

CUBISM User Manual

IRS Spectral Map Analysis and Reduction
Edition 1.8, June, 2011



by J.D. Smith, Lee Armus, Brent Buckalew, Danny Dale,
George Helou, Helene Roussel & Kartik Sheth

This is edition 1.8 of the *CUBISM User Manual* for CUBISM version 1.8, June, 2011.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with the Front-Cover texts being “A GNU Manual”, and with the Back-Cover Texts as in (a) below. A copy of the license is included in the section entitled “GNU Free Documentation License” in the Emacs manual.

(a) The FSF’s Back-Cover Text is: “You have freedom to copy and modify this GNU Manual, like GNU software. Copies published by the Free Software Foundation raise funds for GNU development.”

This document is part of a collection distributed under the GNU Free Documentation License. If you want to distribute this document separately from the collection, you can do so by adding a copy of the license to the document, as described in section 6 of the license.

Table of Contents

1	Introduction	1
2	Installation	2
2.1	Source Installation	2
2.2	Binary Installation	3
2.3	Setup	3
2.4	Memory Requirements	4
2.5	Running	4
2.5.1	Running the Source Distribution	4
2.5.2	Running the Binary Distribution	4
2.6	Upgrading	5
3	Quick Start Guide	6
3.1	Example Data Set	6
3.2	Building Your First Cube	7
3.3	Visualizing the AORs	7
3.4	Navigating the Cube	8
4	The Tools	9
4.1	Tool Inter-Communication	10
4.2	General Interface Tips	10
4.2.1	List Selection	10
4.2.2	Scrolling	10
4.2.3	Mouse shortcuts	11
4.2.4	File Selection	11
4.3	CUBISM Project	13
4.3.1	Project Title	14
4.3.2	Menus	14
4.3.2.1	File Menu	14
4.3.2.2	Edit Menu	15
4.3.2.3	Record Menu	15
4.3.2.4	Cube Menu	17
4.3.2.5	Background Menu	18
4.3.2.6	BadPix Menu	19
4.3.2.7	Info Menu	19
4.3.2.8	Help Menu	19
4.3.3	Data Records	20
4.3.3.1	Record Info	20
4.3.3.2	Record Enabled State	21
4.3.3.3	Record Data Types	21
4.3.3.4	Operating on Records	21
4.3.4	Status Bar	22

4.3.5	Button Bar	22
4.4	CubeView	23
4.4.1	Title Bar	23
4.4.2	Menus	23
4.4.2.1	File Menu	23
4.4.2.2	Options Menu	23
4.4.2.3	Tools Menu	24
4.4.3	CubeView Tools	24
4.4.3.1	Tool Interaction	26
4.4.3.2	Box Regions	26
4.4.3.3	Tool Overview	26
4.4.3.4	Zoom Tool	27
4.4.3.5	Histogram Tool	28
4.4.3.6	Color Tool	28
4.4.3.7	Image Slicing Tool	28
4.4.3.8	Box Statistics Tool	30
4.4.3.9	Aperture Photometry Tool	32
4.4.3.10	Cube Extraction Tool	33
4.4.3.11	Pixel Backtracking Tool	33
4.4.3.12	Bad Pixel Tool	34
4.4.3.13	AOR Visualization Tool	37
4.4.3.14	Pixel Table Tool	38
4.4.3.15	Order Mask Tool	38
4.4.3.16	Compass Rose Tool	38
4.4.4	Colorbar	38
4.4.5	Status Display Line	38
4.4.6	Bad Pixel Codes	38
4.4.7	Image Info Block	39
4.4.8	WAVSAMP Pane	39
4.5	CubeSpec	40
4.5.1	Title Bar	40
4.5.2	Menus	41
4.5.2.1	File Menu	41
4.5.2.2	Maps Menu	41
4.5.3	Buttons	42
4.5.4	Display Panes	43
4.6	CubeSpec Plot Window	43
5	Cube Assembly	44
5.1	Input Files	44
5.2	Build Order	45
5.3	Calibration Sets	46
5.4	Backgrounds	46
5.4.1	In Situ Background	47
5.4.2	Archive Background	47
5.4.3	Background Blend	48
5.4.4	1D Sky Spectrum	48
5.4.5	Combined 1D and 2D Background	49

5.4.6	No Background	49
5.4.7	Background Selection Using Visualization	50
5.4.8	Saving the Background List	50
5.5	Bad Pixels	50
5.5.1	Global and Record Level Bad Pixels	51
5.5.2	Manual Bad Pixel Selection	51
5.5.3	Backtracking to Discover Bad Pixels	52
5.5.4	Automatic Bad Pixels	53
5.5.5	Saving Bad Pixels	55
5.6	Cube Build Settings	55
5.7	WAVSAMP	56
5.8	Building the Cube	56
5.9	QuickBuild	57
5.10	Saving the Project	57
5.11	Saving the Cube as FITS	59
5.12	Reading a FITS Cube	59
6	Cube Analysis	60
6.1	Extracting 1D Spectra	60
6.1.1	Direct Extraction	60
6.1.2	Matched Region Extraction	60
6.2	Creating 2D Maps	62
6.2.1	User-defined Maps	62
6.2.2	Pre-defined Maps	63
6.2.3	Map Sets	64
6.2.4	Redshift and Maps	64
6.2.5	Integrated Maps	64
6.2.6	Wavelength Weighting	64
6.2.7	Line Fits	64
6.3	Complex Maps	65
7	Tips and Troubleshooting	66
7.1	Mouse and Keyboard Shortcuts	66
7.1.1	Cube Project	66
7.1.2	Cube View	66
7.1.3	Cube Spec	69
7.2	Tips	69
7.3	Troubleshooting	70
7.4	Debugging CUBISM	73
	Index	74

1 Introduction

CUBISM, the CUbe Builder for IRS Spectral Mapping, is a package which supports the analysis and reduction of spectral maps created with the [IRS Spectrograph](#) aboard the [Spitzer Space Telescope](#). It is written in the Interactive Data Language (IDL). CUBISM is designed to allow sets of basic calibrated data (‘BCDs’) from IRS mapping observations to be combined into single 3D spectral cubes, with two spatial and one spectral dimension. From these cubes, spectra can be extracted over differing apertures, and arbitrary maps can be made in spectral features (e.g. a continuum-subtracted line image).

CUBISM was developed at the University of Arizona by the Spitzer Infrared Nearby Galaxies Survey (SINGS) team¹, in partial fulfillment of its Spitzer Legacy commitments. It is distributed and supported by the Spitzer Science Center at Caltech, Pasadena.

CUBISM consists of three main components, which together form the core of its analysis and reduction capabilities:

CUBISM Project	Manage ‘BCD’ data, and track all the various information required to build cubes, including calibration data, background information, bad pixels, aperture information, etc.
CubeView	General purpose viewer with a variety of tools for interacting with 2D spectral images, full spectral cubes, and visualization overlay FITS images.
CubeSpec	View and manipulate extracted spectra, and create maps from spectral cubes.

CUBISM is not a general purpose spectral analysis tool. The tools it offers are oriented directly towards the task of creating, validating, and analyzing spectral cubes. Since individual spectra, spectral maps, and full spectral cubes can be output from a single cube project, other tools can easily be used for higher order analysis of these outputs (e.g. multiple Gaussian fitting, etc.).

This manual is organized as follows. After discussing the installation and requirements of CUBISM in [Chapter 2 \[Installation\]](#), [page 2](#), we give a quick start guide to building a spectral cube from an example mapping data set in [Chapter 3 \[Quick Start Guide\]](#), [page 6](#). We then cover in detail the menu options and capabilities of the three main tools which comprise CUBISM (see [Chapter 4 \[The Tools\]](#), [page 9](#)). Then we discuss in greater depth the steps required to build a cube in [Chapter 5 \[Cube Assembly\]](#), [page 44](#), explain methods of analyzing the cube (see [Chapter 6 \[Cube Analysis\]](#), [page 60](#)), and finish with common tips and troubleshooting (see [Chapter 7 \[Tips and Troubleshooting\]](#), [page 66](#)).

Note that this manual does not include information on planning IRS spectral mapping observations; see the [IRS Spectral Map HOWTO](#) for information on observation planning.

¹ CUBISM was based in part on the SCOREX analysis software developed by JD Smith at Cornell University; see Smith, J.D.T. PhD Thesis, 2001.

2 Installation

CUBISM runs under IDL, and requires a working version of IDL to function. There are two means of installing CUBISM: as a set of source `.pro` files which IDL finds on its search path, or as a pre-compiled binary, which can be loaded as a single entity. Both versions can be found on [the CUBISM home page](#).

The advantages of a source installation are:

- Access to CUBISM's code for bug fixing or examining algorithms.
- Source level feedback when bugs occur (see [Section 7.4 \[Debugging CUBISM\]](#), page 73).
- Should continue to function with all future versions of IDL.

The disadvantages of the source installation are:

- Must install the `AstroLib` dependency library, at the required version. Some (small) risk of future changes in `AstroLib` causing problems.
- Subject to routine name conflicts (e.g. two routines named `'routine.pro'` on the path), so you must carefully set your `IDL_PATH` to be sure CUBISM's files are found first. This should not be a common problem, but users of the staring-mode extraction package SMART may experience routine name conflicts (though not for recent versions; see [Section 2.1 \[Source Installation\]](#), page 2).

The advantages of installing and running a binary version of CUBISM:

- All required routines are included in the compiled file at the required version; no need to install external libraries.
- No routine name conflicts should occur.
- Can be used (sans command line) with the freely available *IDL VM*, if you don't have access to an IDL license.

and the disadvantages:

- No access to source code level debugging feedback when errors occur, making it harder to track down problems.
- More closely tied to the IDL version; a given binary may not work with all future versions of IDL (though typically binary compatibility between IDL versions within a few versions is quite good).

2.1 Source Installation

The requirements for installing and running CUBISM from source are:

1. A Linux/Solaris/Unix or MacOSX platform. CUBISM may run under the Windows operating system, but has not been tested on a Windows platform.
2. A licensed copy of IDL, version 7.1 or later. Download [from RSI \(now ITTVIS\)](#).
3. The `AstroLib` library, available from [NASA Goddard](#). Be sure to include it on your `IDL_PATH`.
4. A compiler for C source, typically `gcc`, or whatever the IDL routine `MAKE_DLL` looks for (usually available by default).

The compiler is required to auto-compile a small piece of C code used to speed-up the main cube building algorithm. If this compilation fails, an IDL version of this algorithm will be used, which gives the same results, but operates more slowly.

To install CUBISM from source:

1. Unpack the ‘`cubism_vX.XX_src.tgz`’ file (where ‘`X.XX`’ is the version number) in a directory on the IDL path (e.g. ‘`~/idl`’).
2. Ensure the ‘`irs_cubism`’ directory which is unpacked is on the `IDL_PATH`, e.g. by:


```
setenv IDL_PATH <IDL_DEFAULT>:+$HOME/idl
```

System-wide installation is also possible: just install CUBISM in a location accessible by your entire group.

SMART USERS:

Users of SMART may experience file name conflicts with CUBISM, in particular in an IDL session started by SMART. As of version 6.2.4, SMART has been modified to avoid such conflicts; upgrading to this version or later is highly recommended.

2.2 Binary Installation

The requirements for CUBISM running as a binary:

1. A Linux/Solaris/Unix or MacOSX platform. CUBISM may run under the Windows operating system, but has not been tested on a Windows platform.
2. A copy of IDL at version 7.1 or later. This can either be a fully licensed copy, or the *IDL VM*, the freely available virtual machine. Download either [from ITTVIS](#).
3. A compiler for C source, typically `gcc`, or whatever the IDL routine `MAKE_DLL` looks for (usually available by default).

Note that running CUBISM from the binary ‘`.sav`’ file under the free *IDL VM* does not give you access to an IDL command line, so that only the graphical interface to CUBISM is accessible. With this setup, no analysis can be performed at the command line, though all files, including spectra, maps, and cubes can be output as normal. Running the binary distribution of CUBISM in a licensed version of IDL does not prevent access to the command line.

To install the binary version of CUBISM, simply unpack the ‘`cubism_vX.XX_bin.tgz`’ file (where ‘`X.XX`’ is the version number) in a directory on the IDL path (or anywhere else).

2.3 Setup

CUBISM needs very little setup. As long as the binary or source directory structure is left intact, CUBISM automatically discovers all the necessary calibration and other files needed.

One basic setup issue relates to the color mode. By default, IDL uses *DECOMPOSED* color, in which the RGB value of pixels is directly specified, whereas CUBISM (and most astronomy software) relies on color table indices to specify color. To switch modes, try adding the following to the IDL startup file specified with the environment variable `IDL_STARTUP`:


```
device,DECOMPOSED=0,TRUE_COLOR=24,RETAIN=2
```

You may not need the `RETAIN=2` setting depending on your window manager (this forces IDL to keep track of the contents of windows when they need to be redrawn, and is typically required under Linux). The binary distribution of CUBISM performs this operation by default.

Another potential issue relates to the `IDL_PATH`. If you have a source distribution of CUBISM, you will need to ensure that the directories containing CUBISM as well as the `AstroLib` installation are on your `IDL_PATH`. In principle it should not matter where on the path CUBISM is. However, occasionally two different packages will each define the same routine in two files with the same names (a so-called *name space conflict*). If you encounter this problem, move the directory containing CUBISM higher on your `IDL_PATH`, or use the binary distribution, which doesn't suffer such name space conflicts.

2.4 Memory Requirements

CUBISM is designed to take advantage of the large memory sizes available on modern computers. Rather than load a small amount of data from disk, operate on it, and return it to disk (e.g. as IRAF might do), CUBISM attempts to keep most data in memory, which enables a variety of features which would not be possible with caching to disk.

For modest sized cubes (with fewer than 100 BCDs contributing), this is not a burden. For very large cubes (many hundreds to many thousands of records), CUBISM's use of roughly 0.5–2MB per record (at maximum) requires at least 1–2GB of RAM to avoid loss of performance. CUBISM can work with large projects using less memory, but performance will suffer dramatically as data is paged to disk. Note that data is loaded on demand, so for instance, viewing a pre-built cube without reloading record data will consume only a small amount of memory.

2.5 Running

How CUBISM is run depends on how it was installed.

2.5.1 Running the Source Distribution

For source distributions, after installation on your IDL path, simply type:

```
IDL> cubism
```

and you are prompted to select an existing saved cube project, or create a new one.

2.5.2 Running the Binary Distribution

Running the pre-compiled binary version of CUBISM can be accomplished by putting the `'cubism_vm.sav'` file in your IDL path and using:

```
IDL> cubism_vm
```

Another option allows you to run this binary file from anywhere, not necessarily on your IDL path:

```
IDL> restore, '/path/to/cubism/bin/cubism_vm.sav'
IDL> cubism
```

You can also run the compiled file in the free IDL VM, which does not require an IDL license:

```
% idl -vm=/path/to/cubism_vm.sav
```

MacOSX users can accomplish the same thing by optionally using the precompiled wrapper ‘Cubism.app’ application: simply double-click (or double-click a ‘.cpj’ CUBISM project file). Note that even under OSX, CUBISM runs as an X11 application, and that ‘Cubism.app’ is simply a wrapper to start IDL in the IDL Virtual Machine and load CUBISM.

2.6 Upgrading

Upgrading CUBISM is simple and requires replacing the source or binary installation directories with the newer version and restarting IDL. You can always find out what version of CUBISM you are running with the menu *Help->About Cubism* in the project window. New versions of CUBISM are available at <http://sings.stsci.edu/cubism>.

3 Quick Start Guide

CUBISM has two major functions: building cubes from IRS Spectral Mapping data sets, and analyzing those cubes. Building cubes from collections of mapping data sets is in principle a simple process: the correct ‘BCDs’ are collected together in a project, the cube build parameters are adjusted, and the cube is built. Further refinement of the cube includes identifying bad pixels to exclude from the cube build, selecting and creating the appropriate background frames, configuring the calibration details and cube build options, etc. Although straightforward, this process can take a good deal of time. Users should expect to spend between one-half and one hour per IRS slit, building and cleaning their cubes (flagging and removing bad pixels, etc.).

With the assembled cube, you can perform a variety of types of analysis, including extracting 1D spectra in specific regions and building 2D spectral maps. There are many details which can affect the cube assembly process, and impact the quality of the final assembled product. Here we will quickly go through the basic steps required to build a cube, extract a spectrum, and build a spectral map, leaving all of the (important) details aside for now. These details are covered later in this manual.

An important note before we begin concerns the scope of a CUBISM project. Each CUBISM project pertains to a single IRS slit, e.g. Long-Low order 1 (LL1), Short-Low order 2 (SL2) or Short-High (SH). Each of the low-res slits are a single spectral order, but the high-res slits have 10 orders, each. Just as a reminder, there are four IRS modules (SL, LL, SH, and LH), with each low-res module containing two slits and each high-res module containing one slit. For data sets with mapping observations in all four low-resolution orders (SL1, SL2, LL1 and LL2), four individual cubes will be built, and analyzed as a set. A full low-res and high-res IRS map will therefore produce six spectral cubes (the four for low-res and two additional for SH and LH) and there will be six CUBISM project (‘.cpj’) files.

3.1 Example Data Set

We have assembled a small example data set, using data from the SINGS IRS Short-Low order 1 (SL1) spectral mapping observations of nearby galaxy NGC 3049. The example data set should be available where you obtained this manual. Unpack it, and follow along. The set includes the following:

‘data/’ A directory containing all of the data files.

‘ngc3048_irac_8um.fits’
 A 8 micron IRAC image of the galaxy.

‘ngc3049_sl1.bgl’
 The background record list.

‘ngc3049_sl1.bpl’
 The bad pixel list.

‘ngc3049_SL1.cpj’
 The pre-built cube project.

‘ngc3049_SL1_cube.fits’
 The pre-built cube as a 3D FITS image.

```
'ngc3049_SL1_cube_unc.fits'
```

The pre-built cube uncertainties as a 3D FITS image.

3.2 Building Your First Cube

Here we take you through the process of assembling a cube with the pre-packaged data set.

To get going, start Cubism (see [Section 2.5 \[Running\]](#), page 4):

```
IDL> cubism
```

You can either *Create New Cube Project* to build it from scratch, or skip directly to the “answer key”, choosing the pre-built cube project file ‘ngc3049_SL1.cpj’ bundled in the example, moving ahead to the next section.

Once you have chosen to create a new cube project, give it a useful name when prompted, like ‘NGC3049 SL1’. You’ll be presented with a blank CUBISM Project window. Read in the full example data set by clicking on the *Import AOR* button at the bottom. Select the ‘data’ directory. CUBISM will search for all IRS data files at or beneath that directory, and group all discovered data files by object and mapping AOR, allowing you to select among them. In this case, only two AORs are found, one for SL1, and one for SL2. Select them both to load data from both mapped orders, and all the relevant files will then be loaded into the project. When you are warned that no calibration set has been loaded, and that the latest is being used, simply acknowledge the warning.

Normally, you would now choose background records among the record set (see [Section 5.4 \[Backgrounds\]](#), page 46), and select bad pixels (see [Section 5.5 \[Bad Pixels\]](#), page 50). In this case, we have pre-built the background record list and bad pixel list for you. Specify the background ‘BCD’ records to be used by choosing ‘Background->Load Background Recs...’, and loading ‘ngc3049_sl1.bgl’, choosing *Average + Min/Max Trim* when prompted. Similarly, load the prepared bad pixels using *BadPix->Load Bad Pixels...*, choosing ‘ngc3049_sl1.bpl’.

Choose *Edit->Select All* and then *Record->View Record Stack...* to pop-up a viewer displaying an average stack of all the records. In the viewer, choose *Tools->Scale Image with Histogram*, and draw a scaling box on the dark part of the image at left by clicking and dragging from top-left to bottom right. You should see two spectral orders appear with an obvious spectrum in the center. Move the scaling box around by click and drag to highlight different features.

Back in the main CUBISM Project window, hit the *Build Cube* button, and watch the build progress. You should see the cube build feedback in a window which opens. Click the *Save* button and save the project to disk as a ‘.cpj’ file. Congratulations, you have completed your first spectral cube.

3.3 Visualizing the AORs

Before you move on to examine the cube, visualize the spectral mapping AOR by choosing *Record->Visualize AORs...* Select the ‘ngc3048_irac_8um.fits’ file, and the viewer should fill with an image and some yellow outlines of the AOR regions. Click on the histogram scaling tool button (second from left) and drag to create a histogram scaling box to bring the galaxy into view. Click on the zoom tool (leftmost), and draw a box around these record outlines to zoom in and examine the records overlaid on the 8 micron image.

3.4 Navigating the Cube

Now that the cube is built, hit the *View Cube* button to display the cube in the same viewer window. In this window, click on the *Extract* button (a cube with a line through it), or select *Tools->Extract Spectra and Stack Cubes*. Click and drag a rectangular extraction region, from upper left to lower right, on the cube, and the CubeSpec window will appear showing you the extracted spectrum. Center the purple box on the galaxy seen in the right hand side of the cube.

