EURO 3D

Promoting 3D spectroscopy in Europe

www.aip.de/Euro3D

RESEARCH TRAINING NETWORK

Sponsored by the EUROPEAN COMMISSION

E3D, Euro3D Visualization Tool – v1.2b

——

User Guide

Issue 1.2

19/04/2004

DRAFT

Authors S.F.Sánchez
Contact S.F.Sánchez (ssanchez@aip.de)

This page was intentionally left blank

Change Record

Issue	Date	Section affected	Reason/Initiation/Documents/Remarks
1.0	22/04/2003	all	Creation by S.F.Sanchez .
1.1	01/08/2003	External Packages	Updated by S.F.Sanchez .
1.2	30/12/2003	all	Updated by S.F.Sanchez .
1.2b	19/04/2004	all	Updated by S.F.Sanchez .

This page was intentionally left blank

Contents

This page was intentionally left blank

1 Introduction

1.1 Purpose and scope

The Euro3D Research Training Network (RTN) ([**?**]) was put forward with the intention to promote integral field spectroscopy (IFS), or "3D" spectroscopy, and to help making it a common user technique. In order to accomplish this, one of the major tasks was identified as the need of providing standard software tools for the visualization and analysis of datacubes. These tools should be general enough to be entirely independent of the origin of data, i.e. 3D instrument. Previously, a heterogenous collection of instrument-specific data formats and software tools (e.g. XOASIS), proprietary software packages and a lack of any standard have hampered a break-through of this powerful observing method, leaving it merely as an expert technique with comparatively limited scientific impact.

Recognizing the importance of this problem, a work plan was devised to start creating a package of tools for the analysis and visualization of IFS data. Entitled *3D Visualization*, Task 2.2 of this work plan foresees the development of a programme, which should be capable of reading, writing, and visualizing reduced data from 3D spectrographs of any kind. We have named this tool "**E3D**". This document describes how to install and use the Euro3D visualization tool (E3D).

One of the major problems for the development of a standard visualization tool is the lack of a standard data format. Every group has developed its own *3D data format*, both for the spectral and the position information (cubes, FITS images, FITS tables, MIDAS images, etc...). In order to overcome this problem, the RTN has proposed a unified data format, the "Euro3D Data Format" ([**?**]; [**?**]). Taking into account previous experience from more than a decade of operating 3D instrumentation in the visible and the near-infrared, this data format is supposed to cover most foreseeable requirements of existing and future instruments. The Euro3D visualization tool was written specifically to make use of this data format, although it can read and write 3D data on another different formats (like datacubes).

Warning: The E3D is currently under development, and it could change within the time

1.2 Requeriments

E3D has been coded in `C` and `Tcl/Tk`. It uses the C-coded library ("LCL") as I/O library. This library was developed on propose to handle the input/output of data on the Euro3D format ([**?**]). It uses `PGPLOT` as graphical library. Therefore, as prerequisites it is needed to have installed in your computer both libraries, before to install E3D.

Warning: PGPLOT is not a GPL product. It is property of Tim Pearson and copyrighted by ****. Therefore, it cannot be distributed. However, it can be freely downloaded from the webpage http://*****

E3D comprises four main elements:

- The C-coded low-level functions, including the routines calling the Euro3D I/O library, the SHM routines, basic functions needed for plotting and analizing the data. These functions are included in different libraries in the `lib` subdirectory (like `Euro3D.o`), and can be used by another programmers to create standalone C-programs that handles with the Euro3D data.
- A number of stand-alone C-coded tools that help to handle the Euro3D format. Perhaps the most interesting routine is `any2Euro3D`, transforming single Group IFS data from any instrument into the Euro3D Format.

- The `tk_e3d` Tcl/Tk interpreter. This is a standalone programme that creates its own Tcl/Tk interpreter, adding Euro3D routines to the standard Tcl ones. These routines invoke the C-functions included in the C-libraries from Tcl, and they can be used for different proposes: e.g., load/save Euro3D format files, plot single or coadded spectra, plot monochromatic/polychromatic maps, interpolate these, save maps in FITS format. `tk_e3d` is the core of E3D. Once invoked, it runs a `Tcl/Tk/E3D` shell, where both command line and scripts can be runned. `plot_map.tcl`, included in the `scripts` subdirectory, is an example of the scripts that can be written to be run under `tk_e3d`. We will explain below the E3D routines in detail.
- `tk_e3d.tcl`, the E3D GUI. This is a `Tcl/Tk/E3D` script, that creates a graphical user interface for visualizing and analyzing 3D data. It includes certain number of procedures that makes easy to work with 3D data.

Warning: If you are intended to visualize/analyze 3D data, without programming your own scripts, then you are mainly interested in `tk_e3d.tcl`, the E3D GUI. You can skip the other chapters of this documentation.

1.3 Reference documents

[1]	*Euro3D Data Format* ??	M.Kissler ??/02/2003
[2]	*Euro3D I/O libraries v1.0a – Installation guide* A. Pécontal-Rousset, P. Ferruit and Y. Copin	issue 1.0a 31/03/2003
[3]	*Euro3D I/O libraries v1.0a – Developers guide* Y. Copin, P. Ferruit and A. Pécontal-Rousset	issue 1.0a 31/03/2003
[4]	*E3D, The Euro3D visualization tool* S.F.Sánchez	2004, AN, 325, 167
[5]	*E3D, The Euro3D visualization tool II* S.F.Sánchez, T.Becker, A.Kelz	2004, AN, 325, 171
[6]	*The Euro3D data format* M.Kissler-Patig, Y.Copig, P.Ferruit, A.Pécontal-Rousset, M.M.Roth	2004, AN, 325, 159
[7]	*The Euro3D LCL I/O Library* A.Pécontal-Rousset, Y.Copig, P.Ferruit,	2004, AN, 325, 163

Document [1] is available at `http://www.aip.de/Euro3D`.

