might take a few seconds or more to start up. It depends on how many people are using the machine at the time and how complicated the map is.

6.2 Explorer Log

The Explorer Log will tell you when there are errors. The best thing to do if explorer has some errors you don't understand, especially if modules start being deleted, is to `Quit` [1] Explorer and restart.

In general if you understand the errors then do something about them. If you don't, don't worry. . . Explorer tends to crash frequently, especially when experimenting with viewing large data sets, consequently it is advisable to save maps frequently, preferably in the users directory.

6.3 The Map Editor

The map editor is where you will spend most of your time on Explorer as this is where maps are created and modifyied. When you start up a map, or a module, this is where you place it (using the left mouse button and dragging it from the Librarian).

6.4 Dot file for Explorer

To use Explorer the user must have a file called `.explorerrc`. This file contains information about where Explorer has to load up modules and maps from as well as where to place the arena and what arena size is required. There is a `.explorerrc` file contained in `helen` called `DOTexplorerrc` for the user to copy into his/her directory and modify

[1] Quit is in Admin on the Map Editor

Figure 4: Read module

as required. It is recommended to make two copies of this file in the users directory leaving a working master copy.

To access the modules used and maps created the following lines must be present in this file:

```
modulepath -append /home/explorer/for_helen
modulepath -append /home/helen/Modules
modulepath -append /home/helen/Maps
modulepath -append /home/explorer/IRIX5_modules
```

To include the lists of maps in the Librarian the user must have the following lines in the `.explorerc` file:

```
category  "Meson" \
    ReadQCD slice animation volume1 volume2 volume3 splatter orthoslice \
    displaceorthoslice histogram contour isosurface

category  "Gauge" \
    ReadQCD gaugeslice slice animation histogram volume1 volume2 \
    volume3 displaceorthoslice contour isosurface
```

6.5 What are modules?

In Explorer a module is like a programme that can be run within the Explorer environment. Numerous modules have been provided, each performing a different operation: reading in files, slicing data, viewing images, etc. The user can also write their own modules, or add modules which have been provided by others.

All Modules contain the following features:

- the name of the module - this is placed at the top of the module
- a circle in the top right corner - clicking on this with the left mouse button enlarges the module box to allow the user to see the widgets provided
- a square in the top right corner - clicking on this with the left mouse button will open a larger window of the controls (a help option is available in the `help` menu on the control bar of this window)

- a menu - this can be obtained by clicking the right mouse button on the top bar of the module (clicking on the module window produces a different menu). This menu contains things like *refire* which will restart the module, *duplicate*, *disable*, *interrupt* and the *help* menu mentioned above
- two horizontal bars which are positioned underneath the title - these are the input and output ports which are used to connect the modules together to create a *map*.

The `help` option contains a brief explanation of the function of the module as well as the input and output port requirements. A description of the widgets and any known problems and related modules is also included. This option proves useful, especially with new modules, in determining what form the input and output ports require.

Clicking the right button on either the input or output port will give the user a list of connections and what type they must be. Connecting modules together can often be a trial and error task, although experience and looking at maps that have already been created will help.

A data import module called `ReadQCD`, which was written using a package called `dscribe`, has been used in all the maps created to read in the data for QCD visualisation.

6.6 Reading in data

There are various data import modules available in Explorer. All the maps that have been provided for your use (called *Meson* and *Gauge*) use the module `ReadQCD` to read in data. This will read in data which has been through the `convert` program and has had dimensions added at the top of the file (see section 4.6 on *shell scripts*).

There are however, other data import modules. These can be found in the category on the far left of the Librarian, most of them are called `Read????`. Alternatively the user can create their own import module. There is a package called `dcsribe` which is available to help you do this. To start dscribe type:

```
dscribe &
```

The explorer manual [3] and [4] (see section 7) has details of how to use this. When creating modules using dscribe it is recommended to load up a data import module that has already been written , save it in your own directory under a new name, and alter it. It is also advised to parse (option found in the `File` menu in `dscribe`) the file after each alteration to pick up any errors. To test the module try creating a small test data file of the same form. This will save Explorer from *hanging* on you, or dumping the modules, sometimes causing errors in Explorer which make it necessary to `Quit` and restart.