In the CubeSpec window, click *Map* and then choose ‘**Region: Peak**’. Click and release at 11 micron and then 11.8 micron to define a peak region highlighting the 11.3 micron PAH feature. This will then be highlighted in red on the spectrum, and the image in the CubeView window will change showing a map of NGC 3049 averaged over the wavelength region you have selected.

Congratulations, you have just created your first spectral map with CUBISM. By choosing ‘**Region: Continuum**’, and setting continuum regions on either side of your peak region, you could now create a continuum-subtracted spectral map. Read on to learn more about the detailed steps for cube assembly and analysis.

4 The Tools

CUBISM consists of three main components, which are used together for building and analyzing spectral cubes: CUBISM Project, CubeView, and CubeSpec. We will give here a description of the purpose and reference of the features of each of these three tools, after some general tips on using the interface, and background on the communication paths among the tools.

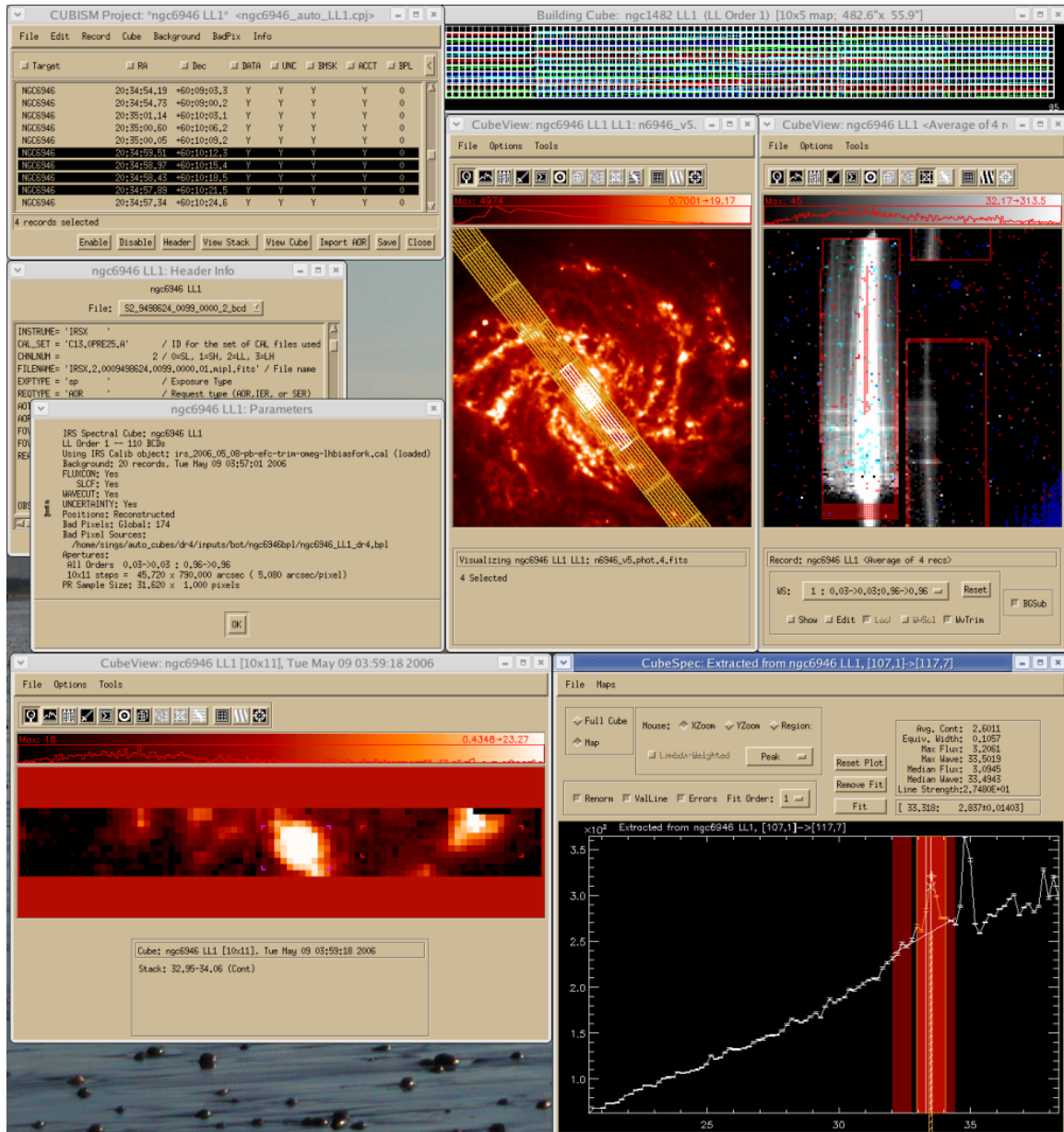


Figure 4.1: The CUBISM tools being used together.

An example of all the CUBISM tools being used together on a single project can be seen in [Figure 4.1](#). Seen in the window are, counter-clockwise from top-left: a CUBISM Project window, the cube build feedback plot, a CubeView tool visualizing the AOR and four selected records, a CubeView tool showing the average of 4 records, with bad pixels overlaid, the CubeSpec tool displaying an extracted spectrum and a 3 component line map, a CubeView window showing a line map image built from the LL order 1 cube, cube build parameters, and header information pane.

4.1 Tool Inter-Communication

The tools which comprise CUBISM constantly communicate with each other, so that updates in one are immediately reflected in the others. For example, when selecting bad pixels (see [Section 4.4.3.12 \[Bad Pixel Tool\], page 34](#)), the bad pixel list for the associated CUBISM Project is immediately updated, or if a new list is loaded from file, the viewer is updated. Similar high level communication occurs between all of the sub-tools, so that you can regard a CUBISM project and all its associated windows and tools as a single, consistent state.

4.2 General Interface Tips

The interactive component of CUBISM is based on IDL widgets, which, at least on the Unix/OSX systems where CUBISM is tested, are derived from the Motif widget set. As such, a variety of common keyboard and mouse shortcuts are available which can simplify interactive operations.

4.2.1 List Selection

In any list (e.g. the CUBISM Project window, see [Section 4.3 \[CUBISM Project\], page 13](#)), individual list elements can be selected by clicking with the left mouse button. A range of elements can be selected by click-dragging. Alternatively, the first element in a range can be selected, and then the last element clicked while holding the SHIFT key. Non-contiguous regions can be selected by holding the CONTROL. These methods can be combined, e.g. to select two non-adjacent regions, click the first, scroll to and SHIFT-click the second, scroll to and CONTROL-click the third, and finally SHIFT-click the last. The arrow keys scroll the selection up or down, and the PAGE UP and PAGE DOWN keys skip entire pages full of list items.

4.2.2 Scrolling

Though not required, list items can conveniently be made to respond to mouse scroll wheel input by modifying a set of resources associated with the X11 windows environment. To do so, add the following to your ‘~/Xdefaults’ file:

```
*XmList.baseTranslations:      #augment Shift<Btn5Down>: ListNextItem()\n\
                               Shift<Btn4Down>: ListPrevItem()\n\
                               <Btn5Down>: ListNextPage()\n\
                               <Btn4Down>: ListPrevPage()\n

*XmScrollBar.baseTranslations: #augment <Btn4Down>: IncrementUpOrLeft(0) IncrementUpOrLeft(1)\n\
                               <Btn5Down>: IncrementDownOrRight(0) IncrementDownOrRight(1)\n

*XmText.baseTranslations:     #augment Shift<Btn4Down>: page-left()\n\
```

```
Shift<Btn5Down>: page-right()\n\  
<Btn4Down>: scroll-one-line-up()\n\  
<Btn5Down>: scroll-one-line-down()\n
```

and restart your X environment.

4.2.3 Mouse shortcuts

Double-clicking on a list item often results in some action being performed on that item (e.g., view a record in the CUBISM Project window). The same effect results when RETURN is pressed with a selected item. Using the arrow keys and RETURN is a quick way to go through a list viewing each item.

In graphics windows (e.g. the CubeView window, see [Section 4.4 \[CubeView\], page 23](#)), mouse presses perform specific actions depending on context. Typically the left-mouse button performs the default options. See the documentation for these tools for more information.

4.2.4 File Selection



Figure 4.2: A file selection dialog.

CUBISM uses a specialized file selection tool throughout, which has some common features. An example dialog is shown in [Figure 4.2](#). A list of wildcard filters is included relevant to the file being selected (for opening or saving), and new filters can be added. Click on the *Filter* button for the current list. *Show All* shows all files, independent of the filter. Directories are at left; double-click to navigate through them. Arbitrary directory paths can be entered at the top. Files in the current directory are listed at right; click to select or double-click to accept and return. To the left of the **Directories** and **Files** header words are two small icons, which, when clicked pop up a list of recent directory or file selections which can be selected.

4.3 CUBISM Project

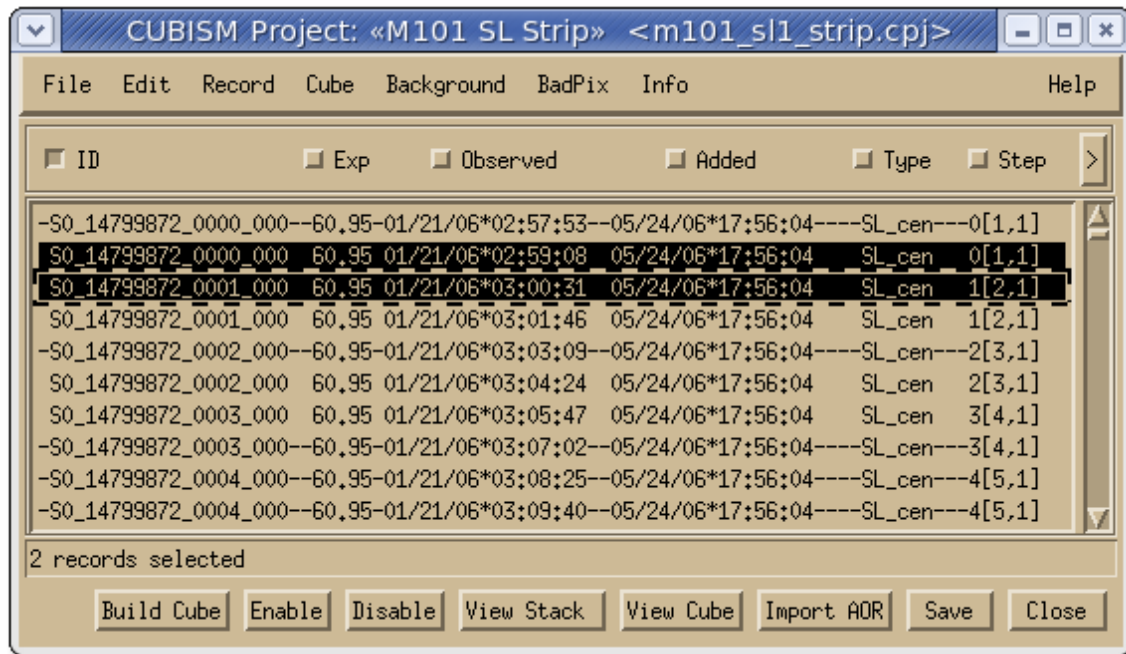


Figure 4.3: CUBISM Project Main Window, with selected and disabled records.

The CUBISM Project is the central storehouse of all information relating to a single spectral cube. This is where the raw spectral data are collected, the calibration parameters are loaded and managed, preferences are set, the cube is assembled, and outputs are saved.

ONE PROJECT PER ORDER:

A single CUBISM project contains information for only one spectral cube, corresponding to a single IRS module and order.

This includes sub-slits for single IRS modules, e.g. SL1 and SL2 would be two separate cubes. There is a separate project window for each open cube. For information on how to extract spectra from matched areas in multiple overlapping cubes, see [Section 6.1.2 \[Matched Region Extraction\]](#), page 60.

CUBISM projects, with all their associated meta-data, can be saved to and recovered from disk, with the default file extension '.cpj' (for "Cubism Project"). In a sense, the "project" is the fundamental file type of CUBISM, and can be manipulated in a similar way as a "document" in other applications (Open/Close/Save/Revert/etc.). You can have as many CUBISM projects open at once as your memory will allow (though all the windows associated with a given project can quickly overwhelm your screen).

Any given CUBISM project can be read from and saved to disk, manipulated from the command line, or interacted with via the GUI interface. Internally, and on disk, a full CUBISM project is a single IDL object, which contains a rich nested hierarchy of data and

other information. Typically, you interact with a project via the graphical interface, though it can be manipulated directly from the command line as well. The figure above shown an example CUBISM Project window, populated with a mapping data set.

4.3.1 Project Title

The title bar of the CUBISM Project encodes the name of the project, surrounded by double arrow brackets, if there are unsaved changes, as well as the file name the project is saved to in '<>' (angle brackets), or '<(unsaved)>' is the project is not yet saved. Note that the file name of a saved project, and the project name can be distinct, though by convention they are usually kept the same.

4.3.2 Menus

Most of the options for assembling and saving information from the cube are available from the menus of CUBISM Project, which are documented here in order of their appearance.

4.3.2.1 File Menu

- New... Create a new CUBISM project, prompting for the name.
- Open... Open an existing CUBISM project, or a CUBISM FITS cube for display.
- Save Setup
 - Set the options for saving projects.
 - Save Data with Project
 - Save all record data with the project (which can greatly increase the project size on disk).
 - Relative File Names
 - Modify record filenames to be relative to the project path. Useful for distributing a project file and data set together, without including the record data in the project itself.
 - Save Clip Accounts with Project
 - Save all clipping information, which enables pixel backtracking, and rapid cube rebuilds, with the project. This will increase the project size on disk.
- Save Save the project, prompting for a file to save to if not already set.
- Save As... Save the project as an alternate file.
- Revert to Saved... Recover the last saved version of the project from disk, discarding current changes.
- Write FITS Cube... Write out the assembled cube (if any) as a FITS cube, along with an associated uncertainty cube (if built).
- Rename Project... Rename the project (which does not affect the file to which the project is saved).

Export to Command Line...

Export the full CUBISM Project object to the IDL command line, prompting for the variable name.

Load New Calibration Set...

Load an alternate calibration set.

Close

Close the project window, prompting to save if changes have been made.

4.3.2.2 Edit Menu

Select All

Select all the records.

Select By Filename

Select records with files matching a given expression.

Select By Keyword

Select records with keyword matching a given expression.

Invert Selection

Select an non-selected records, and de-select all selected records.

Deselect Disabled

Deselect any selected record which is disabled.

Replace File Substring...

Replace a given substring in the file names of the selected records with another string.

4.3.2.3 Record Menu

Add Data...

Add individual data records. Useful for data with filenames or organization not following the convention. Usually not as convenient as adding full AOR data sets at a time using one of the following commands.

Import Data from Mapping AOR

Import data from entire mapping AORs, prompting for selection among the AORs found at or below the selected directory.

BCD... Import BCD data (flat-fielded and straylight-corrected).

DroopRes...

Import DroopRes data (not flat-fielded, not straylight-corrected).

FlatAp...

Import FlatAp data (flat-fielded, not straylight-corrected).

Import Data by module

Import all records found at or below the selected directory, grouped by module (independent of any AOR). Can be useful for loading archive backgrounds which aren't grouped into mapping AORs see [Section 5.4.2 \[Archive Background\]](#), [page 47](#).

- BCD... Import BCD data (flat-fielded and straylight-corrected). The default data type CUBISM uses.
- DroopRes... Import DroopRes data (not flat-fielded, not straylight-corrected).
- FlatAp... Import FlatAp data (flat-fielded, not straylight-corrected).
- Load Record Masks**
 - Load the BMASKs associated with any newly added records.
- Load Record Uncertainties**
 - Load the BCD uncertainty files for any newly added records.
- Switch Record Data Type...**
 - Prompt for and switch the data type of the selected records among 'BCD', 'FlatAp', and 'DroopRes'.
- Restore All Record Data**
 - Recover from file the data for all records (these are normally recovered on demand).
- View Record/View Stack...**
 - View the selected record, or an average stack of records if more than one is selected, using a pre-existing viewer window if available. Note that double-clicking a record or hitting RETURN on a selected record has the same effect.
- View Record (new viewer)/View Stack (new viewer)...**
 - View the selected record, or an average stack of records if more than one is selected, using a new viewer window.
- View Uncertainty...**
 - View the associated uncertainty image of the selected record, or the quadrature sum of uncertainties if more than one is selected, using a pre-existing record viewer window if available.
- View Uncertainty (new viewer)...**
 - View the associated uncertainty image of the selected record, or the quadrature sum of uncertainties if more than one is selected, using a new viewer window.
- Delete** Delete the selected record(s) from the project.
- Rename** Change the ID of the first select record.
- Disable** Disable the selected record(s), preventing them from being built in the cube.
- Enable** Enable the selected record(s), allowing them to be built in the cube.
- Show Filenames...**
 - Show the filenames associated with the selected record(s).
- Show Header...**
 - Show the header(s) of the selected record(s).
- Show Keyword Value(s)...**
 - Show the value(s) of a selected FITS header keyword for the selected record(s).

Visualize AORs...

Load a FITS image and visualize the mapping AORs on it, using a pre-existing visualization viewer window if available. Reuse any image which has already been loaded. See also [Section 4.4.3.13 \[AOR Visualization Tool\], page 37](#).

Visualize AORs (new viewer)...

Load a FITS image and visualize the mapping AORs on it in a new viewer window.

Load New Visualization Image...

Load an alternate image for visualizing AORs.

4.3.2.4 Cube Menu

Build Cube

Build the cube from the enable records.

Reset Accounts

Reset the clipping accounts (which speed cube re-build and enable pixel back-tracking).

View Cube...

View the cube in a pre-existing cube viewer window, if available.

View Cube (new viewer)...

View the cube in a new viewer window.

Show Cube Build Feedback

If enabled, plot cube build feedback while the cube builds.

Build Cube with FLUXCON

Build the cube using flux calibration.

Build Cube with SLCF

Apply the slit loss correction function to the assembled cube, to correct for differential diffractive slit losses for extended sources.

Subtract Background

Subtract the set background from each record when building the cube.

Trim Wavelengths

Trim the unreliable ends of the orders, omitting those wavelength planes from the cube.

Use Reconstructed Positions

Use positions reconstructed from the spacecraft telemetry, rather than the commanded positions.

Build Uncertainty Cube

Build an associated uncertainty cube, if record-level uncertainties are available.

Set Cube Build Order...

Set the cube build order.

Aperture(s)...

Show the WAVSAMP apertures used for the various orders (see [Section 5.7 \[WAVSAMP\], page 56](#)).

4.3.2.5 Background Menu

Set Background from Rec(s)...

Set the record-level background from the selected records, prompting for an average of trimmed-average combination.

Load Background Rec(s)...

Load a saved list of background records to use from a '.bgl' file.

Background Blend

Blend data to create background.

Set and Scale Background A...

Set the first background in a blend from the selected records, specifying its fiducial point.

Set and Scale Background B...

Set the second background in a blend from the selected records, specifying its fiducial point.

Blend A and B Backgrounds...

Create a final background by blending backgrounds A and B according to weights calculated from a target fiducial value.

View Background A...

View the first blend background, and select the records associated with it.

View Background B...

View the second blend background, and select the records associated with it.

Save Background Rec(s)...

Save the list of backgrounds records used as a '.bgl' file.

View Background...

View the record background, if any, in a pre-existing record viewer window, if available.

View Background (new viewer)...

View the record background, if any, in a new record menu.

Remove Background

Remove the assembled background.

Rebuild Background

Recreate background from its associated record data.

Load Background Spectrum...

Load a 1D background spectrum from an extracted '.tbl' file.

Remove Background Spectrum

Remove any loaded 1D background spectrum.

4.3.2.6 BadPix Menu

Load Bad Pixels...

Load a saved list of bad pixels from a '.bpl' file, replacing any existing bad pixels already set.

Load and Append Bad Pixels...

Load a saved list of bad pixels from a '.bpl' file, appending to any existing bad pixels already set.

Save Bad Pixels...

Save the list of global and record-level bad pixels to a '.bpl' text file.

Clear Global Bad Pixels

Clear all global bad pixels.

Clear Record Bad Pixels

Clear all record-specific bad pixels.

Clear All Bad Pixels

Clear all global and record-level bad pixels.

Auto-Gen Global Bad Pixels...

Attempt to automatically generate global bad pixels from redundant information in well sampled cubes, prompting for detection parameters.

Auto-Gen Record Bad Pixels...

Attempt to automatically generate record-level bad pixels for all records, prompting for detection parameters.

4.3.2.7 Info Menu

Project Parameters...

Display the currently configured project parameters.

As-Built Parameters...

Display the project parameters at the time the most recent cube was assembled.

Calibration Set Details...

Show the details of the currently loaded calibration set.

Debug Cubism

Enable CUBISM debugging, so that errors will halt at the command line with full traceback information (see [Section 7.4 \[Debugging CUBISM\], page 73](#)).