1.4 Abbreviations and acronyms

AD	Applicable Document
AIP	Astrophysikalisches Institut Postdam
DIT	Detector Integration Time
DQ	Data Quality
ESO	European Southern Observatory
E3D	Euro3D Visualization Tool
FITS	Flexible Image Transport System
FWHM	Full Width at Half Maximum
HDU	Header and Data Unit
IFS	Integral Field Spectroscopy
IFU	Integral Field Unit
LCL	Lyon C-Library

MPE	Max-Planck Institut für extraterrestrische Physik
NDIT	Number of DIT
NIR	Near Infra-Red
NOST	Nasa / Science Office of Standards and Technology
PA	Position Angle
PSF	Point Spread Function
RD	Reference Document
RON	Read-Out Noise
RMS	Root Mean Square
RSS	Row Stacked Spectra
SPAXEL	SPAtial piXture ELement
S/N	Signal-to-Noise ratio
TBD	To Be Determined
WCS	World Coordinate System

2 Installation

2.1 Presentation

In this section we describe how to install the E3D. Prior to install the E3D, you will need to install the Euro3D I/O libraries (E3D_io_LCL-1.0a, or latter distributions). The installation procedure is detailed in the `Installation Guide` of the Euro3D I/O libraries [2]. You need also to install PGPLOT, including the CPS,PS,GIF,VGIF and TK drivers. PGPLOT can be downloaded from ???

> **Warning:** You need to have properly defined the E3D_io_LCL and PGPLOT environment variables.
> E.g., on a bash SHELL system:
> export PGPLOT_DIR="/usr/local/src/pgplot";
> export IFU_PATH="/usr/local/src/";
> export IFU_DEFAULT_FMT="Euro3D";

2.2 Installing E3D step by step

In the following, we describe, step by step, the procedure to follow to install E3D.

1. Download the current version of the E3D distribution (e.g.,`v3d-1.0d.tar.gz`) from the Euro3D webpage [1], and copy it into the Euro3D directory (given by the environment variable `$IFU_PATH`).
2. Using the command `tar xvfz v3d-1.0d.tar.gz` , extract the archive file `v3d-1.0d.tar.gz` under `$IFU_PATH`. It will create the directory `v3d-1.0d` (or a corresponding one, depending of the distribution).
3. Change to the created directory (`cd v3d-1.0d`), and execute the configure and make scripts.
   (a) `./configure`
   (b) `make`

   > **Warning:** The make can fail due mainly to a different definition of certain libraries. We have test the procedure under the two main Linux distributions, RedHat (7.1 and 8.0) and Suse (8.2). They differ in the definition of TCL/TK libraries. You should edit the file `add_def/makedefs.local` and select the proper definition of `E3D_LIBS` for your distribution.

   > **Warning:** The installation on a Sun/Solaris requires to edit the file `add_def/makedefs.local`, and uncomment the proper definition of `E3D_LIBS`. Before to run the `./configure` script you should copy the file `makefile.in` under the directory `scripts` to the directory `pkg/v3d/source`.

4. Goes to the scripts directory (`cd scripts`) and execute the install script (`./install.pl`). This script will edit the `tk_e3d.tcl` script to be compatible with your directory tree.
5. Add the directory `user/bin` under the E3D tree to your PATH.
6. Test the E3D executing the script `tk_e3d.tcl` , and loading the file `test.fits` under the `scripts` directory.

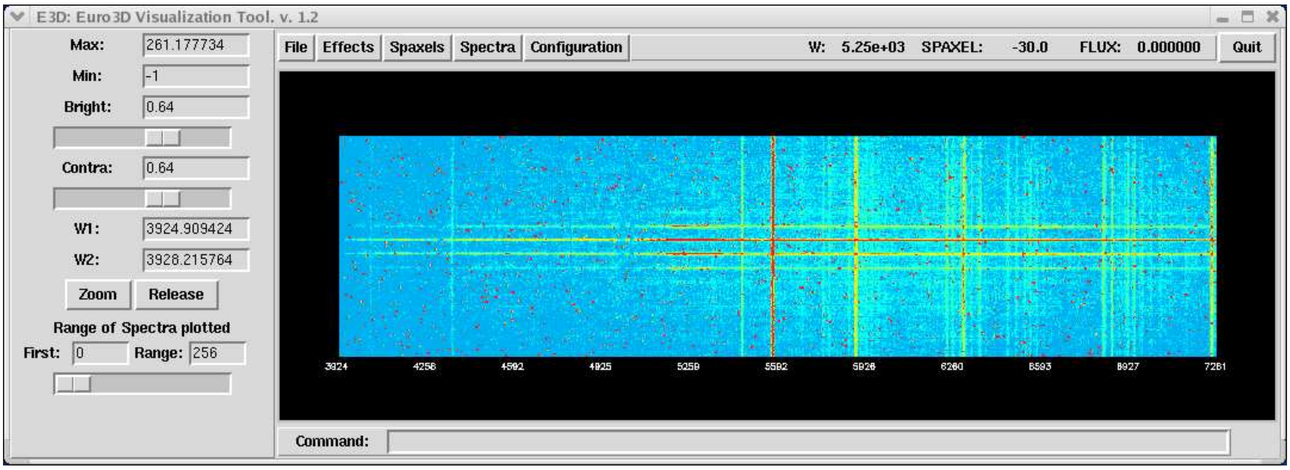

Figure 1: Snapshoot of the Main Window of the E3D GUI.

3 The E3D GUI: tk_e3d.tcl

In this section we describe the main properties of the E3D GUI, the TCL/TK script `tk_e3d.tcl`. If you have followed properly the previous sections you should be able to execute it by just typing `tk_e3d.tcl` . This script will run the `tk_e3d` TCL/TK interpreter, and open a GUI. There are three different windows in E3D: The main windows, the spaxel inspector and the spectral inspector.

Warning: *spaxel* has been defined within the Euro3D network as the spatial element of a 3D spectra. It means a fiber, or a lenslet or portion of a slit-slice, depending on the intrument. This concept has been introduced to unify the different possibilities for spatial element.

3.1 The main window

Figure 3 shows a snapshot of the main window. It contains an imagearea, an information panel and a toolbar containing different menus. The imagearea of this window is used to plot the row staked spectra (`RSS`) representation of the data, an spectral image built including in each row the spectrum correponding to each spaxel. By default it is shown the `RSS` of the first 256 spaxels on the file, at maximum. The range of plotted spaxels can be changed using the prompts on the left information panel (see the figure). The panel on the left size of the GUI includes also information and entries for the current max/min values plotted, bright and contrast selection bars and the current selected wavelength range.