6.7 Using the maps

A map is a collection of modules that have been linked together. The links are represented by coloured pipes (either blue or purple), and these pipes connect the various input and output ports together. Maps are created so that particular parts of the data are visualised in the required way. In a sense it is like writing a programme (the map) to perform a certain task, and creating this programme from a set of routines (the modules). Various maps have been created to carry out specific tasks. Those relevant to visualising QCD data are to be found in the Librarian under Gauge and Meson, there are others under maps in the librarian that are useful for inspiration when creating maps. To use a map drag it onto the Map Editor, the same as starting a module, and wait for it to load. When all the blue pipes (some pipes may be purple) are in place the map is ready to be used. If some of the modules are flashing yellow wait until this has finished.

The following maps can be found in the *Meson* and *Gauge* lists in the *Librarian*.

animation has been set up to record snapshots of the image in the render window for use in a movie. This map should be used as a template for writing images of the form required. Once these images have been recorded there are facilities called `Movie` and `Movieplayer` available on SGI's which can be used to playback the images.

contour displays orthogonal contour slices.

displaceorthoslice takes orthogonal slices (i, j or k axis) or the image. It has also been set up to displace the image, place a wire frame around the data set and place a legend in the render window.

gaugeslice displaces orthogonal slices along with a histogram display showing the distribution of data. Again a legend is provided.

histogram displays the distribution of data along with maximum and minimum values of the data.

isosurface displays an isosurface (a 3-D contour).

orthoslice displays orthogonal slices.

slice can slice the data in any direction. Again this comes with a legend.

splatter renders a full volume display with an isosurface present (a 3-D contour).

volume1 renders a full volume display with legend.

volume2 renders an orthogonal slice of the data and an isosurface (a 3-D contour) in separate render windows — please note that isosurface can take time on large data sets.

volume3 renders an isosurface (a 3-D contour) and a histogram.

Specific ports have been disconnected on each map so that Explorer does not begin loading in a file that does not exist, or begin a loop which may take a long time to execute. Before connecting these ports it is advisable to check the information entered in the widgets on all modules and alter as appropriate, for example check the file name in `ReadQCD` and `Animfile`, and the range of any `For` module.

For use of the **animation** and **gaugeslice** maps the following ports must be connected:

Connect `Animfile` input *value* to the `For` module output *current value* (to enable the for loop to sweep through files). To activate the `For` module connect `Render` *sync* output to the `For` *condition* input.

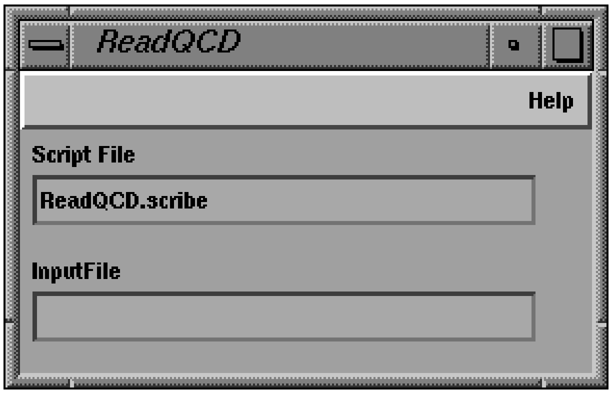

Figure 5: Read module window

For all maps `ReadQCD` needs to be connected to `Animfile`. To do this connect the output port *out file* on `Animfile` to *input file* of the input port of `ReadQCD`. It is not necessary to use `Animfile`, the file name may be entered in directly in the *input file* section of `ReadQCD`, however if many files are to be used which only differ by a number, eg the timeslice, then `Animfile` reduces the entering of filenames. `Animfile` must be used if using a `For` module to sweep through many files.

Please note that `ReadQCD` may still have to be connected. Where not specified this involves connecting the output port of `ReadQCD` to the input–lattice ports of `orthoslice` and `maxmin`.

Once the map is set up click on the square box at the top of the `ReadQCD` module, or if `Animfile` is present this may be used instead. A window will appear (see figure 5.