4.3.2.8 Help Menu

About Cubism...

Show the current CUBISM version, and the version which was used to assemble the loaded cube project (if different).

Cubism Manual...

Load the PDF CUBISM manual.

4.3.3 Data Records

A CUBISM project holds all of the data records (often called simply *BCDs* — see below) necessary for building a given cube. The records either include the data directly, or hold a file reference to the data on disk, which is loaded on demand. In [Figure 4.3](#), two records have been selected. In addition to the spectrum data frame, each record holds (optionally) the associated uncertainty frame, and the ‘**BMASK**’ pixel mask frame.

4.3.3.1 Record Info

A variety of information is shown for each record, and the records can be sorted by individual columns clicking on the column’s header. The information recorded is:

ID	A (hopefully) unique ID formed from filename.
Exp	The BCD exposure time in seconds (from the headers).
Observed	The date and time the BCD observation (GMT).
Added	The date and time this data record was added to the project (local time zone).
Type	The type of the record, encoded as <code>tMMO_pos</code> , with <ul style="list-style-type: none"> • <code>t</code>: The type of data record: <code>d</code> for ‘DROOPRES’, <code>c</code> for ‘COADD’, <code>f</code> for ‘FLATAP’, or blank for the (by far most common) ‘BCD’ (see Section 4.3.3.3 [Record Data Types], page 21). • <code>MM</code>: The module: ‘SL’, ‘SH’, ‘LL’, or ‘LH’. • <code>O</code>: The targeted order: 1, 2, or blank, for high-res or full-slit low-res (e.g. ‘LLBoth’) targeting. • <code>pos</code>: The position within the slit which was targeted: <code>cen</code>: the slit center, <code>a</code>: nod position 1, <code>b</code>: nod position 2.
Step	The step sequence within the map as <code>I[X,Y]</code> , where <code>I</code> is the EXPID of this step, and <code>X</code> and <code>Y</code> are the row and column positions within the map.

A secondary page of information is available by clicking on the right angle toggle button at the extreme right edge of the header bar. This page includes:

RA	RA targeted by the slit field of view position (J2000).
DEC	DEC targeted by the slit field of view position (J2000).
DATA	Whether the data for this record are loaded, rather than just a link to the file. Data are loaded on demand.

UNC	Whether the associated uncertainty data for this record are loaded. These are discovered automatically alongside the primary data products and loaded.
BMSK	Whether the associated BMASK mask data for this record are loaded. The BMASKs are discovered automatically alongside the primary data products and loaded.
ACCT	Whether the “accounting information” for this record is cached, mapping BCD pixels to sky pixels.
BPL	How many record level bad pixels exist for this record (see Section 5.5 [Bad Pixels] , page 50).

The records can be sorted by any of the available data fields by clicking the button associated with each header word, e.g. to sort by RA, click RA.

4.3.3.2 Record Enabled State

All data records can either be *enabled* or *disabled*. Disabled records have lines drawn through them in the project display (see [Figure 4.3](#)). Disabled records can be viewed and interacted with normally, but are not included in the assembled cube. This can be useful to omit the rare frame with garbled data, or to include data in the project solely for the purpose of constructing a background. Note that failing to disable records which are not associated with the AOR(s) constituting the map can cause the cube created to be very large or even fail.

4.3.3.3 Record Data Types

The IRS pipeline produces a variety of different types of output spectral data with differing levels of processing applied. The standard product which CUBISM uses is the ‘BCD’, or basic calibrated data. In addition, however, CUBISM can operate on ‘DroopRes’ (not flat-fielded or straylight-corrected) and ‘FlatAp’ (not straylight-corrected) data files. See [the IRS Data Handbook](#) for more information on these different data products. Typically, they would be used in CUBISM to test results only if problems with the flat-field or straylight-correction (SL) were suspected.

Note that reference to the term ‘BCD’ throughout this manual is inclusive of the other, less commonly used files types (‘FlatAp’, ‘DroopRes’).

4.3.3.4 Operating on Records

Individual records or groups of records can be examined for header information, renamed, deleted, enabled/disabled, viewed as an individual or a stack, averaged into a background frame, and much more. Most of these options are accessible from the *Record* menu (see [Section 4.3.2.3 \[CUBISM Project Record Menu\]](#), page 15). A very common operation is to view a record, which is done by simply double-clicking it, or selecting one or more records and hitting the *View Stack* button.

4.3.4 Status Bar

Beneath the record list, a text-based status bar gives feedback on the state of completion of the current operation, the number of selected records, etc.

4.3.5 Button Bar

The button bar at the base of the CUBISM Project window provides convenient access to common functions, also available with menu options or mouse shortcuts.

Build Cube/QuickBuild

(Re)build the cube.

Enable Enable the selected record(s).

Disable Disable the selected record(s).

View Record/View Stack

View the selected record, or an average stack of the selected records, if more than one selected.

View Cube View the assembled cube, if available.

Import AOR

Import BCD records from full mapping AORs found at or beneath the selected directory.

Save Save the current project, prompting for a file if not yet set.

Close Close the current project, prompting to save any unsaved changes.

4.4 CubeView

CubeView is a custom viewer with both general purpose and IRS-specific tools and configuration options. It is used to view BCD records, a stack of records, special frames like the background frame, or full spectral cubes — as individual planes or maps created from the cube — and AOR visualization images. You can have as many instances of the viewer as you want, and each can be configured differently. By default, individual CUBISM projects attempt to target their own set of viewer windows, only creating a new window if none is presently available.

4.4.1 Title Bar

The title bar of CubeView gives an indication of what is currently being displayed. An example is ‘Record: ngc5194 <Average of 36 recs>’. This information will change depending on whether a record, background record, cube, map, or visualization image is shown. The same information is repeated below the line status bar in the image info block.

4.4.2 Menus

4.4.2.1 File Menu

Save as PNG...

Save the current image as a PNG file.

Save Map as FITS...

Save the current spectral map (if available) as a FITS file, with complete WCS header information.

Export to Command Line...

Export the current image as an array variable on the IDL command line.

Extract Region from File...

Extract a spectrum from the current cube from the extraction region encoded in the header of an existing extracted spectrum (‘.tbl’ file), or DS9 region (‘.reg’).

Save DS9 Region...

Save the current extraction region (whether specified directly, or loaded from another file) as a DS9 ‘.reg’ region file, which can be loaded in DS9 to visualize the extraction area.

Close Close the viewer window.

4.4.2.2 Options Menu

Colormaps

Change the display color map for the image and color bar.

Scale Image

Linear Scale image linearly between the clipping limits.

Square Root

Scale image as the square root between the clipping limits.

Logarithmic

Scale image logarithmically between the clipping limits.

Histogram Equalization

Scale image between the clipping limits to generate a flat final color histogram.

Trim 1%, 5%

Before scaling, first trim 1%, 5% from the distribution of pixel values being scaled (full image or histogram box).

Set Scale Range...

Explicitly set and lock the low and high scale clipping limits.

Freeze Scaling

Lock/Unlock the low and high scale clipping limits at their current value. This can also be done with SPACE.

Set Size**256,384,512,768**

Set the size of the display image to this width. These four sizes can also be selected with 1 – 4.

Wrap

Enlarge the display image size to show the entire image at the current zoom (up to the size of the monitor). This can be selected with 0.

4.4.2.3 Tools Menu

The tools menu provides an alternative means of selecting among the viewer tools in the tool palette at the top of the CubeView viewer window. See [Section 4.4.3 \[CubeView Tools\], page 24](#).

4.4.3 CubeView Tools

The viewer offers a number of individual tools for interacting with the data, some of which are generic and always available, and others of which are specific to certain data types. The tools are accessible via the palette, through keyboard shortcuts, and via the *Tools* menu. Other tools accessible below the displayed window will be discussed in the next section.

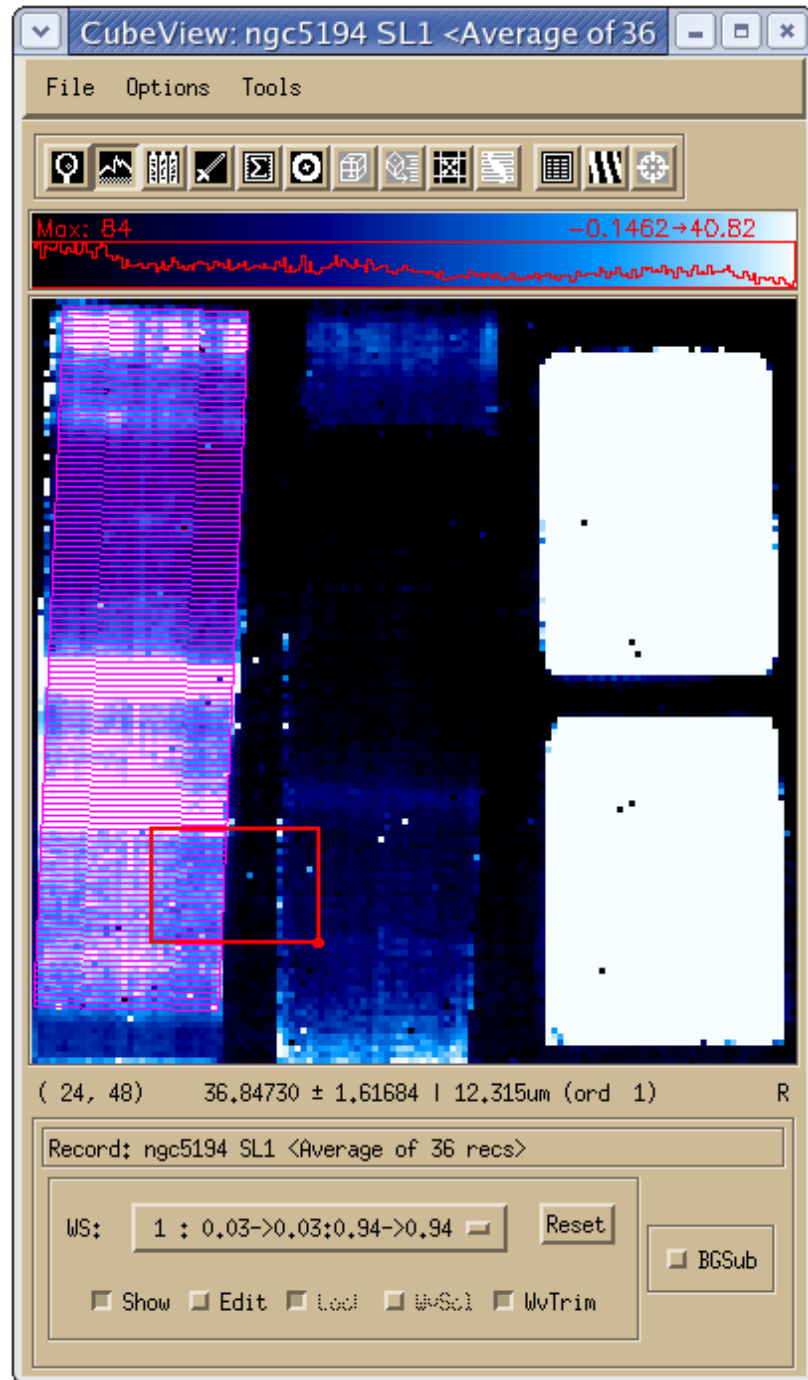


Figure 4.4: CubeView, showing a stack of 36 records, with an active histogram scaling box, and the order 'WAVSAMP' displayed.

An example CubeView showing a stack average of 36 records is shown [Figure 4.4](#).

4.4.3.1 Tool Interaction

There are two categories of main CubeView tools, *exclusive* tools, which require exclusive control of the mouse input, and *non-exclusive* tools, any number of which can be active at any time. These two types are separated in the tool palette, with exclusive tools grouped at left, and non-exclusive tools grouped at right (see [Figure 4.4](#)). Selecting an exclusive tool enters a mode of operation in which mouse inputs are interpreted by that tool alone.

Cubeview tools can be toggled on or off by clicking the button, selecting the *Tools* menu item, or using the key shortcuts. The currently active tool(s) (if any) are indicated by depressed buttons in the tool palette, and with tick marks in the *Tools* menu. If an exclusive tool is already active, toggling it on again (as opposed to simply selecting another tool) serves as a “reset”, which restores it to its initial state, for instance removing any box area which has been defined. Other tools which draw marks or annotations on top of the image keep those marks drawn even when they are not active. An example is the bad pixel tool, which continues to display bad pixel marks even when it is not the active tool (i.e. when mouse clicks don’t affect the bad pixel mask). To remove these displays, *reset* the tool as described.




Tool-tips identifying the tool, its key shortcut, and associated mouse operations (encoded as ‘[*left* | *middle* | *right*]’) can be displayed by hovering the mouse over the tool button in the tool palette.








4.4.3.2 Box Regions

Multiple tools make use of box regions, for instance to define an area for computing statistics or scaling the image (see, e.g., the red box in [Figure 4.4](#)). When another tool is activated, these tools leave behind “corners” of the box area to indicate their selection. Reactivating the tool restores this preexisting box. Click and drag from upper left to lower right to define a box area initially. Click and drag within the box to move it, or on the “handle” at lower right to resize it. The arrow keys also move the position of the box, by one pixel at a time. Reset the box as described in [Section 4.4.3.1 \[Tool Interaction\]](#), page 26.




4.4.3.3 Tool Overview

The individual exclusive tools with their button icons are:

Button	Tool	Purpose	Shortcut
	Zooming	Zoom in and out of images	<i>z</i>
	Histogram Scaling	Re-scale image to highlight local features	<i>h</i>
	Color Table	Adjust color table end points and gamma	<i>c</i>

	Image Slicing	Plot data slices through images, with position feedback	<i>l</i>
	Box Statistics	Calculate and report statistics in box	<i>s</i>
	Aperture Photometry	Perform simple circular aperture photometry on sources	<i>p</i>
	Cube Extraction	Extract spectra from rectangular regions from the cube	<i>x</i>
	Pixel Backtracking	Backtrack cube pixels to contributing BCD pixels	<i>t</i>
	Bad Pixel	Examine and edit the global and record level bad pixels	<i>p</i>
	Visualization	Overlay AOR slit positions and permit selecting records from the overlay	<i>v</i>

The non-exclusive tools are:

Button	Tool	Purpose
	Pixel Table	Display a grid of pixel values under the cursor
	Order Mask	Mask out all data outside the orders
	Compass Rose	Display a compass rose

Not all tools are active at all times — inactive tools are grayed out. Among the tools, Cube Extraction can only be used when viewing a cube, pixel backtracking can only be used when viewing a single cube plane, and bad pixels and order masking are enabled only when viewing BCD data. Visualization is only possible with a visualization image, and the Compass Rose is disabled except with cubes and visualization images (i.e. images with WCS coordinate information).

4.4.3.4 Zoom Tool

The zoom tool allow arbitrary zooming in on image regions, and panning within images. By default, images are displayed at the maximum integer zoom which fits the entire image

in the display window. To zoom in, drag a rectangular region around the area of interest, or simply click and release to double the zoom level and center on the clicked point. To zoom out, right-click. All zoom levels are saved onto a “zoom stack”, which is navigated backwards one step at a time when right-clicking. To zoom all the way out, right double-click.

When an image is zoomed in, middle-click dragging (or CONTROL-click dragging) pans the image smoothly. While panning in this way, holding SHIFT constrains the pan to be vertical or horizontal.

Middle-click (or CONTROL-click) and release re-centers the image on the point clicked, if possible. If the display canvas is resized (either using the *Set Size* menu item, see [Section 4.4.2.2 \[CubeView Options Menu\]](#), page 23, or by re-sizing the entire CubeView window), the image is re-zoomed.

The zoom tool’s key shortcut is *z*.

4.4.3.5 Histogram Tool

The histogram tool is actually an image rescaling tool. It allows you to identify a region of the image, and rescale the image values to emphasize it. Simply create a box area, or move and resize an existing box area, to the region of interest. The scaling mode can be linear, square root, logarithmic, or histogram equalizing, with 1% or 5% trimming available, in the *Options->Scale Image* menu. By default, the entire image is scaled. This tool also draws a histogram of the resulting image colors on the colorbar (see [Section 4.4.3.6 \[Color Tool\]](#), page 28), and gives the scaling range. The scale clipping limits can be frozen with SPACE, or specified directly using *Options->Set Scale Range*. It can be convenient to “reset” the histogram tool to remove it’s box area and define a new one, in particular if you are zoomed in on a different region of the image. To quickly reset and turn it back on, hit the key shortcut twice. The box region for the statistics tool is red, and can be seen in [Figure 4.4](#).

The histogram tool’s key shortcut is *h*.

4.4.3.6 Color Tool

The histogram tool provides a much more direct means to bring out detail in a given image area, but the color tool allows one to use palettes similar to those found in SAOImage/DS9. One can use the click and drag method for adjusting the color map directly. Simply left-click, hold, and drag around the window. Moving down narrows the color map, and up widens it, changing the contrast. Moving left shifts the upper and lower cutoffs to lower color values, and right shifts them to higher values. Right-clicking resets the color map. If you find yourself using this tool often to bring out detail in different regions of an image, try out the histogram tool instead and see if you prefer it.

The color tool’s key shortcut is *c*.

4.4.3.7 Image Slicing Tool

The image slicing tool allows slices to be taken through any image at arbitrary angles, either a single pixel wide, or averaged over a given width. Just left-click and drag to define a slice at any angle, or hold down CONTROL or right click and drag to constrain the slice to be horizontal, vertical, or diagonal. A plot window with the pixel values along the slice vector is displayed as the slice is made, and mousing over it simultaneously highlights the value

and the corresponding pixel position(s) within the image. SHIFT-left-click, or middle-click (and drag) to define an averaging width perpendicular to the slice line (indicated by dashed lines). Only full pixels are averaged, and the selection heuristic is simple.

While other tools are active, the slice plot window will remain active and continue to indicate the position. Creating a new slice replaces the existing line. To remove the slice line, reset the tool by turning it on twice, or destroy the plotting window. An example slice plot is shown in [Figure 4.5](#).

The image slicing tool's key shortcut is `l`.

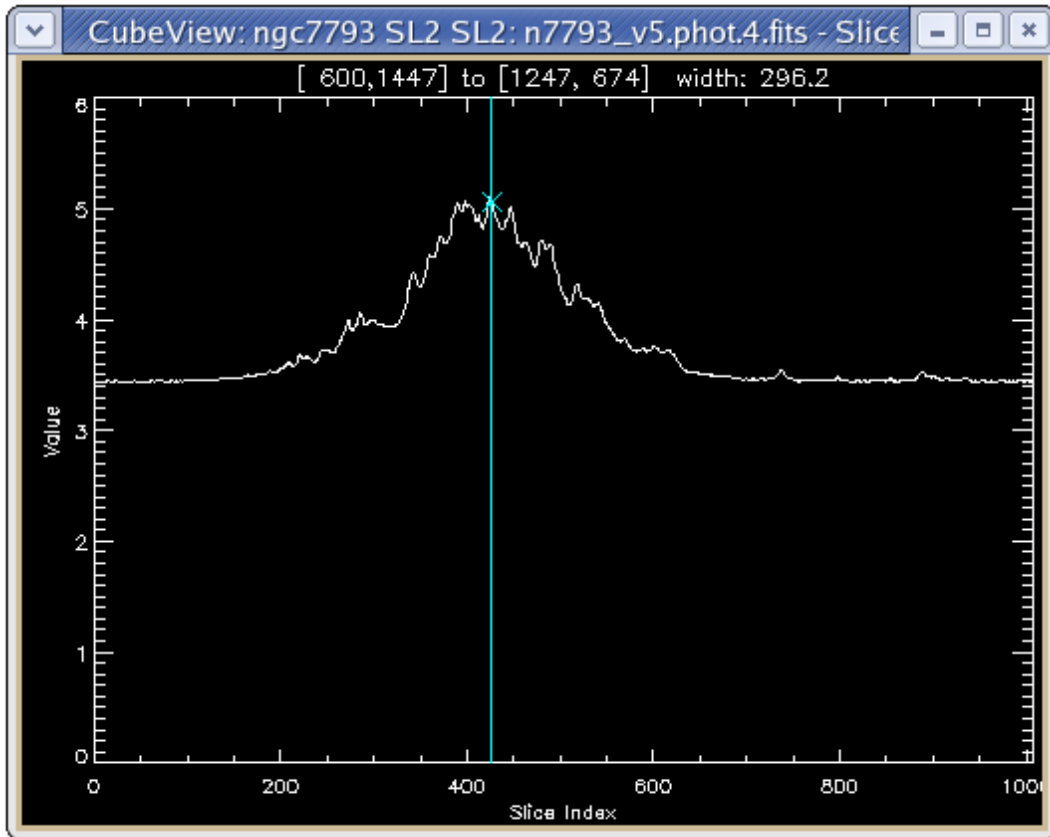


Figure 4.5: A plot of a slice through an image.