It is possible to select both monochromatic or polichromatic cuts or select a range of spaxels on this imagearea, once loaded a file. The default selection is the spectral selection, allowing to select a wavelength cut by clicking with the left mouse in one (monochromatic) or two (polichromatic) spectral positions on the `RSS`. Once selected a spectral region, it is possible to (a) show a zoomed version of the `RSS` of this wavelength range (by clicking in the `ZOOM` button), or/and (b) create a spatial map, by an average of the flux over the selected wavelength range per spaxel. This map is shown in the spaxel inspector by clicking the right mouse-button (or clicking on the `Plot` button on the `Spaxel` menu). Keeping clicked the central mouse-button and moving through the image we send different cuts to the spaxel inspector, of a spectral pixel width of `delta_x`, and initial wavelength corresponding to the position of the mouse. This variable has been arbitrary defined to 5 spectral pixel, to change it type `set delta_x NN` in the command prompt.

[1]the distribution is actually restricted to the Euro3D network members, please ask for a password to Dr.M.M.Roth

The type of selection can be changed on the menu (see below) from the spectral to the spaxel selection. This selection allows to select a reduce number of spaxels by clicking one (single selection) or twice (multiple selection) on the `RSS` image with the left mouse-button. The `RSS` of the selected spaxels is shown in the spectral inspector when clicking the right mouse-button (or clicking on the `Plot` button on the `Spectra` menu). Keeping clicked the central mouse-button and moving througt the image we send the spectrum of the spaxel corresponding to the current position of the mouse to the spectral inspector.

A `COMMAND` prompt has been included in the window to run `tcl` and/or `tcl/E3D` routines and scripts, by typing in the prompt or in a file that can be run using the `source filename` routine.

The menus included in the toolbar are:

1. `File` Menu. It comprises the different I/O options of data files in E3D. The `Load` and a `Save` buttons allows you to load and save files in the Euro3D format. It is possible to import/export files in another different formats, like data cubes (`Import/Export CUBE`) and row staked spectra fits files (`Import/Export RSS`). For the datacubes, the spaxel shape is assumed to be an square with a size difined by the `tcl` variable *dpix*. For the RSS format E3D expects two different files, a fits file 2D image, with a single spectrum per row, and an ascii file for the position table (the definion of a position table on E3D will be explain below, on Section ??). A collection of E3D position tables for different instruments has been included in the subdirectory `data`, in the E3D directory tree.

   Options to import/export a slit spectra (assumed an aperture of `dpix`) and to export a wavenlength cut of the *RSS* have also been included.

2. `Spaxels` Menu to handle the SPAXELS inspector. It allows to open this inspector (`Open`), change the kind of mouse selection in the main window (`Select`) and to plot the map of the current spectral selection in the spaxels inspector (`Plot`).

3. `Spectra` Menu to handle the SPECTRA inspector. It allows to open this inspector (`Open`), change the kind of mouse selection in the main window (`Select`), and to plot the `RSS` of the selected spaxels in the spectral inspector (`Plot`).

4. `Configuration` This menu includes different submenus to change the configuration of the program:

   (a) `Colormap` Menu to change the color palette, among the different included on E3D.

   (b) `Scale` Menu to change the image scaling, among the different included on E3D (Default, Logarithm, Square, Root-Square and Linear).

   (c) `Spectral Representation` Menu to change the representation of the spectra on the spectral inspector. It is possible to plot the spectra as a `RSS`, an average spectrum plot, a plot of the different spectra as single spectrum plots and a combination of the avobe methods.

   (d) `Interpolation` Menu to configure the interpolation options of E3D. The interpolation has been included on E3D to build 2D images of the maps on the spaxel inspector from the spaxel representation and/or generate datacubes from the Euro3D formatted data. Five different interpolation routines has been included: Cubical Spiline (CSA, default), Linear Delaunay, Natural Neighbour (NN), kth Inverse Weight Nearest Neighbour (NN IDW), and 3th Linear Weight Nearest Neighbour (NNL). Some of these routines need of a GRID parameter and all of them need an output pixel size (in arcsecs) of the interpolated map. The GRID parameter has different meanings for the different routines. While for the CSA is not necesary, for the Linear Delaunay and the NN it means resolution of the triangulization (values $\sim 10^{-6}$ works ok), and for the Nearest Neightbours means the number of neighbours to be considered. Only the three first routines are flux conservative.

(e) `Spaxel Selection` Menu to configure the spaxel selection in the spaxel inspector. This selection can be spaxel-to-spaxel (single) or in a circular area of a certain radius (multiple). The selection of spaxels in the spaxel inspector will be explained below.

5. `DAR Tools` This menu includes different options for correcting the Differental Atmospheric Refraction (DAR). The DAR correction is estimated based on the theorietical values, derived from the atmospheric conditions stored in the GROUP information (see Euro3D Format Documentation), by clicking on the `Th.Det.DAR`, or based on emipiral estimations. Different empirical estimations has been included, mainly all of them trace the peak of the intensity of datacube slices at different wavelengths, estimating the correction from the displacement of this peak along the wavelength. `PM Det.DAR`, `Peak Det.DAR` and `Simple Det.DAR` use this technique. The 1st trace the peak by a principal momentum analysis, the 2nd by a direct peak search on the spaxel pattern, and the 3th by an grid interpolation of the data, and a search of the peak. This three methods are useful when we have an object on the data with a clear peak, always brighter than the rest of the data. `Emp.Det.DAR` or empirical determination of DAR correction, performs a crosscorrelation of each interpolated datacube slice through different wavelengths with the previous one, looking for the peak in the crosscorrelated image, and estimating the DAR from the displacement of this peak through the wavelength.

   Once estimated the DAR correction, it is possible to correct the data by clicking on `Correct DAR`. This correction displace each datacube slice, by the estimated correction, spatially interpolating the data (using the interpolation configuration).

6. `QUIT` Button to exit from the E3D GUI.

Warning: The E3D GUI is currently under development and can have severe changes.