Leave the scribe part of the window. Click the left mouse button in the input part of the window and type in the path for your file. If in doubt enter the full pathname. Press `return`. If the module does not start to flash yellow within a few seconds click the right mouse button on the top part of the module and select *refire*.

Images are viewed in the `Render` module window. To enlarge the window click on the square on the module with the left mouse button, this is the same for all modules. The image may take a while to appear, please be patient. They are various facilities within the `Render` window. The best way to learn about these is by playing about with them using a simple map and a small data set. Holding the right mouse button down over the image will produce a menu. The most useful facilities are in the option `Functions` are `View all` which will home in on your image, `Set home` and `Home` — these will set a position for your image which can be homed in on if you lose the image. There is also a `decoration` option which iconizes some of these facilities around the edges of the `Render` wondow. For more details of these please refer to the Explorer manual [4] (see section 6.9).

6.7.1 A WARNING about interpreting images

When interpreting the images created in Explorer the user should be aware of the following:

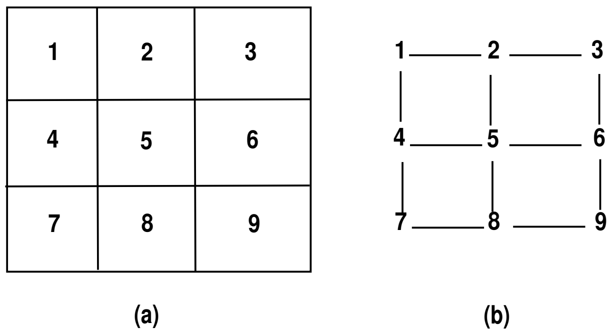

Figure 6: (a) data as read into Explorer, (b) how Explorer interpretes this data

- Explorer blends colours between data values
- for contours and isosurfaces (3-D contours) the data will be interpolated
- the data used to create images is placed at the vertices of the grid, not the centre (see figure 6). This can be a nusaince, and can be eliminated by producing four copies of the data values, giving two horizontally and two vertically.

6.7.2 Modifying maps — an example

This section contains an example of how to modify the `gaugeslice` map to read in the same slice from many files in the `For` loop instead of reading in many slices from the same file. The intention is to give the user an idea of how to make simple modifications. For further help with this please refer to the Explorer manual [4].

Start up the `gaugeslice` map by dragging it onto the map editor with the left mouse button. Wait for it to load, ie. wait until all flashing yellow modules have finished.

Set up the map to move through all the slices in one file as follows:

First input the data for the file name in `Animfile` checking that *out file* is correct. Once this has been done connect *out file* from the output port of `Animfile` to the *InputFile* in the input port of `ReadQCD`. Let this run through and check the image is being displayed correctly. Next check the values that have been entered in the `For` module. Try a small range of numbers initaly. Now connect the *sync – parameter* from the ouput of `Render` to the *condition* input of the `For` loop (this pipe will appear purple). Finally connect the *current value* output of the `For` loop to the *slice number* input of `Orthoslice` and *Refire* the `For` loop (*Refire* is a widget in the `For` window).

The map is now set up to run through the slices in one file.

To alter this to run through the same slice in many files carry out the following operations:

Disconnect the *sync – parameter* output, of the `Render`, from the `For` module. This is done by going to the output port, clicking the right mouse button and holding it down until it is over the connection you wish to disconnect, then let go of the mouse. This will disconnect that link and the relevant pipe should disappear. Now disconnect the