4.4.3.8 Box Statistics Tool

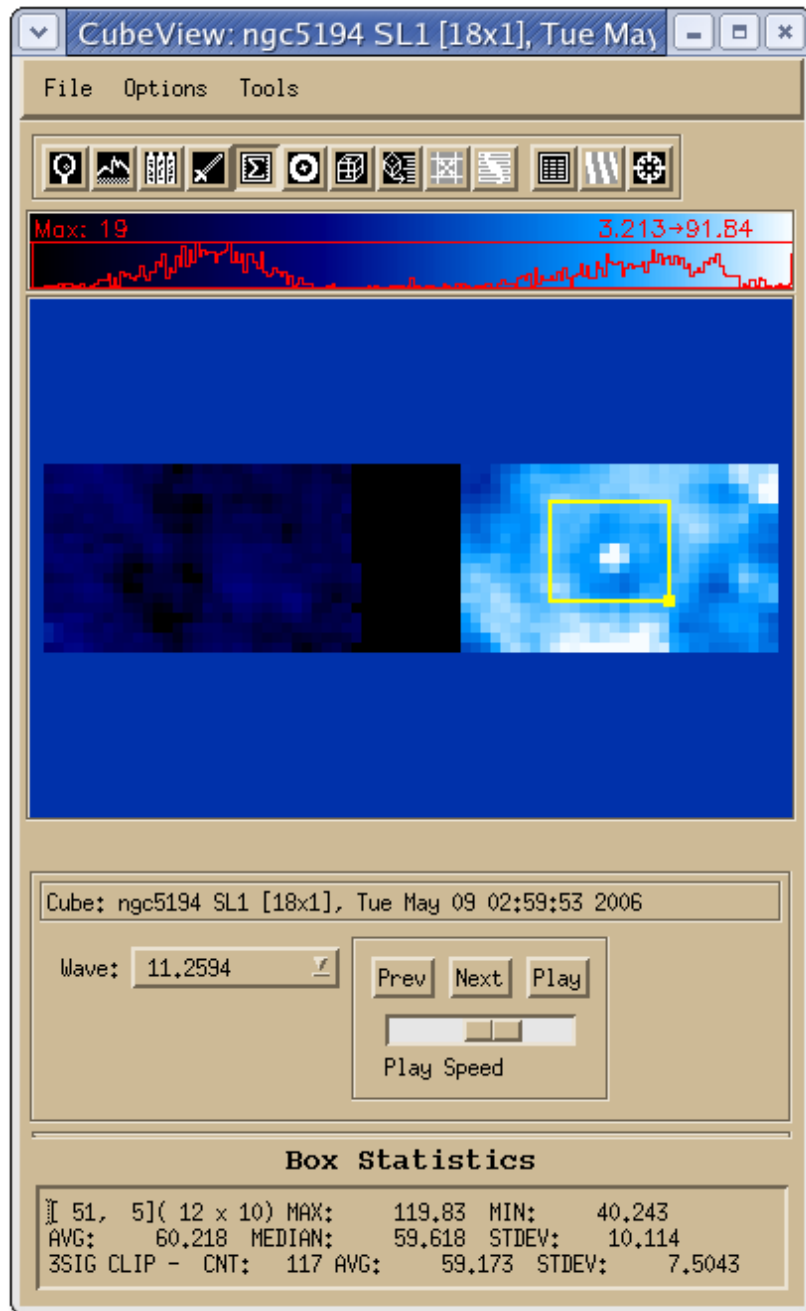


Figure 4.6: Example CubeView statistics computed in the yellow box.

The statistics tool extends the CubeView window, adding a panel of statistics information at the bottom. The first two lines list the box size and position, as well as the minimum, maximum, average, median, and standard deviations within the selected box. The 3-sigma trimmed pixel count, average, and standard deviation in the box are given in the third line.

Simply create or modify a box region (see [Section 4.4.3.2 \[Box Regions\]](#), page 26) to view the statistics within that box. The box region for the statistics tool is yellow. An example of the tool's output is shown in [Figure 4.6](#).

The box statistics tool's key shortcut is **s**.

4.4.3.9 Aperture Photometry Tool

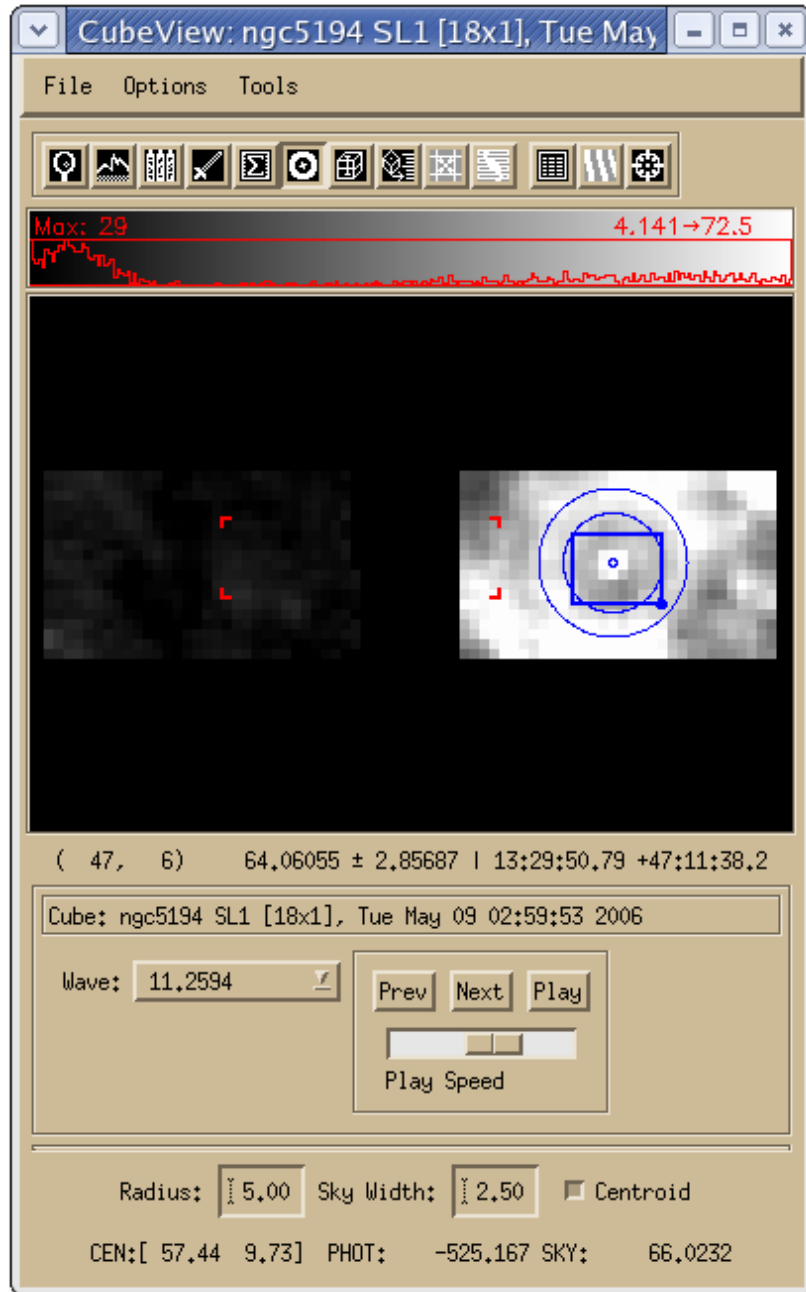


Figure 4.7: Example CubeView aperture photometry, centered on the centroid found in the box area.

The aperture photometry tool performs simple circular aperture photometry with a sky annulus. It extends the CubeView window to report the photometry results at the bottom. This panel includes text fields for entering the primary radius and sky annulus width for circular photometry. A box is drawn (blue, for this tool), and the centroid of the source

in the box is computed, and used to center two circular apertures, source and sky. If the *Centroid* option at bottom is de-selected, the center of the box is used as the center of the circular apertures instead. There are actually two centroids computed: a standard “center of mass” style centroid, and a refined DAOPhot-style “vanishing derivatives” centroid. If the latter is invalid, because it falls too far outside the box, the former is used, and the centroid is marked with an ‘x’ in addition to a circle. Enlarging the search box may help with accurate centroiding.

The total flux inside the inner circular aperture, minus the scaled sky flux (reported as the average within the sky annulus) is given. An example of the tool’s output is shown in [Figure 4.7](#).

The aperture photometry tool’s key shortcut is *p*.

4.4.3.10 Cube Extraction Tool

The cube extraction tool is used to extract spectra from rectangular regions within a cube, and can only be used when a cube is being displayed. See [Section 4.4.3.2 \[Box Regions\]](#), [page 26](#), for more on defining and manipulating a box region, which is magenta for this tool. Once an extraction region is defined, the spectrum is displayed in the CubeSpec tool. See [Section 4.5 \[CubeSpec\]](#), [page 40](#).

The cube extraction tool’s key shortcut is *x*.

4.4.3.11 Pixel Backtracking Tool

BCD	Pix	Frac	Val	Back	(Val-Back)	Flag
S0_9480192_0023_0000_	(31, 64)	0.275	59.1±6.85	-39.0±1.85	98.1±7.10	BP
	(32, 64)	0.117	87.0±12.4	2.60±2.55	84.4±12.7	
	(31, 65)	0.272	118.±5.71	3.87±2.11	114.±6.08	
	(32, 65)	0.114	90.5±7.60	6.46±2.58	84.0±8.02	
S0_9480192_0024_0000_	(31, 65)	0.136	93.0±9.36	3.87±2.11	89.1±9.60	
	(32, 65)	0.0623	100.±6.01	6.46±2.58	93.7±6.54	

Figure 4.8: The backtracking window of CubeView, showing contribution from 6 pixels among 2 BCD records to the cube pixel, one of which is flagged as a bad pixel.

The pixel backtracking tool is an advanced cube analysis and verification tool. It is enabled only when viewing a *single plane* of a spectral cube. For the pixel under the cursor, it displays a list of all the input BCD pixel fragments which contributed to that cube pixel, and simultaneously highlights the contributing records in the associated CUBISM Project window. The more pixel redundancy built into your map, the greater the number of records and record pixels which contributed to a given cube pixel (also, cubes from the high-resolution modules typically have more, since there is also overlap among the multiple high-res orders).

The information displayed includes the record ID, the pixel within that record, and the fractional contribution to the cube pixel being backtracked. It also lists the value of the BCD pixel (with uncertainty, if available), the value of the background, and the ‘Val-BG’

difference. At right is any ‘BMASK’ flag with the same notation as the status line (see [Section 4.4.5 \[CubeView Status Display Line\], page 38](#)), augmented by the flags ‘BP’ or ‘BP(1)’ for global and record level bad pixels (see [Section 5.5.1 \[Global and Record Level Bad Pixels\], page 51](#)).

Simply turn the tool on and mouse around. The BCDs which contributed to the given cube pixel are highlighted in the associated CUBISM Project window. Left-clicking freezes on an individual cube pixel, marking it with a green ‘x’, while right-clicking restores the free motion. To stop backtracking, close the backtrack window, or reset the tool by clicking its icon again after it’s enabled.

In the BackTracking window, right-clicking on any individual list item pops up a context menu which allows the associated BCD pixel to be set or cleared as a global or record-level bad pixel (see [Section 5.5.1 \[Global and Record Level Bad Pixels\], page 51](#)). In addition, you can simultaneously set a given BCD pixel as a record level bad pixel in *all* the listed records contributing that pixel using the *Bad Pixel (These records)* context menu option. By default, the list of contributing pixels is sorted by BCD identifier; clicking on the column head buttons toggles sorting by any of the columns, which can be useful for example to quickly identify major outliers in the distribution for long lists.

You can continue to navigate through the cube while backtracking is active.

The pixel backtracking tool’s key shortcut is *t*.

4.4.3.12 Bad Pixel Tool

The bad pixel tools marks individual bad pixels, either user-flagged, or pipeline-produced, and allows global and record-level user bad pixels to be added or removed (see [Section 5.5 \[Bad Pixels\], page 50](#), for more information on different types of user-level bad pixels, and tips on selecting them). It is only available when BCD images are being displayed. To use, simply left-click to set or remove a given bad pixel from the global bad pixel list, clicking and dragging to set or unset multiple pixels at once. Middle-click (or CONTROL-left-click) instead to set to record-level bad pixels for all of the records being displayed. For instance, if a stack of 10 records is being displayed, setting a single record-level bad pixel will add that bad pixel to all 10 records.

A number of marks are drawn in CubeView to indicate global bad pixels, record level bad pixels, and various conditions and flags from the pipeline generated mask files ‘BMASK’ and ‘PMASK’. The various symbols seen when the Bad Pixel Tool is enabled are:

Cyan ‘x’s	Global user-defined bad pixels, applying to all records.
Green ‘x’s	Record-level user bad pixels, applying to individual records.
Blue ‘+’s	Locations where the ‘PMASK’ (permanent mask) has any bit set. The ‘PMASK’ typically contains a small number of unruly pixels.
Red ‘diamonds’s	Non-fatal bits set in the ‘BMASK’.
Red ‘x’s	Fatal bits set in the ‘BMASK’. These are bits 12, 13 and 14, i.e. none or only one usable sample in the exposure ramp, or pixel fatally flagged in the ‘PMASK’.

See the [IRS Data Handbook](#) for a reference on the individual mask bits which the IRS pipeline applies. Note that despite being a fatal mask value, bit 8 in the ‘BMASK’ (not flat fielded) is not marked, since it occurs everywhere off the orders. All ‘BMASK’ values, whether marked or not, are indicated in the status display line (see [Section 4.4.5 \[CubeView Status Display Line\]](#), page 38).

When viewing and setting bad pixels in CubeView, the right mouse button can be used to cycle through 4 settings controlling which bad pixels are indicated:

1. All mask and user-set bad pixels marks.
2. All mask and user-set bad pixels marks, except for non-fatal ‘BMASK’ marks.
3. All fatal ‘BMASK’ mask and user-set bad pixel marks.
4. Only user-set bad pixel marks.

SHIFT-right-click cycles in the opposite direction.

The bad pixel tool’s key shortcut is **b**.

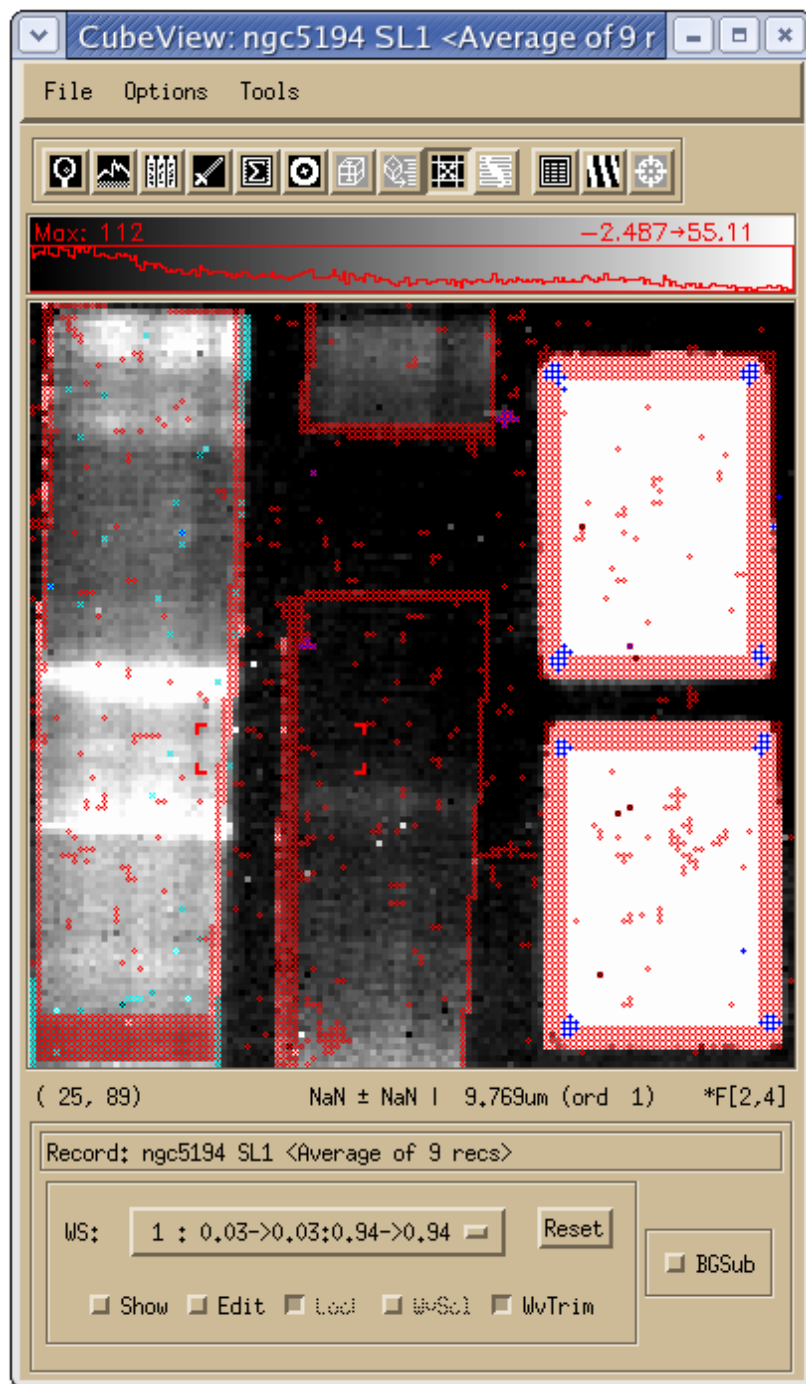


Figure 4.9: A record stack with bad pixels displayed.

4.4.3.13 AOR Visualization Tool

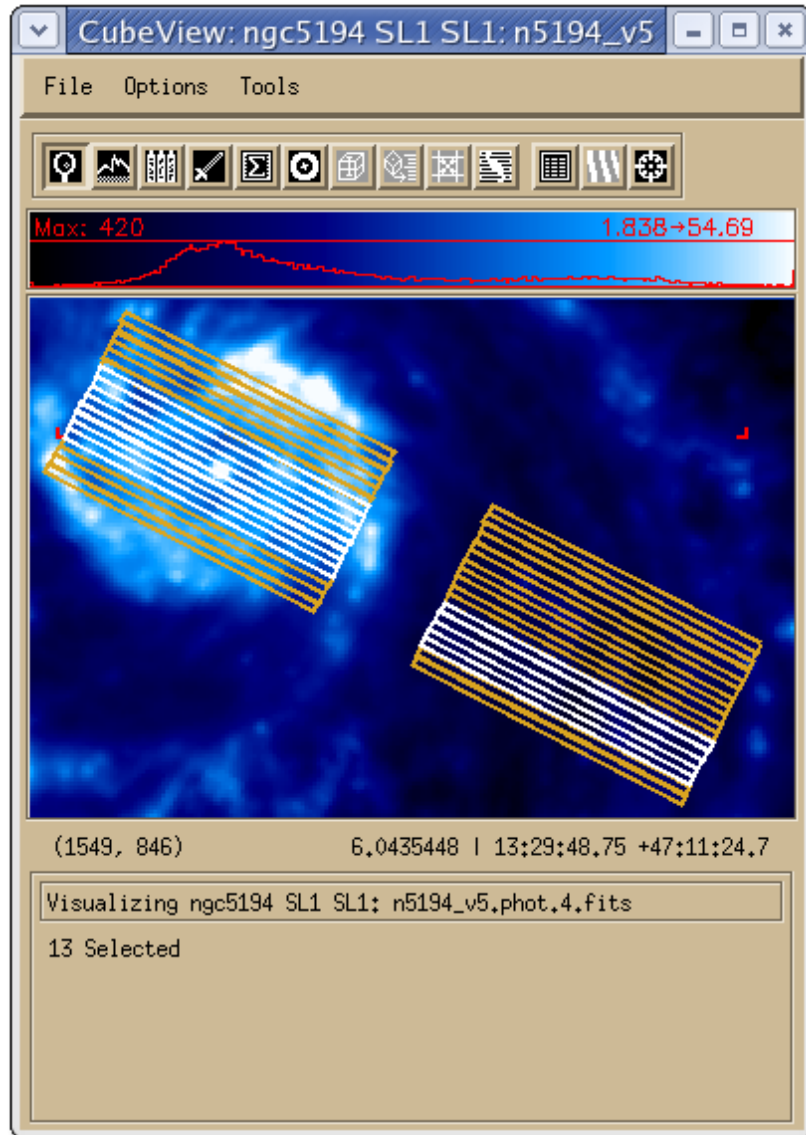


Figure 4.10: A visualization of the records on an IRAC 8 micron image of M51, with 13 records selected.