3.2 The spaxels inspector

The spaxel inspector allows to visualize the monochromatic/polychromatic datacube slices that has been previously selected in the main window or in the spectra inspector (see below). Figure 3.2 shows an snapshot of this window. It comprises a principal imageare, with 8 small ones, a toolbar and a information/entry panel. The selected slice of the datacube is plotted in the main imagearea, an consecutively in one of each 8 small ones. These small imageareas are buffers of the last 8 selected slices. If more than one slice has been sent to the spaxel inspector is always possible to back to replot previous selections by clicking with the right mouse on the corresponding small image area. The active buffer will be mark with a red envelop.

The spaxel inspector allows you to select different spaxels to (a) perform a basic statistic analysis of the intensity corresponding to spaxels in the currently selected slice, (b) visualize the intensity of each one in a 2D plot and (c) visualize the corresponding spectra in the spectral inspector. There are diffrernt methods to select spaxels:

- Single spaxel selection. We select a single spaxle by moving through the image while keeping clicked the central mouse-button. The spectrum corresponding to this spaxel will be automatically send to the spectral inspector.
- Multiple selection by spaxel-to-spaxel selection. This selection is included by default, and can be modified in the `Configuration` menu of the main window. It is possible to select a single spaxel by keeping clicked the left mouse-button an moving through the image. The corresponding spaxel will be added to the selection list, and marked as hashed.

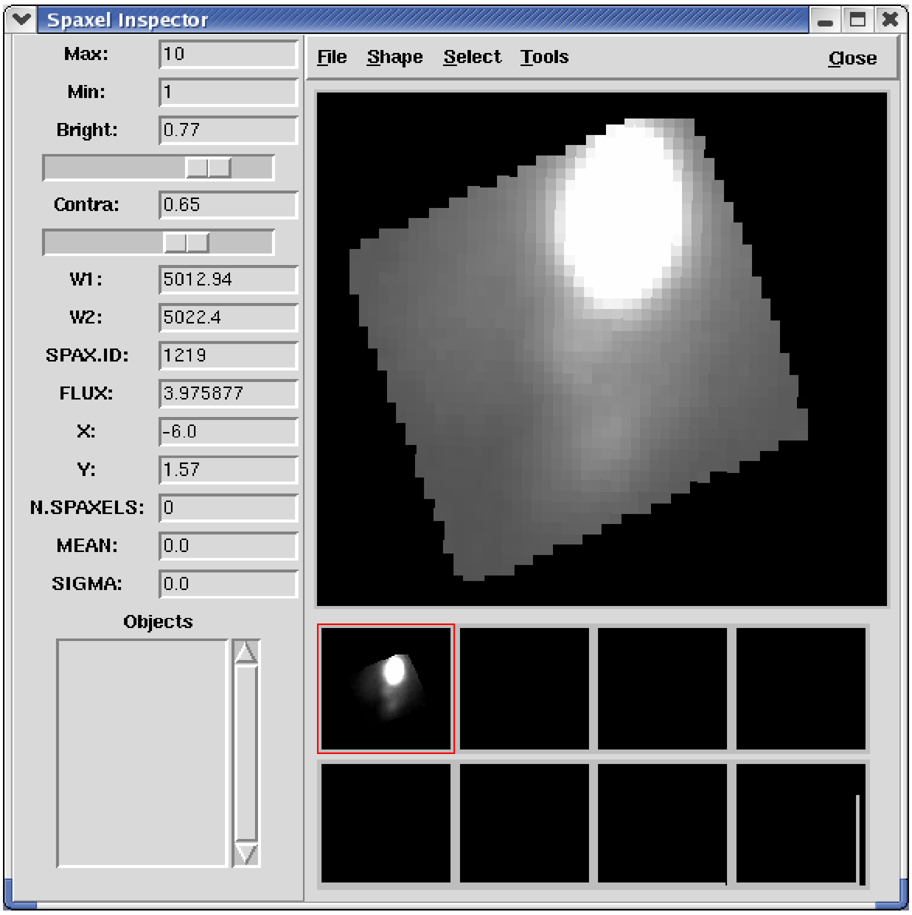

Figure 2: Snapshoot of the spaxels inspector of the E3D GUI

- Multiple selection by selecting circular areas. This selection can be selected in the `Configuration` menu of the main window, defining a radius of selection. We select the spaxels within a distance of this radius to the actual position of the mouse when moving through the image keeping clicked the left mouse-button. These spaxels will be included into the selection list and marked as hashed.

- Multiple selection by pseudo-slit definition. This mode is available by pressing the `s` key. Then, it is possible to define a pseudo-slit by clicking with the left mouse-button on a certain position of the slice, marking an starting point for the pseudo-slice, and, by a second click, marking the ending point and selecting the spaxels along the pseudo-slit. Pressing again the `s` key we return to the spaxel-to-spaxel selection mode.

A basic statistic of the flux on the selected spaxels is shown in the information panel. The spectra of the corresponding selected spaxels is seen in the spectral inspector by clicking on the right mouse-button. Pressing the `v` key we see the flux profile of the selected spaxels in the spectral inspector. Both the spectra and the fluxes are shown in the order of selection of the spaxels [2].

The left size information panel contains similar entries than the same panel in the main window (min,max,bright,contrast and currently selected wavelength range). It constains also specific entries for the spaxel inspector: (1) the identification of the spaxel currently pointed by the mouse (`SPAX.ID`) and its flux (`FLUX`), (2) a basic statistic information of the currently selected list of spaxels.

The toolbar contains the following menus:

1. `File` with different I/O options. It allows to import a MAP (a 2D image), and export the currently selected databuce slice as a FITS and ASCII table, a 2D FITS image (once interpolated to a regular grid), and Postscript or a GIF file. It allows also to save the currently selected object (described below), as a separated Euro3D formatted file.

2. `Shape` Allows to select different representions of the spaxels:

   (a) `Fill` It plots polygons with the shape of the spaxels filled with the color correponding the the spaxel intensity in the current slice, following the pattern defined by the position table.
   (b) `Outline` Similar to `Fill`, but marking the edge of the polygons.
   (c) `Spax.Id` Similar to `Fill`, but writting the spaxel ID inside the polygons.
   (d) `Map` It plots an interpolated regular image of the slice. The characteristics of the interpolation are defined in the `Configuration` menu of the main window (decrived above).
   (e) `OverContour` Similar to `Fill`, but overplotting the counterplot of the intensity levels (for that, an interpolated version of the data was created, using the same configuration as in the previous option).
   (f) `Contour` It plots only the counterplot.
   (g) `Map+Spax Shape` Similar to `Map`, but overplotting the pattern and shape of spaxels, by plotting open polygons.