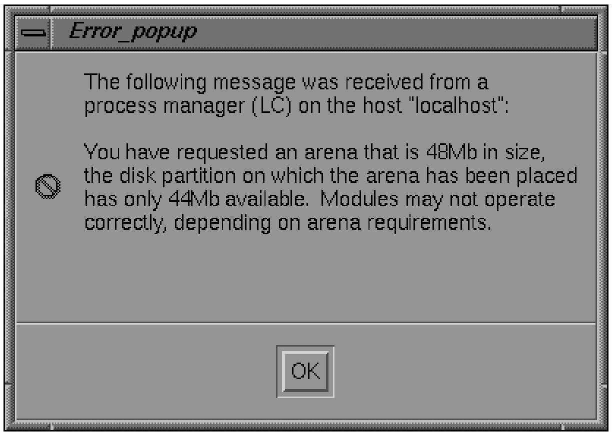

Figure 7: Error message from Explorer

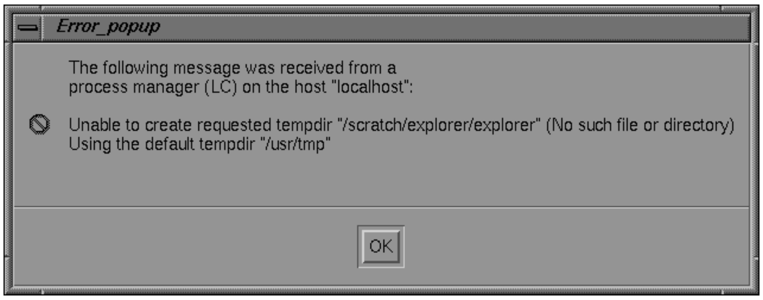

Figure 8: Error message from Explorer

`For` module's ouput of *current value* from `Orthoslice` *slice number* and reconnect the `For` *current value* output to the `AnimFile` input of *value* . Check the values in the For loop and connect up the `Render` *sync – parameter* as before and *Refire*.

6.7.3 Creating your own maps

Maps are best created by using a map that has already been created and adding and deleting modules as necessary. It is recommended that when creating maps small sets of test data are used. Explorer provides a module called `GenLat` which creates data for you. The best way to proceed is to add in modules one at a time and check what is created at each stage. The `help` option on the modules will aid the user in connecting the input and output ports correctly. The user manual [4] has information about maps and some examples on creating simple maps, however through experience the user should be able to create maps of their own with relative ease.

6.8 Some error messages and how to deal with them

The two main error messages that will appear when using Explorer are shown in figures 7 and 8.

Figure 7 is referring to the arena size for Explorer, that is the area where modules, maps etc. will be stored whilst Explorer is being used. Figure 8 refers to a destination which has been specified for the arena in the users `.explorerrc`. Both errors can be ignored in many cases, however a small arena size (15Mb or less) can cause many modules to be dumped by Explorer.

6.9 Further help

There are two manuals for Explorer, the first is a general manual [4], and the second for those interested in writing modules [3].

7 making movies

To make a movie you must fist create some images in Explorer and save these using the `WriteLat` module (see *animation* map in *Meson* or *Gauge*). These images can them be viewed using `movie` or `movieplayer`:

`movie` *filenames* `&`

or

`movieplayer` *filenames* `&`

For example if the files are

```
pion00 pion01 pion02 pion03 pion04
pion05 pion06 pion07 pion08 pion09
```

Then

```
movie pion* &
```

will start movie. A window will appear on the screen and the images will be loaded in one by one. Once this has been done the movie will playback at a faster speed. The left mouse button can also be held down and dragged horizontally across the image to playback the image manually.

These movie can then be put onto video tape for further viewing (for further information contact *EPCC-support*).

8 Summary

Having read this manual and used the facilities present the user should be able to create mesons and view these and the gauge fixing data. When viewing these there are certain types of images the user should be able to obtain. This section contains examples of most of these types of images (this may also aid the user in deciding which maps to use):