It can be useful to view a graphical representation of the AOR layout overlaid of an image of your target. The AOR Visualization tool can be activated from the CUBISM Project window by choosing *Record->Visualize AORs...* see [Section 4.3.2.3 \[CUBISM Project Record Menu\]](#), page 15. This will prompt for an image to use for visualization (if none is set). Select a FITS image with valid astrometry (DSS, 2MASS, Spitzer IRAC/MIPS, etc.), and the records will be shown, as in [Figure 4.10](#). Any selected records in the cube project will be shown in white. Disabled records are gray.

When this tool is active, records can be selected directly in the image itself. Click on a record to activate it, at which point it is drawn in white. Click and drag to select contiguous records. SHIFT-click to select the range of records from the last selected, and CONTROL-click to select non-contiguous ranges (similar semantics as described in [Section 4.2.1 \[List Selection\]](#), page 10). Click outside the record overlays to deselect all records. Direct selection on the image can be very useful from defining background records directly from an infrared image see [Section 5.4.7 \[Background Selection Using Visualization\]](#), page 50.

4.4.3.14 Pixel Table Tool

The pixel table tool can be activated at any time, and displays a group of pixel values around the cursor position.

4.4.3.15 Order Mask Tool

The order mask tool sets to zero all pixels outside the order, as defined by the ‘WAVSAMP’ aperture (see [Section 5.7 \[WAVSAMP\]](#), page 56). It is useful when checking for bad pixels, and is only available when BCD record data are displayed.

4.4.3.16 Compass Rose Tool

The compass rose draws NE compass lines on images data with available astrometry (cubes, maps, and visualization images).

4.4.4 Colorbar

The colorbar shows the currently selected colormap (as adjusted by *Options->Colormap* menu, and the Color tool — see [Section 4.4.3.6 \[Color Tool\]](#), page 28), along with a plot of the current color pixel histogram of the displayed image region, and the scale clipping limits. See [Section 4.4.3.5 \[Histogram Tool\]](#), page 28.

4.4.5 Status Display Line

Below the display window, a status display line reports the pixel position and value (with uncertainty), and depending on the data being viewed:

1. **BCD images** (see, e.g., [Figure 4.9](#)) Wavelength and order of the cursor position, along with a flag code identifying the bad pixel type.
2. **Cubes and Visualization FITS Images**, (see, e.g. [Figure 4.7](#)) Coordinates of the current position from the WCS information.

4.4.6 Bad Pixel Codes

The flag code at the far right of the status display line indicates the common mask values from the ‘BMASK’ (OR’d together when viewing averages of multiple records):

- * The pixel was flagged in the ‘PMASK’.
- s The pixel was saturated during the read ramp, but the value was recovered.
- S The pixel was saturated and uncorrectable.
- F No flat field was applied to the pixel.
- R The pixel suffered a radhit, but it was corrected.

- 0 No samples in the read ramp were valid.
- 1 Only one sample in the read ramp was valid.

All relevant flags are listed together, and for bit values from the ‘BMASK’ not listed here, one or more numeric bit value is listed in brackets, like ‘[2,4]’. See the [IRS Data Handbook](#) for a reference of the ‘BMASK’ mask bits.

4.4.7 Image Info Block

Information on the image being displayed is provided here, the same as is shown in the title bar (see [Section 4.4.1 \[CubeView Title Bar\]](#), page 23).

4.4.8 WAVSAMP Pane

Below the image info block, when a BCD record is being displayed, is the *WAVSAMP Pane*, where the ‘WAVSAMP’ (see [Section 5.7 \[WAVSAMP\]](#), page 56) can be manipulated. *Show* overlays the current ‘WAVSAMP’ for the current build order, *Edit* adds handles which can be used to adjust either side or end of the ‘WAVSAMP’. The normalized aperture coordinates of the 4 corners of the ‘WAVSAMP’ are shown for each order. The *Lock* toggle controls whether all orders are set to the same ‘WAVSAMP’ aperture. The *WvSc1* toggle creates a ‘WAVSAMP’ aperture which grows linearly with wavelength. The *Reset* button reset the full aperture.

Editing the ‘WAVSAMP’ involves clicking on one of the 6 adjustment handles and dragging. When in *WvSc1* mode, only the central controls adjust the width of the aperture, and the central wavelength. Otherwise, all controls change the width, either at one end, or both ends together. SHIFT-click and drag moves both sides of the aperture at once.

See [Section 5.7 \[WAVSAMP\]](#), page 56, for more information about when and how to edit the ‘WAVSAMP’. See [Figure 4.4](#) for an example displayed ‘WAVSAMP’.

4.5 CubeSpec

CubeSpec is the tool which permits viewing and manipulating extracted spectra, and create maps from spectral cubes. The CubeSpec tool is a custom viewer, used to view one dimensional spectra extracted from a cube (see [Section 4.4.3.10 \[Cube Extraction Tool\]](#), [page 33](#)). It is entered by extracting a spectrum from a cube, and is always associated with a CubeView window displaying a cube.

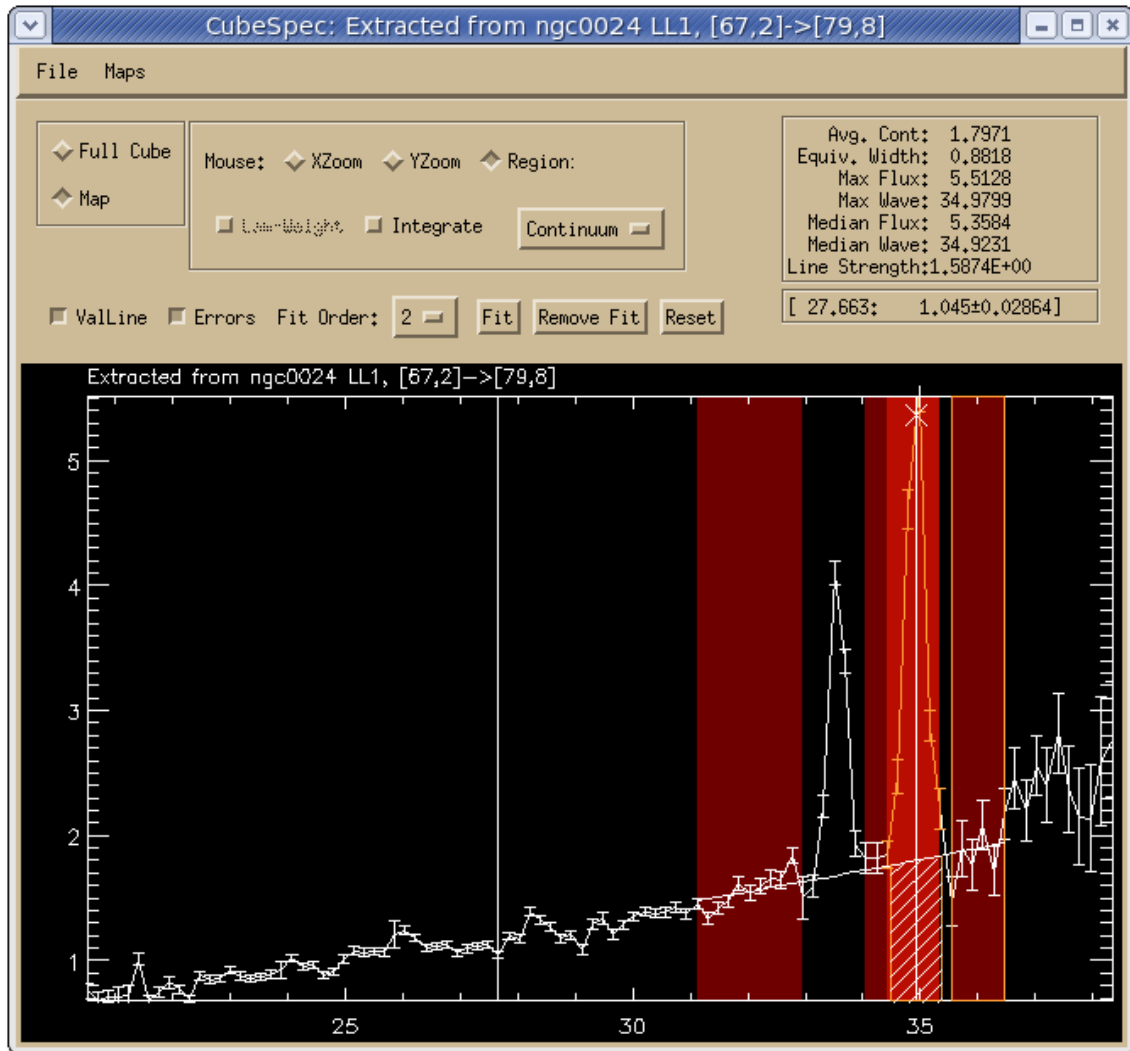


Figure 4.11: CubeSpec window showing an extracted spectrum and 4 component map.

4.5.1 Title Bar

The title bar of CubeSpec gives an indication of what is currently being displayed. An example is 'CubeSpec: Extracted from ngc2403_extranuc00, [2,1]->[10,5]'. This information will change depending on the cube extracted and region extracted from. The same information is repeated as the title in the plot.

4.5.2 Menus

4.5.2.1 File Menu

Save Spectrum as...

Save the current spectra as a '.tbl' file (ASCII text, IPAC table format).

Export Spectrum to Command Line...

Export the current spectra as an array variable on the IDL command line.

Close Close the CubeSpec Window

4.5.2.2 Maps Menu

Save Current Map...

Save the current map into a '.map' file for later use on other spectra.

Load Maps...

Load a spectral map that has been previously saved.

Reset to Default Maps

Re-load the default map sets saved in 'cubism/map_sets', removing others.

Set Redshift...

Set the Redshift of the object spectrum in 'km/s'.

Clear Redshift

Clears any entered redshift.

IRAC 5.8 um

Overplots a shaded red area of the IRAC 5.8 micron system response curve.

IRAC 8.0 um

Overplots a shaded red area of the IRAC 8.0 micron system response curve.

IRS Blue Peak-Up 16um

Overplots a shaded red area of the IRS Blue Peak-Up 16 micron system response curve.

IRS Red Peak-Up 22um

Overplots a shaded red area of the IRS Red Peak-Up 22 micron system response curve.

ISOCAM LW2 7um

Overplots a shaded red area of the ISOCAM LW2 7 micron system response curve.

ISOCAM LW3 15um

Overplots a shaded red area of the ISOCAM LW3 15 micron system response curve.

MIPS 24 um

Overplots a shaded red area of the MIPS 24 micron system response curve.

4.5.3 Buttons

Most of CubeSpec's functionality is accessible in the form of buttons.

Full Cube Default setting. In this mode, CubeView sends views of a single wavelength plane of the cube at a time to its associated CubeView viewer. Key shortcut *u*.

Map In this mode, pre-defined or custom defined maps (see [Section 6.2 \[Creating 2D Maps\]](#), page 62) are sent to the viewer. Key shortcuts *p* and *c* enable this mode (for peak or continuum definition, respectively).

Valline Enable the display of the vertical line in the spectrum plot.

Errors Pressing this button turns the uncertainty bars on and off in the plot.

Fit Order: 1, 2, 3, 4, or 5

Set the polynomial order for the continuum fit (see *Fit* button).

Mouse: XZoom

In this mouse mode, when you click the left mouse button in two locations, the plot will zoom in on that section of the wavelength range. Double-clicking the left mouse button in one location, or right-clicking, will return to the full wavelength range. Key shortcut *x*.

Mouse: YZoom

In this mouse mode, when you click the left mouse button in two locations, the plot will zoom in on that section of the flux range. Double-clicking the left mouse button in one location, or right-clicking, will return to the full wavelength range. Key shortcut *y*.

Mouse: Region/Lambda:

When *Full Cube* is depressed, this button is *Lambda*, and enables sending a single wavelength plane of the cube at a time to the viewer on click. When in *Map* mode the button is *Region*, and enabling it means that clicking a start and finish wavelength range in the spectrum will define a peak or continuum region.

Continuum/Peak

This button is used in conjunction with the Region button stated above. With Continuum chosen, you will select points to use for the continuum. With Peak chosen, you will select points to use for the emission lines.

Lam-Weight

If set, the continuum subtracted will be based on an average weighted by the distance away in wavelength from the peak to continuum point. Normally, all continuum planes are averaged with equal weight. Only available when both peak and continuum regions are defined.

Integrate

If set, an integral over the background-subtracted foreground region of ' f_{ν} d_{ν} ' is performed. In the case of weight maps (e.g. MIPS 24um), the ' d_{ν} '-weighted integral ' $\int(f_{\nu} w d_{\nu}) / \int(w d_{\nu})$ ' is performed, where ' w ' is the weight vector.

Reset Plot

Pressing this button removes all zooms and region selections.

Remove Fit

Pressing this button removes any fit generated.

Fit

Perform a fit over the peak and continuum regions, which must be defined first. The polynomial order of the fit is determined by *Fit Order*. Resulting fit parameters will be displayed at right.

4.5.4 Display Panes

Two display panes provide information on any line fit and the cursor position:

Right display box

This box displays several values from any fits performed: average continuum, equivalent width of the line, maximum flux, maximum wavelength, median flux, median wavelength, and line strength. These values will disappear when you press the Remove Fit button.

Lower Right display box

This box gives the wavelength and flux with the uncertainty of the current cursor position.

4.6 CubeSpec Plot Window

The plot window of CubeSpec is where maps are defined, the view is modified, etc. What operations are performed by the mouse is determined by which ‘**Mouse:**’ mode is selected in the button panes above:

XZoom/YZoom

Click twice to zoom in on the wavelength or flux scale.

Region/Lambda

Full Cube Clicking on a wavelength plane sends that plane to the viewer. The plane can be moved using the arrow keys. Holding SHIFT with the arrow keys moves 5 wavelength steps; SHIFT and CONTROL together moves by 10 steps.

Map A *Peak* or *Continuum* region is formed by clicking twice: once at the start and once at the end of the desired region. Middle-clicking selects regions, SPACE to switch among the selected regions. When a region is selected, left and right arrows move it, up and down increase or decrease its width.

When defining a region (for a map, or X/YZoom) the region can be canceled after the first click by right-clicking. During region definition, a line is left at the location of the first click.

5 Cube Assembly

This chapter describes in detail the steps required to assemble a cube from a set of mapping ‘BCDs’. A quick checklist of these steps is:

1. See [Chapter 3 \[Quick Start Guide\], page 6](#), for more on starting and running CUBISM.
2. Create a new cube project, and enter an appropriate name for the project (this can be changed later, and is not the same as the project’s ‘.cpj’ file).
3. Accept the default (latest) calibration set, or load a specific set.
4. Load all data records from the AOR(s) which will be built into a single cube.
5. Load any additional data records to be used for background subtraction. Disable them if they are not intended to appear in the cube (see [Section 5.1 \[Input Files\], page 44](#)).
6. Select and set the background records (see [Section 5.4 \[Backgrounds\], page 46](#)).
7. Double-check the cube build settings under the *Cube* menu, in particular use *Set Cube Build Order...* to select the order for building the cube (note that high-resolution cubes are typically built using *All* orders at once).
8. Build the cube using *Cube->Build Cube*.
9. Save the cube project to file.
10. Manually or automatically generate bad pixels, and re-build the cube. Iterate as necessary.
11. Enjoy your newly assembled cube: extract spectra, build maps, etc.

5.1 Input Files

CUBISM works from a set of input ‘BCD’ or other record level data to produce spectral cubes. It reads positional information from the headers, loads pipeline mask files and uncertainty images, and uses all of this information together to map record-level pixels to the sky plane. Details of the algorithm employed by CUBISM can be found in Smith et al., 2007 (PASP, submitted).

CUBISM works best when applied directly to raw AOR directories downloaded from the Spitzer archives. To import all data (spectra/masks/uncertainties) from all records in a given mapping AOR (or set of related AORs), simply hit the *Import AOR* button, select a directory which contains one or more AORs, and CUBISM searches all ‘BCD’ files contained in that directory or its subdirectories, assembling them into constituent AOR sets. Select one or more of these AORs to load the associated data sets into the cube project. An example can be seen in [Figure 5.1](#).

See [Section 4.3.3.3 \[Record Data Types\], page 21](#), for more information on non-‘BCD’ data types which CUBISM can use (typically only for debugging problems with the pipeline processing). Note that CUBISM projects can contain data from only one IRS module at a time, and that CUBISM does not check to see that the data sets you have selected are physically grouped, covering a moderate sized region on the sky.

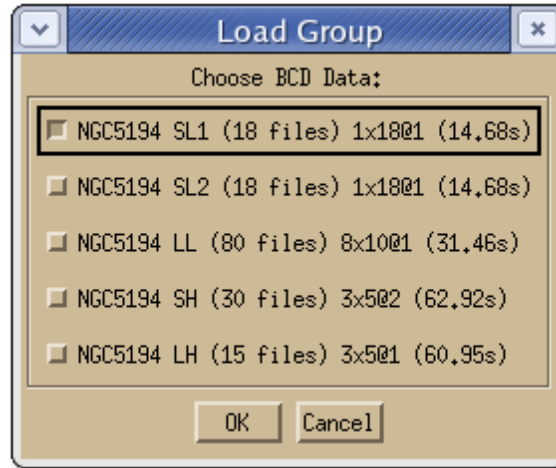


Figure 5.1: Importing AOR data, grouped by AORs found in the selected directory.

Loading sets of non-mapping data, for the purpose of background subtraction (see [Section 5.4 \[Backgrounds\], page 46](#)), can be accomplished conveniently using the `Record->Import Data by Module...` Simply point at a directory containing the data of interest, and select the appropriate data sets, grouped by object and IRS module.

5.2 Build Order

For low-resolution IRS spectral maps, there is a subtle distinction between the *build order* of a given cube produced by CUBISM, and the *target order* of the observations from which it was built, which is worth discussing in detail.

Each low-resolution IRS slit consists of two sub-slits, separated by a blocked region. When planning low resolution IRS observations, you must select where your chosen target position will be placed in the full slit. You can choose to have the spacecraft place your chosen target coordinates in the center of the full slit (the blocked location), in which case your record type will be, e.g. ‘LL_Cen’. Alternatively, you might place your target coordinates in the center of one of the sub-slits (e.g. ‘SL1_cen’). Staring mode observations further subdivide each sub-slit, placing the target at roughly 1/3 and 2/3 along the sub-slit; these positions are denoted ‘a’ and ‘b’, e.g. ‘SL1_a’.

None of this, however, has any impact on the order information present *in the data themselves*. If you have placed your target in ‘SL1_cen’, the other sub-slit will still image the sky onto the detector and record a spectrum. Each record’s data will thus contain both orders (as well as the “bonus” order).

It is often useful to build a cube from data in the “other” sub-slit’s order, i.e. the one not targeted during the map. When an object is mapped consecutively with both sub-slits, using all the data records to build a cube in a given order results in two potentially disjointed regions. We often call the region of cube built from the sub-slit *not* being targeted the “outrigger”. Data in the outrigger region can be very useful for improving the statistics for automatic bad pixel selection (see [Section 5.5.4 \[Automatic Bad Pixels\], page 53](#)), and for confirming the data levels in low signal regions. Alternatively, for large source, the outrigger field can contain interesting data in its own right.

By default, CUBISM sets the cube build order to the most common target order among the loaded BCD records. Obviously, if the full slit center was targeted, no individual order is preferred, so an arbitrary default is used. In general, you should check the cube build order to ensure you are building the cube you want. In the project window, *Info->Project Parameters...* provides this information, and you can set the cube build order with *Cube->Set Cube Build Order*. Recall that each low-resolution cube is built using data from a single IRS order (e.g. ‘SL1’). High resolution order wavelengths are much more continuous, and are combined directly in the cube.

5.3 Calibration Sets

CUBISM needs a large number of different pieces of calibration data to perform its tasks, including flux calibration functions, order positioning and wavelength solutions, extended source flux correction functions, etc. These calibration data are tied directly to those produced by the SSC for point sources, although they are extended and supplemented by additional information. CUBISM uses the concepts of *calibration sets*, unit bundles of calibration products which snapshot the latest calibration data, appropriate for the most recent pipeline processing.

By default, CUBISM loads the most recent calibration set which is bundled with it. Typically, you will not need to select a calibration set. As long as you are using the most recent IRS pipeline processing version of your data, the latest set is appropriate. The calibration set used with a given cube is saved in its project file, so that future use, even after new calibration sets are created, will continue to use that data. You can see which calibration set you are using, and inspect all of the parameters and input calibration files it consists of, with *Info->Calibration Set Details...* New calibration sets (‘.cal’ files) can be loaded with *File->Load New Calibration Set...* All input calibration files (in incrementing versions) can be found in the ‘cubism/calib/data/ssc’ directory in your CUBISM distribution.

Note that the general presumption is that more current calibration data is always better when used with data processed with the most up to date SSC IRS pipeline. This is true in general, although there is currently one time dependent calibration (the input bias voltage for the LH detectors was changed). CUBISM automatically uses the appropriate flux calibration values depending on the date of observation.

Though normally not necessary, new calibration sets can be created from the assembled inputs using the included helper routine ‘irs_make_calib_set’; see ‘cubism/calib/irs_make_calib_set.pro’ for more information.