3. `Select` This menu comprises different options related with the spaxels selection:

   (a) `Clear` Clean the current list of selected spaxels.
   (b) `Plot Spectra` Plot the spectra corresponding to the selected spaxels in the spectral inspector. This can be also done by clicking with the right mouse-button over the main imagearea of the spaxel inspector.

[2] In this case, the pseudo-slit selection has a better physical meaning

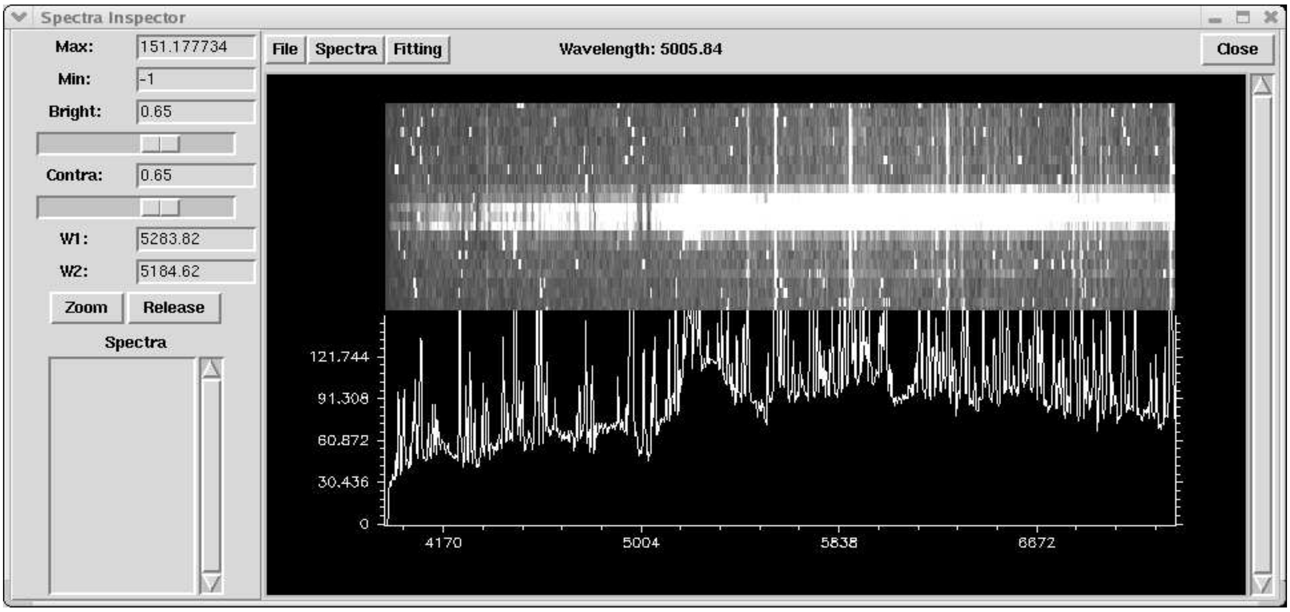

Figure 3: Snapshoot of the spectral inspector of the E3D GUI

(c) `Plot Flux Profile` Plot the fluxes of the current slice corresponding to the selected spaxels in the spectral inspector as a 2D plot. It performs a simple fitting to a single gaussian plus a continuum. This option is convinient to determine the seeing of an observation, the extension of an object or the distance between two different peaks.

(d) `Create object` It saves the currently selected list of spaxels into a buffer. We have defined as `object` to this selected list of spaxels. E3D stored a maximun of 10 `objects`, that are listed in the left size information panel. To load a previously created object, just click on the corresponding `object` on the list. These objects can be saved as a separated Euro3D formatted file, using the `Save object` option on the `File` menu.

(e) `Reverse object` It reverses the list of selected spaxels, selecting the non-selected and de-selecting the selected. This option is useful for creating an spectra of the sky.

4. `Tools` A miscellaneus of useful tools, including:

   (a) `Blink` Allows to blink between different slices previously selected and included in the small buffers. For that, click over the `Blink` button and then select the different selections by clicking over the corresponding small frames. Once you have completed the selection click again on the `Blink` button to start the blinking. To stop it, click again on the button.

   (b) `Arithmetics` It pop-ups a menu for performing basic arithmetic between the buffered slices (add, subtract, divide and multiply). The buffers are ordered from 1 to 8, from top to botton and from left to right. In order to perform an arithmetic operation, the menu ask for the number of three slices (the two operators and the output slice) and the operation to perform.

5. `Close` Close this window.

A zooming capability has been included in the spaxel inspector. In order to zoom over a certain area press the `z`-key while over the image. The mouse pointer will transform into a rectangle which defines the zooming area. Pressing again `z`, E3D will create a new map, with similar characteristics than the currently defined, but zoomed over the spatial region defined by the rectangle. E3D keeps a record of the zoomed area for any following created map. To restore the default ploting size press the `u`-key.

3.3 The spectral inspector

The spectral inspector allows to visualize and analyze spectra. It comprises an imagearea, an information panel (left) and a toolbar. The spectra plotted in the image area are sent to the spectral inspector by a direct selection of one or more spaxels in the `main window` or by an spaxels selection in the `spaxels inspector`. These spectra could be plotted with different representations, selected in the `Configuration` menu on the `main window` (see above). Once plotted, it is possible to select a certain wavelength range or move throw the wavelengths in a similar way that in the `main window`, when the spectral selection is selected (i.e., by clicking with the left mouse-button, once or twice, selecting a range, an sending it to the spaxel inspector by clicking the right mouse-button, or, moving through the wavelength while clicking the central mouse-button).

The information panel contains the same information than the same panel for the other two windows, plus an expecific entry for buffered `spectra`, that willbe explained below.

The toolbar contains the following options:

1. `File` It contains different I/O options. It allows to save the currently selected spectra as a FITS spectral image, and the average spectrum as a FITS and ASCII table. It is also possible to save the plotted image as a Postscript or a GIF file.