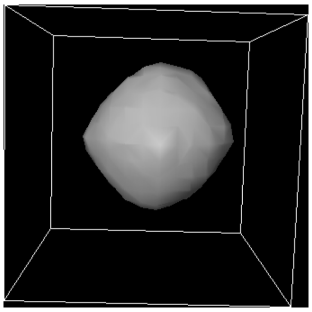

Figure 9: an isosurface of a pion, timeslice 02

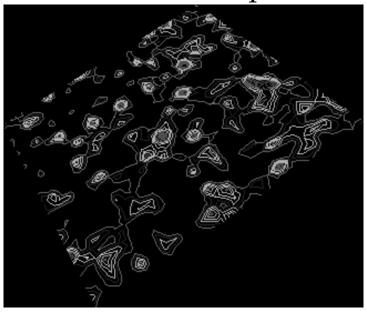

Figure 10: contour of a slice of a gauge dataset

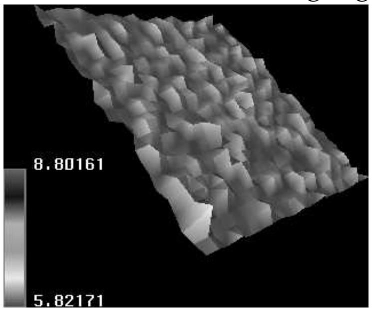

Figure 11: displaced orthogonal slice of a gauge dataset

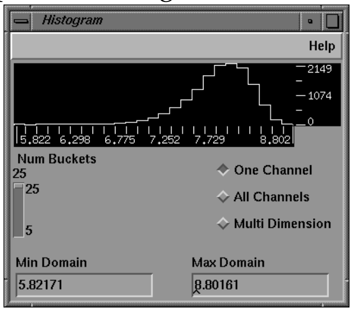

Figure 12: histogram display from a gauge dataset

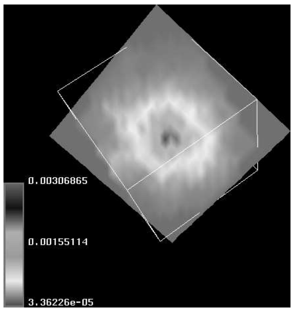

Figure 13: displaced slice of a pion dataset

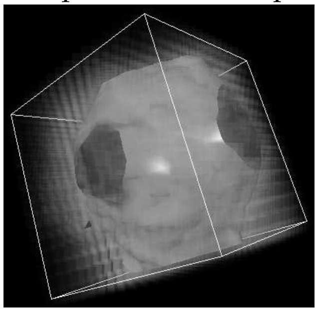

Figure 14: a complete volume with an isosurface from a pion timeslice

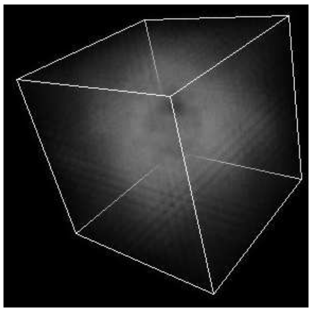

Figure 15: a complete volume of a pion dataset

9 Glossary

angular momentum a property relating to rotational motion

anti-particle a particle which is the same as its respective particle in almost every way except its charge is reversed.

colour a threefold property possessed by quarks, a bit like electric charge, which is believed to be the cause of the strong force.

colour force the force that binds quarks together

electron a negatively charged particle that is one of the constituents of atoms

flavour the property that distinguishes between types of quarks, there are six *flavours*: up, down, strange, charmed, top and bottom

gamma matrix a matrix used to describe the internal organisation of the constituent quarks inside the relevant particle

gauge theory a theory of particles and the forces which act on them

gluon the carriers of the strong force between quarks

lepton a particle that does not experience the strong force, ie. the electron

maps a collections of linked modules in Explorer

meson a particle constisting of a quark anti-quark pair

modules a *building block* within Explorer in which each module performs its own function

Monte Carlo a random statistical method used in many physical simulations

neutrino an electrically neutral, massless lepton of which there are three known varieties one of which partners the electron

pion the lightest known meson of which there are three forms: π^+, π^0 and π^- with charges +1, 0 and -1 respectively

QCD — Quantum Chromodynamics: the theory describing the interaction of colour and electric charges

quark regarded as one of the fundamental building blocks of matter

rho a meson

spin a particle property that is a measure of angular momentum

strong force the force that holds protons and neutrons together. It is believed to result from the force between quarks

UKQCD United Kingdom Quantum Chromodynamics Group

References

[1] Helen Ashton. Visualisation of quantum chromodynamics. *World Wide Web URL:http://www.epcc.ed.ac.uk/ epccssp/project09.html*, September 1994.

[2] Ken Chin-Purcell. *All about BOB - A tool for browsing 3D data sets*. Minnesota Supercomputing Center Inc.

[3] Margaret-Anne Halse. *IRIS Explorer ^{TM} Module Writer's Guide Guide*. Silicon Graphics Inc., 1992.

[4] Margaret-Anne Halse. *IRIS Explorer ^{TM} User's Guide*. Silicon Graphics Inc., 1992.