5.4 Backgrounds

All IRS observations are affected at some level by the foreground or background emission of zodiacal dust and Galactic cirrus. For faint sources, subtracting off a local “sky background” in the form of a 2D ‘BCD’ frame constructed from data obtained nearby in position and time is the most effective method for removing the astrophysical background. In addition, ‘BCD’-level background subtraction greatly reduces the number of bad or “rogue” pixels contaminating IRS data, especially in the longer wavelength models, and effectively reduces residual systematic uncertainties which dominate at low flux intensity (often in the form of correlated patterns, bands, etc.). BCD level background subtraction is a recommended

step for all cubes, ideally performed with associated background data obtained during the same AOR, using the same integration time, etc.

Here we explore a variety of options for background subtraction, in decreasing order of preference. The degree to which backgrounds must be suppressed also depends on the source flux. Very faint sources at or well below the sky brightness level are more affected by residual background and low-level systematics (which often vary in magnitude and shape among orders and modules).

IMPROVING CUBES WITH BACKGROUND SUBTRACTION:

Subtracting a 2D background frame can greatly improve the quality of the assembled cube, especially for faint sources.

5.4.1 In Situ Background

The primary and recommended technique for assembling a background frame is to use dedicated background observations you scheduled along with your mapping AOR, or to use data obtained as part of your mapping observations which went far enough off the source to remove all source flux from the slit.

Finding records in which the *full slit* is uncontaminated by source emission, either within the spectral map itself, or as part of a dedicated background AOR grouped with the main observations, is the goal. This is often aided in maps using the low-resolution modules by the fact that each low-res slit (LL and SL) is comprised of two sub-slits which define the different spectral orders. For small objects, less than the size of one sub-slit, it is common to map first with one sub-slit, and then with the other (e.g. a SL1 map and, separately, a SL2 map). In this case, the “outrigger” data from those records collected when the source was placed in the other order can be used as a background set. Even if the full slit was used to map a source, the fact that the two orders extend away from the map center to either side often allow useful background frames to be found at the extremities of the map. For a very helpful method for making such selections, See [Section 5.4.7 \[Background Selection Using Visualization\]](#), page 50.

For dedicated offset sky observations, simply load the records (for example, with `Record->Import Data by Module->BCD`), disable them, and `Background->Set Background from Recs...` For backgrounds which appear among the mapping data set (e.g. as was done in the [Chapter 3 \[Quick Start Guide\]](#), page 6), simply select these records and choose `Background->Set Background from Recs...` Using either the average or min/max trimmed average is fine. These records can be built into the cube directly (though by definition they should not contain much source flux), or disabled to save time and space.

5.4.2 Archive Background

If you didn’t obtain dedicated sky observations, and none of the records in your map are free from source emission, you can still recover 2D level BCD maps by “borrowing” suitable data from the Spitzer archives. Useful sky data would have been obtained with the same instrument module, using (ideally) the same exposure time, targeted within roughly 10 degrees in ecliptic latitude from your source, and within 2–3 days of the target observation.

The efficacy of archive sky subtraction can vary widely, and depends on the level of the background at your source, and the historical behavior of the IRS detectors at the time of your observations. The closer in time and ecliptic coordinate the data, the better the subtraction will be (both in terms of removing the astrophysical foreground, and mitigating the effects of rogue pixels). The complete [log of all Spitzer observations](#) can be useful for identifying potential data sets which could contain useful background data. Remember that you must decide yourself whether a given BCD record constitutes a valid background frame, i.e. is free from source emission. For low resolution data, often the safest bet is the “outrigger” order from a staring mode observation (which often target unresolved sources) — e.g. SL2 from a SL1 targeted observation. The disadvantage of this technique is that the data must be available in the archive, which for many program requires waiting for the proprietary period to expire.

5.4.3 Background Blend

A related technique to the archive background is the “Background Blend”, useful if a set of observations from the archive which closely match the background properties of your source cannot be identified. In this method, a pair of observations from the archive (or elsewhere) which bracket the target background are linearly blended together to approximate a local 2D background. The same constraints on data of exposure apply: ideally all background observations would have been obtained within 2–3 days of the target observations, to maximize the effectiveness of mitigating rogue pixels (the behavior of which drifts on the time-scale of days).

To use this method, load the two data background sets, ensuring their records are disabled to avoid having them built into the cube. Select the subset of the first which you want to include in the background, and select *Background->Background Blend->Set and Scale Background A...* The selected background is displayed, with ‘BackgroundA’ indicated in the image info block (see [Section 4.4.7 \[CubeView Image Info Block\]](#), page 39). Enter a fiducial background value when prompted. This value serves as the interpolant for determining the weights of the two backgrounds in the blend. For SL data, it is convenient to use the flux measured in the Peak-Up Blue image at 16 microns (the lower of the 2 peak-up fields visible in all SL BCD images), if it is clear of source contamination. The box statistics tool can be useful (e.g. use as a fiducial value the 3-sigma clipped average of a large section of the PU field). Do the same for ‘BackgroundB’, and then select *Background->Background Blend->Blend A and B Backgrounds...*, entering a fiducial of your target data set, formed in the same way (e.g. in from the PU field statistics). Another potentially useful fiducial is the predicted background at the source position and date, as available in Spot.

5.4.4 1D Sky Spectrum

If no 2D BCD-level data are available for subtraction, some of the same benefits can be obtained by specifying a 1D spectrum to subtract from all the cube planes. You can build a cube, extract a spectrum from a *carefully chosen* area believed to be free from source contamination (use the visualization if possible, see [Section 5.4.7 \[Background Selection Using Visualization\]](#), page 50), and use this extracted spectrum as a 1D level background.

FLUX UNITS WARNING:

Note that any fluxed 1D spectrum must be extracted from cubes built with the same flux options (e.g. the SLCF correction) as the cube to which it is being applied. Spectra extracted from *raw* cubes (in ‘e/s’ units) do not have this restriction.

This method does *not* help mitigate rogue pixels, and is essentially equivalent to extracting spectra and differencing them after the fact, but operates on the entire cube at once.

5.4.5 Combined 1D and 2D Background

Occasionally, a 2D sky spectrum obtained from the archive in a different region of the sky, or from ‘BCD’ data taken with different exposure times, will do a reasonable job mitigating rogue pixels, but leave a residual background (positive or negative), which is obvious in parts of the cube without signal. This background offset can have a large effect on sources at or below the absolute residual background flux level. The “dark” areas in a cube where the residual is obvious should of course be smaller than the slit length, or else an *in situ* background could have been used (see [Section 5.4.1 \[In Situ Background\]](#), page 47).

Assuming you have located a region within the cube which you feel is free of source emission, the residual background can be removed by combining the original 2D-level background, archive or otherwise, together with a further 1D correction (applied as a separate scalar subtraction to all wavelength planes in the cube).

Assemble the 2D background-subtracted cube, extract the region, and save the resulting spectrum to file. At this point, you might like to load in the spectrum at the command line, fit a low order polynomial to it, and write it out again, to avoid adding noise into the cube. Something like:

```
IDL> p=read_ipac_table('file.tbl',h,UNITS=u)
IDL> fit=poly_fit(p.wavelength,p.flux,2,YFIT=yf)
IDL> p.flux=yf
IDL> write_ipac_table,'file.tbl',DATA=p,HEADER=h,UNITS=u
```

will accomplish this. Then load this 1D background, using *Background->Load Background Spectrum...* You will receive a warning that both 1D and 2D (record-level) backgrounds are being applied. Re-build the cube, and extract the same region: the spectrum should be near zero (with noise).

EXTENDED SOURCE WARNING:

CUBISM does not verify that fluxed 1D background spectra have the same extended source corrections applied as the target cube. Ensure this yourself.

5.4.6 No Background

Using no background is a reasonable last resort option for brighter sources, though the number of bad pixels in the raw unsubtracted data will be much higher, so more work will be required to identify them (see [Section 5.5 \[Bad Pixels\]](#), page 50). The (astrophysical)

foreground flux will also of course remain in the final assembled cube, which may affect analysis of extracted spectra or maps, and the degree to which matched extractions from cubes built in different IRS modules or orders will stitch with each other.

5.4.7 Background Selection Using Visualization

It can be convenient to select suitable records for inclusion in the background frame by using the AOR visualization of an IRAC or MIPS (or other zodiacal and/or cirrus sensitive waveband) image of the target region. See [Section 4.4.3.13 \[AOR Visualization Tool\], page 37](#), for more information on the visualization tool. Here we describe how to use the “outrigger” method to select suitable records from a set of low-resolution observations in the SL module (LL would be similar).

1. Load all the relevant SL (SL1+SL2) frames, which may span more than one AOR. Using *Import AOR* will permit selecting more than one AOR at once. Don’t mix data taken more than a few days apart, to minimize the effects of day-to-day changes in the pixel response of the detectors.
2. Visualize the AORs using, e.g., an 8 micron IRAC image (see [Section 4.4.3.13 \[AOR Visualization Tool\], page 37](#)). After using the histogram box to scale the image such that low-level diffuse emission is apparent, switch to the visualizer tool. To define background BCDs for, e.g., SL1, set the cube build order to 1 (*Cube->Set Cube Build Order*), and see which records are free of source flux. Highlight these outriggers with the mouse by click and drag.
3. For reference, save the resulting record set into a ‘.bgl’ file using *Background->Save Background Recs...*

5.4.8 Saving the Background List

The background list can be saved to a text ‘.bpl’ file using *Background->Save Background Rec(s)*. While not strictly necessary, this is a useful record of the background records for future reference in other cubes. The format is list of unique record IDs (‘DCEID’) which do not change with updated pipeline processing.

5.5 Bad Pixels

Hot, “rogue” or bad pixels occur in IRS data with a broad range of intensities, and they vary in position and frequency on the array on scales of hours to months. Over time, damage to the IRS arrays from solar particles has increased the number of these pixels, in particular on the two long wavelength arrays. Such pixels dominates the noise properties of spectral cubes, maps and extracted spectra, and if left untreated, severely degrade the quality of assembled cubes. Cleaning a cube of bad pixels is typically an iterative task: mark pixels, re-build the cube, inspect the cube for anomalous spectral or spatial features, mark additional pixels, etc.

CUBISM contains a variety of tools for identifying and flagging such pixels, which are then ignored when assembling the cube. Bad pixels can be identified either “manually” or automatically (or a combination of both). As discussed in [Section 5.4 \[Backgrounds\], page 46](#), subtracting a background record built from data obtained nearby in time, with the same record exposure time, can mitigate many rogue pixels (which can be thought of as pixels which respond normally to light, but have a bright and variable underlying

dark current component). However, adjacent-in-time background subtraction is not always available, and even when it is, a number of problematic pixels often remain. CUBISM has multiple facilities for discovering and flagging out these bad pixels.

5.5.1 Global and Record Level Bad Pixels

There are two different types of user-settable bad pixels in CUBISM: *Global* and *Record Level*. A global bad pixel is a pixel presumed bad in *all* ‘BCD’ records. Since global bad pixels remain bad throughout the duration (or a large portion) of the spectral map, they produce “stripes” at individual wavelength planes in the final cube as the pixel is rastered across the map; an example of this effect is seen in [Figure 5.2](#). The number of stripes depends on the map; on how many steps are executed along the slit. Occasionally, partial stripes will appear, indicating a bad pixel which persisted for only a portion of the map.

The second type of bad pixel is a *record level* bad pixel, which affects only a single record. These can be useful for treating intermittent bad pixels which affect only a single or small number of records; rather than discarding that pixel’s data for the entire map, only the portion affected is removed. This is particularly useful for *BCD* frames with horizontal band artifacts (a group of rows with garbled data). The number of record level bad pixels set for a given record can be seen in the secondary pane of information in the CUBISM Project window. The total number of bad pixels marked, and their sources, is given in the project parameters (*Info->Project Parameters...*).

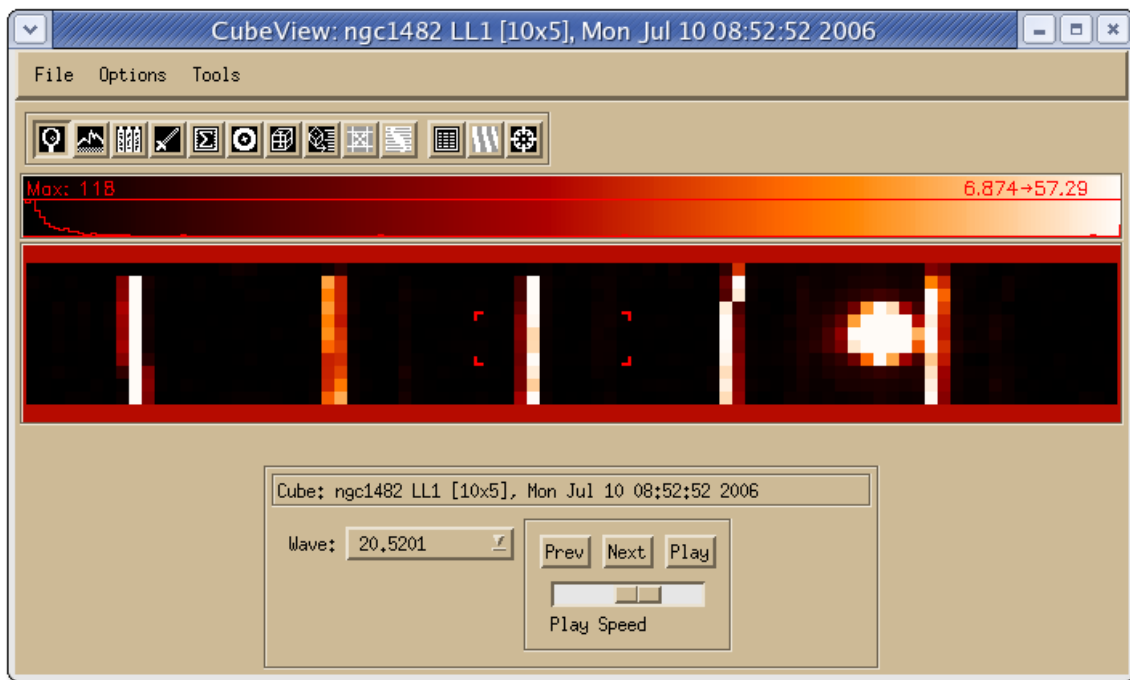


Figure 5.2: An example of a global bad pixel impressing “stripes” in the output cube.

5.5.2 Manual Bad Pixel Selection

Bad pixels can be selected manually at several levels. The most straightforward is at the ‘BCD’ level: simply examine a single or stack of records, and mark errant pixels. See

Section 4.4.3.12 [Bad Pixel Tool], page 34, for more on marking user or record level bad pixels, and on the identifying symbols used to mark pixels.

MULTIPLE RECORD BAD PIXELS AT ONCE:

Setting record level bad pixels on a stack of more than one record sets that pixel in *all* the records in the stack.

A typical workflow for manually marking bad pixels at the ‘BCD’ level:

1. View the stack of relevant ‘BCDs’.
2. Show the ‘WAVSAMP’ (see Section 4.4.8 [CubeView WAVSAMP Pane], page 39).
3. Click ‘BGSub’ (if available) to remove the appearance of many bad pixels using the record level sky background data previously defined.
4. Enlarge the stack viewer window (see Section 4.4.2.2 [CubeView Options Menu], page 23).
5. Show the user (cyan/green ‘x’) bad pixel and pipeline fatal (red ‘x’) masks only, by right-clicking two times (see Section 4.4.3.12 [Bad Pixel Tool], page 34).
6. Set the histogram scaling stretch to show only the very brightest pixels.
7. Mark >4 sigma deviations, but ensure you are not marking line data!
8. Reset the scaling to show the next brightest set of pixels.
9. Mark >4 sigma deviations.
10. Iterate this process until you can clearly see emission lines or other real features in the data, to the point where the viewable pixels are no longer >4 sigma deviations.
11. Save the bad pixel mask (see Section 4.3.2.6 [CUBISM Project BadPix Menu], page 19).
12. (Re)-build the cube with this mask (uses QuickBuild – See Section 5.9 [QuickBuild], page 57).

Note that most ‘BCD’ records contain a rim of pixels flagged with ‘[7]’, which appear as red diamonds surrounding each order. These are “flat-field questionable” pixels, but experience shows that many of these pixels are useful. They can be trimmed away by narrowing the WAVSAMP (see Section 5.7 [WAVSAMP], page 56), but aren’t considered fatal by default.

BAD PIXEL TIP:

When selecting bad pixels in cubes being built with a background, be sure to enable the *BGSub* button *before* selection, since this can mitigate many rogue pixels, which would otherwise appear bad.

5.5.3 Backtracking to Discover Bad Pixels

Backtracking is the term used for listing all the ‘BCD’ records, and pixels within those records, contributing to a given cube pixel. This is a very powerful tool for validating features in a

cube. In particular, it is useful for identifying global or persistent record-level bad pixels, which impress stripes in the cube (see [Figure 5.2](#)). See [Section 4.4.3.11 \[Pixel Backtracking Tool\]](#), [page 33](#), for more on using this tool.

A typical workflow for identifying bad pixels using the backtracking tool is:

1. Inspect each wavelength plane of the cube for out-of-place columns/rows. Using the left/right arrow keys in an extracted spectrum (Full Cube view) is a quick way to step through the cube.
2. Using the backtracking tool, select a cube pixel inside a problematic feature (e.g. a stripe).
3. Note any pixels in the backtracked data set, comparing to other cube pixels along the feature to note record pixels which are consistently outliers.
4. Right-click a given pixel to mark it as a global or record level bad pixel.
5. Re-build the cube after one or more pixels are marked to see whether the cube was improved. If not, undo and try again. With QuickBuild (see [Section 5.9 \[QuickBuild\]](#), [page 57](#)), this process is very fast even for large cubes. The displayed cube and any extracted spectra are automatically updated.

A related useful technique is to extract spectra in different parts of the cube, and look for non-physical spectral features (typically positive or negative spikes, one pixel wide, not corresponding to known lines). Using CubeSpec's *Full Cube* mode, select the affected cube plane, and look for anomalous stripes or other features. Use the backtracking tool to identify and mark bad pixels, and re-build with QuickBuild.

5.5.4 Automatic Bad Pixels

USING AUTO BAD PIX:

Since CUBISM uses the BCD pixel contributions to the cube to discover bad pixels automatically, you must first build the cube before the automatic bad pixel options are enabled.

CUBISM offers a powerful method to automatically discover and mark bad pixels in an assembled cube. Selecting *BadPix->Auto-Gen Global Bad Pixels* or *BadPix->Auto-Gen Record Bad Pixels* will show a dialog in which the parameters for the automatic selection are set (see [Figure 5.3](#)). The method by which bad pixels are automatically discovered is similar in concept to the backtracking method described in the previous section, except that *all* cube pixels are considered simultaneously, and statistics are collected on individual BCD pixels, which can be used to flag them as bad.

AUTO BAD PIX WARNING:

The automatic bad pixel tool must be used with caution. It can flag real data such as spectra lines or sharp spatial features (e.g. point sources). Always check the resulting bad pixels it finds, and avoid very low sigma thresholds or fractions.

An example of the auto-badpix parameter dialog is seen in [Figure 5.3](#). The “Sigma-Trim” option sets the rough number of standard deviations a pixel must lie away from the median record pixel value contributing. The “MinBad-Fac” lists the minimum fraction of occurrence of a given ‘BCD’ pixel in which it must deviate to be considered bad. For example, if ‘sigma-trim’ is 7, and ‘minbad-fac’ is .5, then a given ‘BCD’ pixel is considered bad if at least 50% of the time it appears in the cube, it is 7-sigma from the median pixel value.



Figure 5.3: The dialog for setting auto bad pixel parameters.

The *BG* button determines whether the test for lying outside the sigma trim threshold is performed on the background subtracted data (comparable to using the ‘Val-Back’ column in the backtracking tool). This is typically a good idea, since many pixels are not outliers when background-subtracted. The *UNC* option controls how the fiducial spread of data values is determined. If it is set, the pipeline-estimated uncertainties are used. If not, a median absolute deviation is used as an estimate for the fiducial sigma value. Since the pipeline uncertainties severely under-estimate the true spread of data above $S/N \sim 10$ or so, for bright sources, it is best not to set the *UNC* option. It can be useful for faint sources, in particular when the redundancy is small.

Record-level automatic bad pixels are similar to global bad pixels, except instead of demanding that a given ‘BCD’ pixel must deviate at least ‘minbad-fac’ of the time it appears in the cube, only *that record’s* pixel must satisfy this cut. For example, if record 20, pixel ‘(40,79)’ appears 4 times in the cube, ‘minbad-fac=.4’ requires at least 2 of these appearances to be outliers at the ‘sigma-trim’ level to consider it a bad pixel. Many more bad pixels will be generated by record-level auto-badpix, but each of them will have a much smaller impact on the final cube (since they disable that pixel for only a single record). Often, global bad pixels are sufficient. In certain cases, record-level pixels, automatically discovered, are very useful, for example for very short exposures (6s) in which cosmic rays are incompletely identified by the pipeline. This results in small 3–4 pixel stripes along rows, which can often be identified automatically with record-level auto bad pixels.