2. `Spectra` It allow to handle with the average spectrum, created from the average of the selected spectra. E3D allows to save 10 different average spectra in a buffer. Spectra that are listed in the information pannel on the left size of the spectral inspector, when created. There different option in this menu:

   (a) `Save Spectrum` Clicking this button E3D saves the currently average spectrum in the next available buffer. It is possible to re-plot this spectrum by clicking in the list included in the left information panel.

   (b) `Single/Multiple Plot` Allows to select a single spectrum from the list or multiple spectra. With this second option the different spectra will be overplotted in the imagearea, with different colors each one. It can be used to visually compare different spectra.

   (c) `Clear Multiple Selection` Clean the list of selected spectrum, when the `Multiple Plot` option is selected.

   (d) `Artithmetic` It allows to perform basic arithmetic operations between spectra, like `Add`, `Subtract`, `Divide` and `Multiply`. It allows also to perform a subtraction of a certain spectrum over all the datacube, by clicking on the `Sky Subtraction` button, and selecting a spectrum from the list.

3. `Fitting` This menu include simple fitting routines for a quick analysis of the emission lines in the spectra:

   (a) `Single Line Fitting` It performs fitting to a model that comprises a single gaussian function plus continuum, over the last selected average spectrum. This simple procedure allows to measure the line flux, the FWHM and the continuum level. Figure 3a shows a snapshoot of the output of this routine.

   (b) `Full Automatic Kinematic Analysis` It performs a Kinematic analysis by a single line fitting over all the spectra in the cube. It determines the emission line flux distribution, the velocity map and dispersion, and the continuum distribution. Both 4 output maps

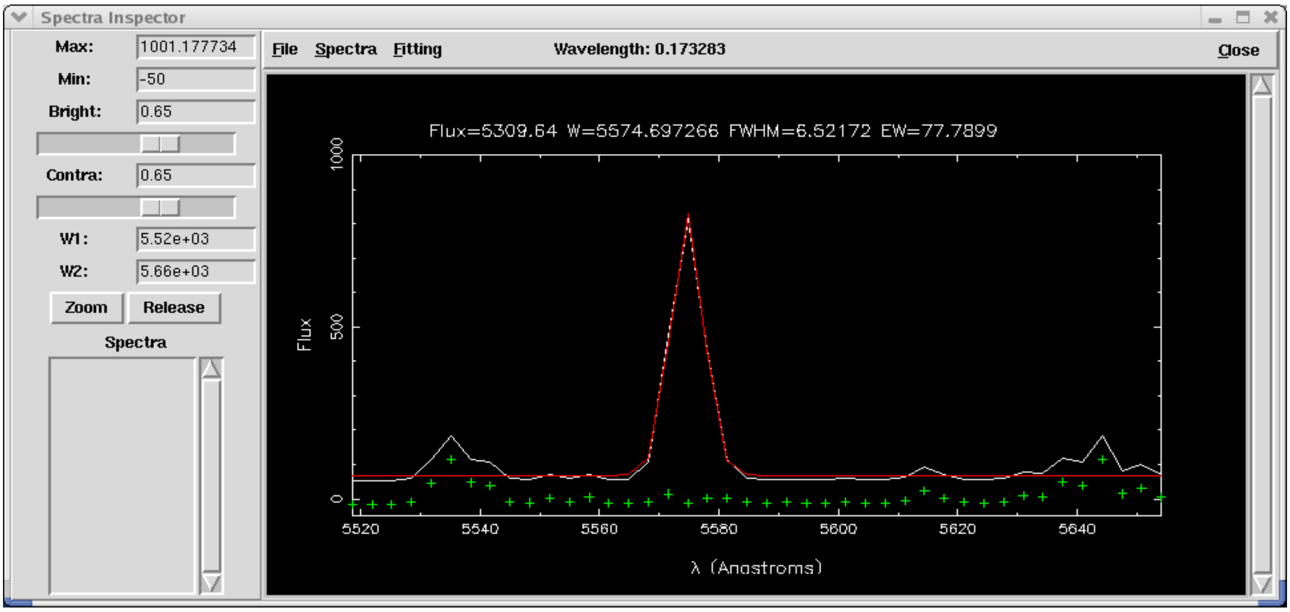

Figure 4: Snapshoot of the output of the Single Line Fitting on the spectral inspector

are sent to the `spaxels inspector` in this order. This simple procedure allows to get a rough estimation of the kinematic structure of the data.

4. `Close` Close this window.

4 The E3D Tcl/TK routines

4.1 Presentation

As we explained before, E3D includes a Tcl/Tk interpreter (`tk_e3d`), that incorporates the E3D routines to Tcl/Tk. The interpreter can be invoked to enter in a E3D shell, where the routines can be run in an interactive way. A complete scriptable enviroment is defined, for visualizing and analyzing 3D spectra. E.g., the E3D GUI is a Tcl/Tk/E3D script (`tk_e3d.tcl`) that uses the new defined routines. It also defines its own procedures, that can be invoked from the command prompt on E3D (see previous section).

We describe here the E3D routines incorporated to Tcl/Tk, there use, and we explain briefly how to built new scripts using them.

4.2 The E3D Tcl/TK routines

The E3D tcl/tk interpreter creates a new tcl/tk object, the Euro3D enviroment, that can be created by the routine `create_env`. This routine requires two entries, the name of the enviroment and the associated plotting device. E.g.,
`create_env euro3d /XS`
The available devices depend on the PGPLOT installation, although, as indicated in the Installation Section, E3D requires at least the following ones:/XTERM,/XWIN,/XSERVER(/XS),/PS,/CPS,/GIF,/XTK.

E3D plotting and analysing routines works with the following elements: 2D spectra, average spectrum, slice and object. The last selected element of these four types is stored in memory, and therefore any analysing/fitting routine would refer to this "current in use" element. E3D incorporates two different kind of buffers, to stored previously used elements. Slices are automatically saved in a circular buffer with 8 slots. Average spectra and objects are interactively stored in a buffer with 10 slots. It is always possible to load a stored element in the "current in use" slot, in order to plot it again or analyze it. Therefore, E3D incorporates routines to load and save the currently in use element (slice or spectrum) into one of the buffer slots.

The /XTK device cannot be directly invoked. It requires to call the `pgplot` routine, that creates a pgplot imagearea, and then call the `device` command over it. E.g.,

```
#!/usr/local/src/v3d/user/bin/tk_e3d
frame .frame -width 30 -height 30
pgplot .frame.pgplot -share true -width 300 -height 300 -mincolors 256 -maxcolors 12000
-bg black -fg white
pack .frame.pgplot -side left -fill both -expand true
pack .frame -side left -fill both -expand true
create_env euro3d [.frame.pgplot device]
```

4.3 Anotated list of commands over the E3D enviroment

- `ask_e3d_info`

  It returns the basic parameters of the 3D spectra: the number of spectral pixel (maximum), the number of spaxels, the starting wavelength, the final wavelength, the wavelength increment per spectral pixel, the data minumum and maximun values, the first and the last spectra actually defined to be plotted (these last values are, by default 0 and 256).

- `bc` ,USE: `bc x`

  Deprecated

- `clean_device dev_id`

  Indicates not to overplot on the device with identification `dev_id` the next time to plot on eat, but clear it before plotting.

- `clean_server`

  Clean the SHM server.

- `close_device dev_id`

  Close the device indicated by the index `dev_id`

- `correct_dar`

  Correct the data for DAR, differential atmospheric refraction. A DAR correction should be defined.

- `create_map i1 i2 [dpix] [square=0/1]`

Creates an interpolated map from the datacube slice defined between the spectral indeces `i1` and `i2`. The size of the interpolated pixel, `dpix` and if the final map is an square or nor not `0/1` are optional values. See also `set_grid` .

- `create_obj`

  Creates an `object`, i.e., a memory buffer of the currently selected list of spaxels. It returns the index of the saved `object`

- `create_slice i1 i2 slice_index flag`

  Creates a datacube slice, between the spectral indeces `i1` and `i2`, and tries to allocated it in the slice index `slice_index`. The `flag` indicates the kind of slice to be created, ie, if `flag`>`90` it save the slice from the currently in use, in other case, it creates a new slice. See also `plot_spaxels`.

- `create_spec`

  Save the currently used average spectrum in a memory slot. It returns the slot index.

- `div_spec i1 i2`

  Deprecated. Divide the spectrum stored in the memory slot `i1` by the spectrum in the memory slot `i2`. It returns the memory slot of the result. See `specarith` .

- `draw_raw min max device clean palette bright contra sign first_spec last_spec`

  Draw the spectra contained in the 3D file on its RSS format. It requires as parameters the `min` and `max` values to plot, the *device* index to plot it, the `palette` [3], brightness (`bright`) and contrast `contra`, an inversion parameter (`sign`) and the first and last spectra to be shown (`first_spec` and `last_spec`).

- `empirical_dar index delta_index`

  Determines the empirical DAR correction, based on the comparison of the centroid position of the crosscorrelated databuce slices. It requires the spectral `index` of the slice used as reference, and the spectral indeces width of the datacube slices, `delta_index`.

- `encircled_spax X Y R n`

  It returns the spaxel ID of the `n`th spaxel that lies in a distance smaller than `R` arcsec from the position defined by the coordinates `X` and `Y`.

- `export_cube filename`

  Export the 3D data to datacube. It regularizes the data into a grid defined by `set_grid` (see below), creating, for each monochromatic slice, an interpolated 2D image.

- `export_rss fits_file pt_file`

  Export the 3D data to a RSS fits file (`fits_file`), and a ASCII position table (`pt_file`). E3D uses its own definition of how to stored a position table (see ???).

- `feed_slice n_feed list`

  Save the slice `n_feed` with the data included in the Tcl `list`. This routine can be used to load into the visuzalizatio tool slices created by external tools, like velocity maps created by line fitting tools, without affecting the 3D data currently in use.

[3] currently available: grey, rainbow, heat, iraf, aips

- `fit_single_line min max device palette bright contra sign Ig D_Ig fit Wo D_Wo fit sigma D_sigma fit Continuum D_Continuum fit CHISQ_LIM plot`

  Fits a single emission line to the currently in use average spectrum. The model includes a gaussian plus a constant value for the nearby continuum. It requires the same parameter than `draw_raw` for the resulting plot, plus the initial guess values for the fitting, including, for each parameter (Intensity peak, central wavelength, sigma of the gaussian function and the continuum value), the value(e.g., `Ig`), the range of valid values (e.g.,`D_Ig`), and a flag indicating if to fit this values or not (1 for fitting). It also requires the desired reduces χ^2, and a flag indicating of to plot or not the results (0 for not plotting).

- `flux_spax spax_index`

  Returns the flux of the spaxel defined by `spax_index` in the currently in use slice.

- `guess_single_line`

  Guess the initial parameters to use `fit_single_line`. It returns a tcl list including `Ig, Wo, sigma and Continuum`.

- `import_cube fitsfile`

  Imports a 3D datacube, `fitsfile`, into E3D.

- `import_map fitsfile`

  Imports a 2D image, `fitsfile`, into E3D. The image is considered as a single spectral index 3D cube.

- `import_rss rss_fitsfile positiontable`

  Imports a 3D data in the RSS format into E3D. It requires a 2D spectral image, the RSS image (`rss_fitsfile`) and an ASCII file containing the E3D position table (`positiontable`).

- `import_spectra fitsfile`

  Imports a 2D spectral image into E3D. It considers the data to be a slit spectra, with an slit width of `dpix`. See `set_grid`.

- `intensity i j`

  Returns the intensity of 3D spectra of the jth spaxel at the spectral index `i`.

- `load_file filename`

  Loads the Euro3D formatted file `filename` into E3D.

- `load_n_spectra_sel`

  Returns the index of the currently loaded memory buffer spectrum.

- `load_obj n_obj`

  Loads the object number `n_obj` into the currently selected list of spaxels.

- `load_spec n_spec`

  Loads the spectrum in the buffer slot number `n_spec` into the current use spectrum.

- `load_sepectra_sel n`

  Returns the spaxel identification of the ith element of the currently in use object (i.e., list of selected spaxels).

- `mark_range dev_id colorindex lwidth lstyle text x1 y1 x2 y2`

  Mark the device `dev_id` with the `text`, in the color defined by the `colorindex` and a two headed arrow, with a line width `lwidth` and a line style `lstyle`, between the positions (x1,y2) and (x2,y2).