CLEARING BAD PIXELS:

Note that by default the bad pixel list is appended to with subsequent runs of auto-badpix. Clear the bad pixel list first to avoid this, and save intermediate results to ‘.bpl’ files for added flexibility.

Obviously, the amount of redundancy in a spectral map has a major impact on the degree to which outlier pixels can be automatically discovered. A ‘1xn’ map (no steps along the slit) offers a minimum amount of pixel redundancy, whereas a map stepped many times along the slit, placing the same source position at multiple positions along the slit, will offer much better redundancy. With minimum redundancy, the number of record pixel samples mapping into a cube pixel is small, and the natural spread of those samples, which results from the edges of spectral lines or sharp spatial features falling into a single spectral cube, can overwhelm bad pixels. Extreme caution must be exercised in this case to avoid marking legitimate pixels at these edges.

5.5.5 Saving Bad Pixels

The list of global and record level bad pixels can be saved as text in ‘.bpl’ files using `BadPix->Save Bad Pixels...`. This is useful for sharing bad pixel lists among projects, and providing a backup of the marked pixels outside of the cube project (recommended). Bad pixels can be loaded in from ‘.bpl’ files as well, either replacing the current set, or appending to it. The format of the bad pixel file is a list of 1D pixel indices (‘128*y+x’), prepended (in the case of record level bad pixels) by a line with two entries: the unique record ID (‘DCEID’) and the number of bad pixels associated with that record. The global bad pixel list follows.

See [Section 4.3.2.6 \[CUBISM Project BadPix Menu\]](#), page 19.

5.6 Cube Build Settings

The cube build settings are accessed in the *Cube* menu of the project window. Typically the default values (most options enabled) are fine, but for occasional uses you may prefer to disable some:

- FLUXCON** Typically you would build the cube in units of ‘MJy/sr’ using this option. It enables use of the extended source flux calibration files. One reason you might prefer to build a cube *without* fluxcon is to make a 1D background in ‘e/s’ units for loading as a background spectrum (see [Section 5.4.4 \[1D Sky Spectrum Background\]](#), page 48).
- SLCF** The slit-loss correction function is used to correct the native IRS pipeline calibration, tuned for point sources, to the case of perfectly extended sources. While most sources are not entirely smooth and slit-filling, this option should be enabled in most cases, except for point sources. Note that the CUBISM calibration of extended sources is made using the SLCF; point source spectrophotometry will be more accurate using SPICE, or another single-pointing extraction tool.

Subtract Background

Only available if a background is set, this is usually a good idea, unless you'd like to create a 1D background spectrum directly, or check the cube without background removal.

Trim Wavelengths

Trimming the unreliable order ends is never a bad idea, unless you are specifically interested in the spectrum there. Note that some additional trimming may be required; this is a conservative trim by default.

Reconstructed Positions

Unless the reconstructed slit positions are thought to be in error, use this option. Disabling it uses the pre-programmed requested slit positions, which can be off by several arc seconds.

Uncertainty Cube

Only an option if all your records have uncertainties, this builds a full uncertainty cube in parallel.

PIPELINE UNCERTAINTIES:

Note that pipeline uncertainties are statistical ramp uncertainties only, valid only in the low S/N limit. Above S/N~10–20, systematic uncertainties, not accounted for by CUBISM, dominate.

5.7 WAVSAMP

The 'WAVSAMP' is the set of *pseudo-rectangles* which define the spectral order's location on the array, the (tilted) locus of constant wavelength, and the wavelength calibration (see, e.g., [Figure 4.4](#)). CUBISM uses a custom built version of the 'WAVSAMP'. which differs slightly from that used by the SSC. The exact position and size of the 'WAVSAMP' changes with new SSC calibrations, and it typically includes a small amount (1-2 pixels) of spurious data at the extremities of the slit. For spectral maps without any stepping along the slit (*1xn* maps), leaving this *crust* of spurious data will have little impact, since it can be ignored when extracting spectra and making maps. For maps with along the slit redundancy however, it is important to exclude this data directly.

See [Section 4.4.8 \[CubeView WAVSAMP Pane\]](#), [page 39](#), for more information on editing the 'WAVSAMP'. Typically, the 'WAVSAMP' is edited, and the central handle is used to drag each side in by 2–5% (the reported 'WAVSAMP' coordinates are in normalized units, 0.0–1.0), to exclude the bulk of the spurious data at the slit ends. Non-uniform edits (different normalized range at each end of the order) are not recommended. Note that most 'BCD' records contain a rim of pixels flagged with '[7]' in their 'BMASK', which indicates the flat field is questionable there. These are not fatal bad pixels, and should not be excluded simply because they are flagged.

5.8 Building the Cube

Once the data records are assembled, backgrounds are set, and obvious bad pixels are marked, a first-pass cube can be assembled. Note that disabled records are not included in

the cube build. Frames which are completely garbled (a rare occurrence) should be disabled, as should data records which are used only for setting the background (which may be far away on the sky).

Select *Cube*->*Build Cube*, or press the *Build Cube* button, and watch the cube build progress plotted (records laid out and clipped against the sky grid, and then final cube pixels assembled from the record data). Typically several iterations of cube building are performed, refining the background and bad pixel selection, before the cube is considered complete.

Once the cube has been built once, subsequent builds will proceed much more quickly, since the “account” information, which includes all the clipped pixel mapping from ‘BCD’ to the sky grid, is saved. Changing the ‘WAVSAMP’, build order, or manually resetting the accounts with *Cube*->*Reset Accounts* will invalidate the accounts, which will need to be reconstructed in a slow cube build. Changing backgrounds, bad pixels, etc. can be done without invalidating the accounts.

5.9 QuickBuild

As a special case of cube rebuild, changes to the bad pixel lists (global or record level) enable a special mode called ‘QuickBuild’. The *Build Cube* button and *Cube* menu items in the project window will change to reflect this. In this mode, only cube pixels which are affected by *dirty* BCD pixels (those recently added or removed from one or more bad pixel lists) are recomputed. The status bar mentions the fraction of cube pixels which are being re-built. This is much faster than a full cube rebuild, typically requiring only a second or two even for large projects.

QuickBuild means that experiments with bad pixels to improve the quality of the cube (see [Section 5.5 \[Bad Pixels\]](#), page 50) can be performed very rapidly. This is particularly powerful with back-tracking (see [Section 5.5.3 \[Backtracking to Discover Bad Pixels\]](#), page 52), since you can try out a bad pixel, quick-build the cube, and the cube view and any extracted spectra are automatically updated. If the problem isn’t solved, undo and try again.

5.10 Saving the Project

Saving the project to a ‘.cpj’ project file records all of the information presented in a CUBISM Project window, including data records, background settings, bad pixels, cube build preferences, calibration set details, etc., and allows you to recover a cube project from disk to continue working on it. Saving the cube can be accomplished with menu option *File*->*Save*, or *File*->*Save As...* to save a version under a different name, or with the convenient *Save* button below the record pane. *File*->*Revert to Saved...* allows you to discard any changes to the project, and restore the version most recently saved to disk.

Not all of the data is saved with the project by default. There are three options related to saving the project file, accessible in the *File*->*Save Setup* menu:

1. *Save Data with Project*: This saves a copy of all the loaded data (including ‘BCD’, associated masks, and uncertainties) into the project file. This adds approximately 192kB per record to the size of the file, and is only necessary when you intend to distribute the file independent of the data directories. This can be useful for sharing ‘.cpj’ files with colleagues without access to the Spitzer archives, but can result in

large files when many tens or hundreds of records are involved. It is also redundant when the data files themselves are already available.

2. *Relative File Names*: By default, the absolute paths of the data files are stored in the project, and, if the data themselves are not stored, they will be recalled from disk when needed. Unfortunately, this makes cube project files non-portable (leading to errors like “Couldn’t restore data from file”), unless the absolute path to the ‘BCD’ files is fixed on all systems where the cube is loaded. While you can easily replace the root directory from the file names to fix this issue (see [Section 7.3 \[Troubleshooting\], page 70](#)), another method is to enable this option to specify that all path names will be saved *relative to the path of the ‘.cpj’ file itself*. Then, as long as the relative file layout of the project file and data files remains the same, all record data should load without problem. This has the advantage of allowing cubes to be passed back and forth without needlessly re-transferring all the data, but also not requiring any absolute file layout for the data on disk. For instance, a full directory structure as:

```
myproj/myproj.cpj
myproj/data/r123456/ch0/bcd/...
myproj/inputs/myproj.bpl
myproj/inputs/myproj.bgl
myproj/myobject_8um_viz.fits
```

could be distributed as a unit, with ‘myproj.cpj’ referencing the data in the ‘data’ directory, a visualization image, and even including saved copies of the bad pixel and background list inputs.

3. *Save Accounts with Project*: The clipping accounts consist of all the information which maps ‘BCD’ pixels to the sky grid. When the accounts are valid, (you don’t see *needs rebuilding* in the *Info->Project Parameters...* page), the cube can be rebuilt quickly, and pixel backtracking can be used (see [Section 4.4.3.11 \[Pixel Backtracking Tool\], page 33](#)). Saving this account information in the project file gives these benefits to cube projects loaded from disk as well, but comes with a steep cost in terms of disk space: approximately 400–700kB per record.

For distributing CUBISM projects as ‘.cpj’ files, it’s easiest to save the project without data or accounts (the default), since this produces the smallest files by far. However, this requires everyone to have access to the data, in the same path (for absolute file names), or with the same relative path layout (for relative file names) as on the build host.

One approach for sharing the data and cube project together is to save the data in the project itself. This creates a larger project file, but is fully self-contained. Another reasonable approach is to use relative file names, and bundle the ‘.cpj’ together with the data directories and visualization image (if any) in a single directory for distribution. The advantage of the latter is it allows the ‘.cpj’ file to be updated independent of the data themselves.

For users who simply wish to examine the cube itself (extract spectra, make maps), access to the record data isn’t necessary — any ‘.cpj’ with an assembled cube will do, or even just the FITS Cube (see [Section 5.12 \[Reading a FITS Cube\], page 59](#)).

5.11 Saving the Cube as FITS

The cube can be saved as a 3D FITS file for further analysis, using *File->Write FITS Cube...* If an uncertainty cube exists, it will be saved alongside the main cube, with `_unc.fits` replacing `.fits`. Both FITS files include headers from the first assembled `BCD`, and an `NAXIS=3` FITS cube in the primary data unit, with WCS header records recording the astrometric positions for the first two (celestial) dimensions, and the third (wavelength) dimension delineated by a lookup table, available in the first cell of the binary table in the single FITS extension. This dimension coordinate description follows the lookup table spectral format of Greisen, E.W. et al., A&A 446, 2006.

5.12 Reading a FITS Cube

A saved FITS cube can later be read from disk for immediate display using CubeView (see [Section 4.4 \[CubeView\], page 23](#)). No project window will be displayed, instead the cube will be viewed directly in CubeView, with a name ending in `[FITS]`. The associated uncertainty cube will also be loaded if available. Note that most of the features associated with a full CUBISM project will be unavailable when viewing a cube loaded from a FITS file, including the CUBISM project interface, record and overlay visualization, bad pixel lists, pixel backtracking (see [Section 4.4.3.11 \[Pixel Backtracking Tool\], page 33](#)), etc. Cubes cannot be (re)-built from versions saved as FITS, since they do not contain all the information on input data records, bad pixels, backgrounds, calibration settings, etc. Spectra can be extracted and maps can be created and saved from a FITS cube; see [Chapter 6 \[Cube Analysis\], page 60](#). Except for low-level analysis of completely validated cubes, it is preferable to work with full cube project files (see [Section 5.10 \[Saving the Project\], page 57](#)).

LOAD FITS CUBES:

By default, when opening a cube from *File->Open...* or when calling `'cubism'` directly, only `.cpj` files will be displayed. To load a FITS cube, select the filters list to display `.fits` files, and ensure you select a 3D FITS cube generated by CUBISM.

6 Cube Analysis

Once a given cube is fully built and validated, it can be used to generate 1D line extractions and build arbitrary 2D spectral maps, both of which are straightforward using the CUBISM interface. These types of cube analysis can be performed either from a full CUBISM project, or from a 3D FITS cube (with some limitations).

6.1 Extracting 1D Spectra

To generate a 1D spectrum, you must first identify the spatial region over which the spectral cube is to be averaged. This can be done in one of two ways: directly defining a rectangular aperture on the cube, or from a matched extraction region read in from an existing ‘.tbl’ spectrum file or DS9 region file.

6.1.1 Direct Extraction

See [Section 4.4.3.10 \[Cube Extraction Tool\]](#), page 33, for more information on defining a rectangular extraction aperture on a cube. Briefly, choose the extraction tool (hit **x** in the CubeView window displaying the cube), drag from top-left to bottom right to define a rectangular region, and a CubeSpec window opens showing the flux intensity spectrum averaged over that region, in the units specified in the cube build settings (see [Section 5.6 \[Cube Build Settings\]](#), page 55). If the CubeSpec window was already open, the spectrum will change to reflect the newly-defined region, and the area of extraction (in cube pixels) is displayed at the top of the CubeSpec window.

When you move or resize the region, the spectrum will update automatically. This is an excellent way to quickly see how the IRS spectrum of your source changes with position over the cube, or extraction size. Remember that the arrow keys allow the selection box to be moved one pixel at a time in CubeView (or 5, or 10 with **.**).

Any extracted spectra can be saved using *File->Save Spectrum As...* (see [Section 4.5.2.1 \[CubeSpec File Menu\]](#), page 41). The save format is an ASCII file with encoded headers, column delimiters, and units, in the IPAC table format (the utility routine `read_ipac_table` supplied with CUBISM can read this format). Any ‘.tbl’ file saved in this way can also be used to define the region of extraction for another cube — see below.

EXTRACTION APERTURE SIZE:

Avoid using extraction apertures smaller than roughly ‘2x2’ pixels, since over small regions, pixel aliasing driven by the undersampled IRS PSF can lead to anomalous continuum “sawtooth” patterns near high spatial frequency regions within the cube.

6.1.2 Matched Region Extraction

Matched extraction is the extraction of spectra from a cube in an aperture defined elsewhere. All spectra saved by CUBISM include header information giving the celestial coordinates of the region of extraction. This aperture can be reused to recover spectra from different cubes built in the same region. It is most commonly used to generate single, fully stitched

spectra by matching the physical extraction region between different cubes in different IRS modules or orders: e.g. matching SL1 and SL2, or SL1 with LL1, SH with LH, etc., and import non-rectangular extraction regions defined elsewhere. It can also be convenient to recall existing extraction regions from the same cube.

MATCHED APERTURE SIZE:

When defining an aperture intended to be matched across multiple IRS modules, ensure it is large enough to avoid pixel aliasing in all modules being extracted.

In the CubeView window displaying your cube, use *File->Extract Region from File...* to load a saved region file, either in the form of a `.tbl` spectrum file created by CUBISM, or a (limited subset of) `.reg` DS9 region file. Once this is done, the spatial region will be displayed on your current cube in the CubeView window, and a spectrum will automatically appear in a new CubeSpec window. Aperture regions loaded from file in this manner are not directly editable.

Note that since cubes are built with position angle parallel to the IRS slit at the time of observation (which rotates throughout the year), in general extraction apertures recovered from other slits, or spectral maps executed at different times, will not align with the rows or columns of the cube. The aligned apertures are clipped against the cube to generate the partial-pixel weighted average over the region. The same is true for polygonal and circular apertures (which are converted to polygons). Regions in DS9 can be defined on any image with valid coordinates, which can be useful for extracting areas based on additional information (e.g. an X-ray hot-spot, etc.).

DS9 REGIONS:

Only a very specific subset of DS9 region files is supported: circles, and polygons of any length. The output file format *must* be set to DS9, WCS coordinates, J2000 (FK5). In addition, only *one* polygon or circle per region file is read (the first). All other region information is quietly ignored. Note that self-intersecting polygons create multiple polygons in a region file (all but the first of which will be quietly ignored). Attempting to read any other type of DS9-generated region file will result in an error. See DS9's *Region* menu and documentation.

The name of the file from which the aperture was loaded appears at the top of the CubeSpec window. The new spectrum can now be saved in the same manner as described above, and its header will contain the file from which its extraction region was drawn.

Matched extraction can be used to extract Long-Low spectra using a region defined in the SL1 or SL2 cubes, thus obtaining a complete spectrum over a single, pseudo-aperture in the map. Since Spitzer is diffraction-limited beyond about 6 microns, and the IRS modules have different pixel scales, users should be very careful to avoid defining critically-sampled

regions at the shortest wavelengths, then using these to extract spectra from maps with significantly larger beam sizes. This may result in long-wavelength spectra generated over areas smaller than the beam. Also be aware that diffraction smoothes the angular source distribution to greater degrees at longer wavelength, so that a single extraction aperture can probe slightly different physical regions depending on wavelength. This effect is particularly pronounced with sharp spatial structures: e.g. an extraction formed inside a small ring of emission at the shortest wavelengths will likely be contaminated by ring emission at longer wavelengths.

If you read in a region that is larger than your current cube, and attempt to extract a spectrum over this region, CUBISM will produce a warning message. In this case, the problem will be obvious, as the extraction aperture will extend beyond the edges of your cube in the CubeView window. You can still generate the spectrum and save it to disk, but you should probably select a smaller aperture if you are attempting to compare or stitch spectra across IRS modules. Similarly, regions entirely outside the cube will generate an error.

SPECTRA MISMATCH:

When comparing spectra extracted from the same physical region, flux offsets between modules and orders can occur. To minimize this, ensure the background was subtracted in the same manner for all cubes being compared, and that the extraction region was large enough.

Slight astrometric errors in the final cube astrometry (limited by the spacecraft pointing system's precision) will compound offsets between matched spectra in very small regions near steep spatial gradients in the source.

6.2 Creating 2D Maps

Just as the cube can be used to generate 1D spectra over any extraction region, so too can the 1D spectra be used to create 2D images (slices of the cube) averaged or integrated over any specified wavelength interval. See [Section 4.5 \[CubeSpec\], page 40](#) for more detail on defining regions on the displayed spectrum.

SAVE MAP AS FITS:

Any spectral map produced by CubeSpec can be saved to a FITS file by selecting *File->Save Map as FITS...* from the CubeView menu.

6.2.1 User-defined Maps

Once a 1D spectrum is visible in the CubeSpec window, change the selection from *Full Cube* to *Map*, and select either *Peak* or *Continuum* (the convenient key shortcuts *p* and *c* set both of these at once and permit quickly switching between peak and continuum region definition). A spectral map consists of any number of peak and continuum regions. The

continuum regions are averaged, and then subtracted from the average or integral of planes over the peak regions. Continuum regions are optional.

For example, if you wish to create a continuum-subtracted 11.3 micron PAH map from a SL1 cube, you can first use the mouse to define the continuum on either side of the PAH feature. Once the continuum regions are defined, change the selection mode to *Peak*, and select the emission feature with the mouse in the same way. For emission line maps, select at least three or more pixels around the peak of the line. Once the peak region is selected, it will show up in (a lighter shade of) red, and the image in the CubeView window will change to reflect the newly defined map. This is now the continuum-subtracted map of your science target. The CubeView window will indicate the type of map being shown, the range of peak wavelength coverage.

SPECTRAL REGION TYPES:

Though they are called *Peak* and *Continuum*, the two region types can be used to make arbitrary differential maps averaged over different wavelength intervals.

Once regions are defined, they can be selected by middle-clicking, SPACE cycles selection among them, moved with middle-click-drag, or with the left/right arrow keys. Up/down arrow increase or decrease the width of the region. Regions which come in contact with each other are merged. When regions are re-sized or moved, the accompanying map is automatically updated. Defining a small peak region (several pixels), and then moving it across the spectrum is a good way to get a quick feel for how the cube is changing with wavelength.

To save any map generated in this way, use *File->Save Map as FITS* in the CubeView window where it is displayed. The FITS file saved in this manner has all the required header keywords to be read into the common FITS readers (e.g. DS9), displayed, and aligned with any other FITS image of your science target. Using the CubeSpec window, any number of spectral regions can be defined and used to make images, which can then be saved to FITS from the CubeView window. The FITS headers will include the wavelength peak and continuum regions used to generate the map.

6.2.2 Pre-defined Maps

Besides user-defined maps, CUBISM offers a number of pre-defined filter curves available under the *Maps* menu in the CubeSpec window. This will allow you to generate spectral maps in, for example, any the IRAC-5.8, IRAC-8, MIPS-24, IRS Peakup, ISOCAM LW2 or LW3 filters. Once selected, the spectral ranges will be highlighted on the 1D spectrum in the CubeSpec window, and the image in the CubeView window will change to the corresponding 2D map. Note, these maps are pseudo-filter images, and are not continuum-subtracted, and may lie partially outside the wavelength range of the spectra. Unlike maps based on wavelength regions, maps based on filter curves cannot be directly edited in CubeSpec. A selected map will appear at far right in the title of the plotted spectrum.

6.2.3 Map Sets

User-defined map sets can be saved for later use using the CubeSpec *Maps* menu. A given set of peak and continuum regions can be saved as a `.map` file using *Maps->Save Current Map...*, and later recovered. Once saved or restored, it will appear below the default pre-defined maps at the bottom of the *Maps* menu. For example, a map defined in LL1 on the [SiII] line with adjacent continuum could be saved as `siII.map`, and later re-used to create similar [SiII] image from another LL1 cube. To make a map saved in this way one of the default maps, move it to the `‘cubism/map_sets’` directory.

6.2.4 Redshift and Maps

Sources at non-negligible redshift can have saved maps automatically shifted into the observed frame by specifying their redshift `‘cz’` in `‘km/s’` with *Maps->Set Redshift...*. The currently set redshift will be displayed at right in the plot title window, and will be used to shift any saved map set to the rest frame. For example, in the [SiII] map example above, specifying a redshift will shift the saved map to track the line in sources at higher redshift.

Note that pre-defined map sets based on filter curves are not shifted in this way (since they are meant to define pseudo-images simulating direct observation).

6.2.5 Integrated Maps

By default, CubeSpec produces average surface brightness maps, typically in units of `‘MJy/sr’`, averaged over all foreground wavelength planes (after optional continuum subtraction). If the *Integrate* button is selected, the foreground region(s) will instead be integrated over $\sum f_\nu d\nu$, which will change from flux density to flux units, e.g. `‘W/m2/sr’`. For weight maps, e.g. the MIPS 24um or IRAC 8um maps, the frequency-spacing weighted integral $\sum f_\nu w d\nu / \sum w d\nu$ is performed instead, where w is the weighting vector (e.g. filter transmission function). Often this results in only a small difference in the resulting map image.

6.2.6 Wavelength Weighting

If the *Lam-Weight* button is selected, rather than average all continuum planes together, CubeSpec performs a weighted average for each foreground plane, with the weight set to $1/(1 + |\lambda_f - \lambda_c|)$ where λ_f is the wavelength of the given foreground plane, and λ_c is the wavelength of a continuum plane. In this way, each foreground plane gets a custom background with more weight given to the closest continuum regions. This can be useful when averaging multiple peaks with a strongly varying continuum.

6.2.7 Line Fits

The CubeSpec window can also be used to measure simple line parameters before or after the line map is made. By selecting the *Fit* button, a polynomial fit, as defined by the *Fit Order*, will be performed between the continuum points, and the basic parameters (e.g. line equivalent width, line flux, average continuum flux density, etc.), will be displayed in the box in the upper right of the CubeSpec menu. By clicking on the *Reset Plot* button, all the continuum and peak selections will be erased, and new spectral regions, and their corresponding spectral maps, can be defined. Such fits are for informational purposes only, and do not affect the map displayed.

6.3 Complex Maps

CUBISM offers only map creation based on simple averages or wavelength-offset-weighted averages among planes of the cube. For higher order map creation, e.g. to create derived maps by fitting multiple components to at each position, it is convenient to export the full cube to the IDL command line or as a separate FITS file. See [Section 5.11 \[Saving the Cube as FITS\]](#), page 59, for more information.

7 Tips and Troubleshooting

Here we document issues which are commonly encountered, and provide tips on debugging CUBISM and providing useful feedback to the SSC.

7.1 Mouse and Keyboard Shortcuts

All collected shortcut keys and mouse actions for all three major tools are assembled here.

7.1.1 Cube Project

UP/DOWN Arrows	Move among records.
PAGEUP/PAGEDOWN	Move record lists by pages.
<i>Shift-click</i>	Select range of records.
<i>Control-click</i>	Select additional record.
<i>Control-click-drag</i>	Select additional range of records.
<i>Control-shift-click</i>	Select additional range of records (from last click).
RETURN	View record (same as double clicking).

7.1.2 Cube View

SPACE	Lock the current scaling limits.
z	Toggle zoom tool.
	<i>Click and release</i>
	Zoom-in on region 2x.
	<i>Click-drag</i>
	Zoom in on defined rectangular region.
	<i>Right-click</i>
	Zoom out one level.
	<i>Right-double-click</i>
	Zoom out all the way.
	<i>Middle-click</i>
	Recenter on point.
	<i>Control-click</i>
	Recenter on point.
	<i>Middle-click-drag</i>
	Pan image (if zoomed in).
	<i>Middle-click-drag + Shift</i>
	Pan image, constrained to horizontal or vertical (if zoomed in).
	<i>Control-click-drag</i>
	Pan image (if zoomed in).
	<i>Control-click-drag + Shift</i>
	Pan image, constrained to horizontal or vertical (if zoomed in).

- h** Toggle box scale histogram tool (on twice to reset).
- Click-drag* Define scaling box. On handle, resize box.
 - Arrows* Move box 1 pixel (with Shift: 5 pixels, with Control+Shift: 10 pixels)
- c** Toggle color adjust tool.
- Click-drag* Change color map (up/down=narrow-widen, left/right=shift).
 - Right-click* Reset color map.
- l** Toggle line slicing tool (on twice to reset).
- Click-drag* Define slice line
 - Control* Constrain Slice line to be vertical, horizontal, or diagonal.
 - Right-click-drag* Constrain Slice line to be vertical, horizontal, or diagonal.
 - Middle-click* Define strip width for averaging.
 - Middle-click-drag* Update averaging strip width.
- s** Toggle statistics tool.
- Click-drag* Define statistics box. On handle, resize box.
 - Arrows* Move box 1 pixel (with Shift: 5 pixels, with Control+Shift: 10 pixels)
- p** Toggle photometry tool.
- Click-drag* Define photometry box. On handle, resize box.
 - Arrows* Move box 1 pixel (with Shift: 5 pixels, with Control+Shift: 10 pixels)
- x** Toggle cube extraction tool (on twice to reset, viewing cubes only).
- Click-drag* Define extraction box. On handle, resize box.
 - Arrows* Move box 1 pixel (with Shift: 5 pixels, with Control+Shift: 10 pixels)

- t** Toggle cube back-tracking tool (on twice to reset, viewing single cube planes only).
- Click* Freeze backtrack on current pixel.
 - Right-click*
 - Release freeze.
- b** Toggle bad pixel tool (on twice to reset, bcd only).
- Click* Add/remove global bad pixel.
 - Middle-click*
 - Add/remove record-level bad pixel (for all records in stack).
 - Control-click*
 - Add/remove record-level bad pixel (for all records in stack).
 - (Middle/Control-)Click-drag*
 - Add/remove multiple bad pixels at once (add vs. remove based on initial point).
 - Right-click*
 - Rotate through the four bad pixel view sets.
- v** Toggle AOR visualization tool (on twice to reset, visualize image only).
- Click* Select record (will be outlined in white). Outside of record overlays, select no records.
 - Click-drag*
 - Select range of records.
 - Shift-click*
 - Select range of records from last selected.
 - Control-click*
 - Select additional record.
 - Control-click-drag*
 - Select additional range of records.
 - Control-shift-click*
 - Select additional range of records.
- t** Toggle displaying pixel table.
- w** Toggle masking all off-order data (BCD only).
- 1..4** Set size of view window to 245, 384, 512, 768 pixels.
- 0** Set size of view window to “wrap” full image at current zoom (if possible).

7.1.3 Cube Spec

<i>Click</i>	Define beginning or ending point for region.
<i>Right-click</i>	Cancel region selection, or zoom out.
<i>Middle-click-drag</i>	Move peak or continuum regions.
<i>u</i>	Full cube mode (single wavelength plane at a time).
<i>m</i>	Map mode.
<i>x</i>	X-zoom mouse region (click twice to zoom in on wavelength).
<i>y</i>	Y-zoom mouse region (click twice to zoom in on spectrum flux).
<i>l</i>	Select wavelength plane mode.
<i>c</i>	Select continuum region define mode.
<i>p</i>	Select peak region define mode.
<i>v</i>	Toggle displaying the value line.
<i>d</i>	Delete selected region
SPACE	Select next region.
UP/DOWN Arrows	Enlarge/reduce peak or continuum region size.
LEFT/RIGHT Arrows	Move selected region or wavelength plane by one step (with Shift: 5 steps; with Control+Shift: 10 steps).
<i>f</i>	Perform line fit (peak and continuum regions required).
<i>r</i>	Reset plot.
<i>q</i>	Close CubeSpec window.

7.2 Tips

The most common and useful tips:

1. **Save Often.**

You can always revert to the file on disk, or save multiple versions of a project under different names.

2. **Save bad pixels and background records.**

You can save your bad pixel and background records in separate files from your project ‘.cpj’ file. These are ‘.bpl’ (for bad pixels) and ‘.bgl’ (for backgrounds). You can then load these files into a new project file and save yourself time, if you are breaking a large Project into several smaller project files, and they are a useful backup in the case of loss or damage to the ‘.cpj’ file.

3. Use auto-badpix with care.

This can be a big time saver, but can mark valid data bad if used improperly, especially on maps without much cube-level redundancy.

4. Check spurious spectral features using backtracking.

Unexpected positive or negative spikes in an extracted spectrum often indicate remaining bad pixels. Use backtracking to track them down — see [Section 5.5.3 \[Backtracking to Discover Bad Pixels\]](#), page 52. Quick-build, and the cube and spectra are instantly updated.

5. Avoid over-zealous bad pixel removal.

Sources with strong high-frequency spatial structure (e.g. point sources) will necessarily have a broad range of data contributing to some pixels in the cube (in particular due to the limited PSF sampling). Marking all outliers of any sort bad, either by hand or using auto bad-pixels, will result in depressed or elevated “chunks” in the spectra with no obvious cause. Using backtracking, try un-setting related global bad pixels and quick-building.

6. Use keyboard shortcuts to speed up the interface.

Keyboard shortcuts are available in CubeView (e.g. *z* to switch to the zoom tool, *SPACE* to freeze the scaling range, etc.), in CubeSpec (e.g. *p* to enter peak region define mode, arrows to move or resize regions, etc.), and in the CUBISM Project (e.g. arrow keys to change the record selection, *RETURN* to view it). Arrows move the active item and can be accelerated by holding *SHIFT* or *CONTROL + SHIFT*. See [Chapter 4 \[The Tools\]](#), page 9, for more information, and [Section 7.1 \[Mouse and Keyboard Shortcuts\]](#), page 66 for a handy table.

7. Use multiple viewers.

This allows you to simultaneously scrutinize ‘BCD’ records, the built cube, and an AOR visualization image. Using color maps to theme them allows viewing data from multiple projects without confusion.

8. Use the AOR visualizer to examine in-situ backgrounds.

This lets you quickly check for source contamination (ideally using 8 or 24 micron Spitzer image). Since the selected records are indicated in white in the AOR visualization, and viewing the background selects the background records, it’s easy to get a quick view of where the background is being drawn from. See [Section 4.4.3.13 \[AOR Visualization Tool\]](#), page 37, for more.

7.3 Troubleshooting

Commonly encountered problems and their solutions:

1. I get the message “Couldn’t restore data from file...”.

By default, to save space, CUBISM does not save the BCD and associated data itself in the ‘.cpj’ project file, instead recording a file path *reference* to the files. If you’ve

since moved or modified the BCD files, or are loading the project on another system where they are in a different location, you can *Edit->Select All*, then *Edit->Replace File Substring...* to correct the base pathname (you can use regular expressions in the search). Another option is to store the data inside of the project, or save it with *relative* pathnames. See [Section 5.10 \[Saving the Project\]](#), page 57.

2. I'm attempting to upgrade a project from pipeline version S14 to S15, but the project data won't load.

The SSC increments a version number in the filename for all processed files every time the pipeline processing changes. You can use the same technique as in the last tip to, e.g. Find `'_4_'` and replace with `'_5_'` (or whatever the relevant version numbers happen to be).

3. CUBISM freezes.

There a number of reasons why CUBISM could freeze, but commonly it only *appears* to be frozen, due to a dialog which is awaiting your attention, and must be dismissed before the rest of the program is responsive. Alternatively, if you are using the command-line version of CUBISM, you may have run other routines at the command line which have halted due to errors. Usually, `'retall'` takes care of these cases and restores response. See [Section 7.4 \[Debugging CUBISM\]](#), page 73, if these are not the problem.

4. My CubeView window gets overwritten when I visualize or display cubes or records.

You can have as many CubeView windows as you want associated with a cube. By default, however, CUBISM recycles CubeView windows, to avoid too much clutter on your display. If you want a new window, for instance to visualize an AOR or cube, while examining data records, use the *(new viewer)...* version of the view commands. See [Section 4.3.2 \[CUBISM Project Menus\]](#), page 14.

5. My cube seems to have skipped some rows, or seems to be missing data.

There are two possible reasons for this. One is that you disabled some of your records when making the cube (see [Section 4.3.3.2 \[Record Enabled State\]](#), page 21). The other is that your initial map was not properly sampled. We have seen at least one example of a map that was done using full slit spacings which was not contiguous. The most likely reason for this is the pointing uncertainty in making the observations. If you do full slit spacings, it is possible that small gaps may result in your final map. To avoid this, please see the [IRS Spectral Map HOWTO](#) for information on observation planning.

6. My cube seems to be built in the wrong spectral order.

Did you double check the cube build order (*Cube->Set Cube Build Order...*)? Remember that saving the cube with a different name does not change the cube build order, and that the target order of the data does not control the build order (see [Section 5.2 \[Build Order\]](#), page 45).

7. I have 4 different projects open, and I can't keep track of what is what.

Look in the title and status bars of each tool for information on the source cube. Consider adopting a different color table for all CubeView tools displaying a given project's data.

8. I get a message about missing calibration files.

You are probably using another cube built with a version of CUBISM newer than yours, and need to update your version. See *Help->About Cubism* to check the local CUBISM version, as well as the version which was used to build the loaded cube project.

9. **I removed all data records, loaded more, and built a new cube, but something isn't right.**

Remember that a cube project includes saved information on bad pixels, background records, etc. Check *Info->Project Parameters...* to get a top level view of the settings. Set the cube build order, and clear bad pixels and background records if necessary, or (easiest) just start from a clean, new cube project.

10. **My source emission disappears when I use outrigger sky data and then click BGSub in the viewed stack.**

Make sure that you are subtracting records with background information in the the correct order. You might have inadvertently selected records pointed to the first order for your sky when you actually meant second order, for example, causing target self-subtraction.

11. **I get a message 'Failed compiling DLM clipper...'**

CUBISM makes heavy use of polygon clipping to map all the partial BCD pixels onto the sky grid, so to speed up this crucial operation, a small piece of clipping code is written in C, and automatically compiled as needed. If your system is not configured with a C compiler (typically `gcc`), this can fail, generating this message. An equivalent fallback IDL-native version of the clipping algorithm will take over, but it is much slower (10–20x). See the documentation for IDL's `MAKE_DLL` to learn more about compiler configuration on your system.

12. **Selecting *Help->Cubism Manual* doesn't do anything.**

On Unix systems, IDL opens PDF files using its distributed script `'online_help_pdf'`. By default, this script uses `'acroread'` to load the specified file ('Preview' on OSX). If you don't have Acrobat reader installed or would prefer to use another reader, you can specify another PDF reader by setting the environment variable `IDL_ONLINE_HELP_PDF_CMD`, e.g. (in your `'.cshrc'`):

```
setenv IDL_ONLINE_HELP_PDF_CMD xpdf
```

13. **I get an IDL compile syntax error like:**

```
(*self.DR)[which].BCD=ptr_new(readfits(file,/SILENT,hdr))
                                     ^
% Syntax error.
```

You have an outdated version of `AstroLib`, update it.

14. **I can't understand the difference between build order and the record order shown under the 'Type' column in the project viewer.**

You need to read [Section 5.2 \[Build Order\]](#), page 45.

15. **All my colors come out red in the viewer, cube build visualizer, everywhere!**

You are running CUBISM from source and forgot to turn off decomposed color, which is usually done in the `idl` setup. See [Section 2.3 \[Setup\]](#), page 3. Binary versions of CUBISM running under the IDL virtual machine do this for you.

7.4 Debugging CUBISM

By default, CUBISM traps and reports all errors which occur in a dialog. The resulting error message may or may not allow you to determine the root of the problem. To debug CUBISM yourself, or to provide useful information to the SSC help desk to allow them to investigate the problem, enable debugging with *Info->Debug Cubism* in the CUBISM Project window. This requires access to the IDL command line, which will not be available for versions of CUBISM running under the IDLVM (see [Section 2.2 \[Binary Installation\]](#), [page 3](#)). After enabling debugging, repeat the action which generated the error. Instead of trapping the error with a dialog prompt, IDL will halt, with a full back-trace message showing the exact code path which led to the error in the IDL shell. This code path might be enough to give you a hint as to the nature of the problem, but is also useful for the upstream maintainers for debugging purposes.

Report problems at the Spitzer Science Center helpdesk, including, at minimum:

1. A brief statement of the problem.
2. A detailed description of the steps which can lead to the problem, isolated to the smallest necessary number.
3. Example inputs which trigger the problem, if applicable (e.g. a FITS file which gives trouble).
4. The full traceback message reported in the IDL console window after you have enabled Debugging.

If CUBISM freezes and you cannot make any progress, try *CONTROL-c* at the command line, which should force it to break (possibly after generating some input in the GUI). The backtrace at this location can be useful for understanding why CUBISM was halted.

Index

- [
[n], as bad pixel code 38
- ## A
- AOR visualization 37
Aperture photometry tool 32
- ## B
- Background menu, CUBISM project 18
Background subtraction 46
Background, archive 47
Background, blended 48
Background, in situ 47
Backtracking tool 33
Bad pixel codes 38
Bad pixel tool 34
Bad pixels 50
Bad pixels, automatic selection 53
Bad pixels, manual selection 51
Bad pixels, saving 55
BadPix menu, CUBISM project 19
'BCD' 21
'BCD' files 44
'BCD' records 20
Box statistics tool 30
Build Order 45
Building Cubes 56
Buttons, CubeSpec 42
Buttons, project window 22
- ## C
- Calibration sets 44, 46
Colorbar 38
Compass rose tool 38
Cube analysis 60
Cube assembly 44
Cube menu, CUBISM project 17
Cube, Building 56
CubeSpec 40
CubeView 23
CUBISM project 13
Cubism tools 9
- ## D
- Data records 20
Debugging CUBISM 73
Disabled, records 21
Display FITS cube 59
'DroopRes' 21
DS9 Regions 61
- ## E
- Edit menu, CUBISM project 15
Enabled, records 21
Extraction Regions 60
- ## F
- File Menu, CubeSpec 41
File menu, CubeView 23
File menu, CUBISM project 14
File names, relative 58
File selection 11
Files, input 44
FITS, reading cube from 59
FITS, saving cube as 59
FITS, saving spectral map as 23
'FlatAp' 21
- ## H
- Help menu, CUBISM project 19
Histogram tool 28
- ## I
- Image scaling 28
Image slicing tool 28
Info menu, CUBISM project 19
Input files 44
Installation 2
Installation, binary 3
Installation, from source 2
Integrated maps 64
Interface tips 10
Introduction 1
- ## K
- Keys, shortcut 66
- ## L
- Line fits 64
- ## M
- Map sets, saving 64
Maps Menu, CubeSpec 41
Maps, creating 62
Maps, integrated 64
Maps, pre-defined 63
Maps, user-defined 62
Memory Requirements 4

Menus, project window	14	Shortcut keys	66
O		Sky, 1D + 2D	49
Options menu, CubeView	23	Sky, 1D spectrum	48
Order mask tool	38	SLCF	55
P		Slicer tool	28
Photometry, aperture	32	SMART, using CUBISM with	3
Pixel backtracking	33	Spectra, extracting	60
Pixel table tool	38	Spectra, fully stitched	60
PNG, saving image as	23	Spectra, Viewing	40
Project window buttons	22	Spectrum, plotted	43
Project window menus	14	Statistics, image	30
Project, CUBISM	13	Status line, CubeView	38
Q		T	
Quick Start	6	Tips	69
QuickBuild	57	Title bar, project	14
R		Tool palette, CubeView	24
Record information	20	Tools menu, CubeView	24
Record menu, CUBISM project	15	Tools, CubeView	24
Record operations	21	Troubleshooting	66, 70
Redshift, map creation	64	U	
Relative file names	58	Upgrading	5
Rogue pixels	50	V	
Running CUBISM	4	Viewing cubes	23
Running, binary distribution	4	Viewing maps	23
Running, source distribution	4	Visualization, background selection	50
S		Visualizing AORs	37
Saving image as PNG	23	W	
Saving projects	57	Wavelength Weighting	64
Saving, with accounts	58	WAVSAMP	56
Saving, with data	57	Z	
Settings, cube building	55	Zooming images	27
Setup, of CUBISM	3		