- `message dev_id message`

  Writtes the text `message` in the center of the device `dev_id`

- `nearest_spax X Y`

  Returns the spaxel identification of the nearest spaxel to the position (X,Y).

- `open_device DEVICE`

  Opens a new PGPLOT DEVICE, from the list if allowed devices included in the current PGPLOT installation. It returns the device identification for the opened device.

- `peak_dar index delta_index`

  Determines the empirical DAR correction, based on the comparison of the position of the peak intensity of the datacube slices. It requires the spectral `index` of the slice used as reference, and the spectral indeces width of the datacube slices, `delta_index`.

- `plot_flux_spax dev_id clean`

  Plots the flux of the slice corresponding to the currently in use list of selected spaxels (or object) along the distance to the first of these spaxels. This routine is useful to visualize the flux profile with a pseudo-slit selection. It requires the id of the PGPLOT device where to plot (`dev_id`), and a flag indicating if to clean or not this device the next time that we plot on it (`clean`, 0 for not to clean).

- `plot_line dev_id colorindex lweight lstyle X_points Y_points`

  Plots a line defined by the points in the lists X_points and Y_points in the device `dev_id` with the color defined by the index `colorindex`, the weight `lweight` and the line style `lstyle`.

- `plot_map min max dev_id palette bright contra sign n_map`

  Plots the map (2D interpolated image of an slice) corresponding to the slice in the buffer slot `n_map` in the device `dev_id`, with the plotting parameters indicated: min, max, palette, bright, contra and sign. See `draw_raw`.

- `plot_scale min max dev_id bright contra sign palette units top_min top_max border`

  Plots an intensity scale on the right size of the device `dev_id`, with the indicated plotting parameters: min, max, palette, bright, contra and sign. See `draw_raw`. The plotted range will be between the values `top_min` and `top_max`. The units can be plotted on top of the scale.

- `plot_spaxels min max dev_id i1 i2 palette bright contra sign save fill type [new_img] [back_color] [n_contours]`

  Plots the 2D map created from the average of intensities between the spectral indeces `i1` and `i2` from the 3D cube in the device `dev_id`, with the indicated plotting parameters: min, max, palette, bright, contra and sign. See `draw_raw`.

  E3D includes different representations of the 2D maps, that can be selected by selecting different `type` values: (0) for the spaxels representation, keeping the geometry of each element at each group, (1) spaxel representation, overplotting the outpart of each spaxel, (3) spaxel representation, overplotting the spaxel Id, (4) interpolated reconstructed map (using the parameters defined by `set_grid`, see below), (-1) spaxel representation with an overlapped contour-plot and (-2) contour-plot.

Other options are included: `new_img` can be 0, for not plotting a flux scale and a frame border (e.g., when plotting in the GUI), or 2, for plotting a flux scale and a frame border (e.g., for generating postscripts); `back_color` defines the PGPLOT color index of the background color; `n_contours` defines the number of contours to be plotted by the contour-plot; `fill` has no use actually.

- `plot_spectra min max dev_id clean palette bright contra sign i_spax`

  Plots the spectra corresponding to the spaxel indeces included in the `i_spax` list in the device `dev_id`, with the indicated plotting parameters: min, max, palette, bright, contra and sign (see `draw_raw`).

- `plot_spectra_mem min max device 1 palette bright contra sign [clean]`

  Plots the actual average spectrum storaged in memory in the device `dev_id`, with the indicated plotting parameters: min, max, palette, bright, contra and sign (see `draw_raw`). If `clean` is 0, it wipes out the plotting area.

- `pm_dar index delta_index`

  Determines the empirical DAR correction, based on the comparison of the centroid position of consecutive slices, computer by a principal momentum analysis. It requires the spectral `index` of the slice used as reference, and the spectral indeces width of the datacube slices, `delta_index`.

- `read_cont_map`

  Deprecated.

- `read_vel_map`

  Deprecated.

- `recolor_image palette dev_id`

  Defines a new palette for the device `dev_id`.

- `release_raw dev_id`

  Redefines the spectral range to be plotted to the full range, set the number of spaxels to plotted in the raw-stacked spectra image to the default value of 256 and clean the device `dev_id`.

- `reverse_obj n_obj`

  Reverses the object selection of the object `n_obj`, i.e., select all the spaxels not included in the object selecting as the new object `n_obj`.

- `save_cut filename`

  Saves a raw-stacked spectra image with the currently selected spectral range in the fitsfile `filename`.

- `save_file filename`

  Saves all the data in memory as an Euro3D formatted file `filename`.

- `save_map n_map filename`

  Saves the maps number `n_map` in the fitsfile `filename`. (Note: `n_map` goes from 0 to 7).

- `save_obj_file n_obj filename`

  Saves the object selected `n_obj` as an Euro3D formatted file `filename`.

- `save_PT filename`

  Saves the position table of the Euro3D data on memory to an ascii file `filename`. The position table is storaged in the E3D-defined format, including the information of the shape of the spaxels (SHAPE SIZE1 SIZE2 ANGLE), in a line per group, followed by the position table information (ID X Y GROUP), with a line per spaxel.

  E.g.
  ```
  R 0.5 0.5 1
  H 0.5 0.5 2
  1 -1.0 1.0 1
  2 -1.0 0.0 1
  5 1.0 1.0 2
  6 1.0 0.0 2
  ```

  This position table describes an IFU with two groups, with two different geometries (R=Rectangle,H=Hexagon), and sizes of 0.5 (SIZE1 and SIZE2). Each group contain 2 spaxels each one.

- `save_raw filename`

  Saves the full wavelength range raw-stacked spectra as a fitsfile `filename`. (NOTE: for saving the currently selected wavelength range `save_cut` should be used.

- `save_slice n_slice filename units outputtype`

  Saves the slice number `n_slice` in the file `filename`. The units of the data can be indicated. The type of the output file can be selected between fits talble (outputtype=0) and an ascii table (outputtype=1).

- `save_spec filename i_spax`

  Saves the spectra corresponding to the list of spaxels `i_spax` in the fitsfile `filename`, as a row-stacked spectra.

- `save_spec_table filename units outtype i_spax`

  Saves the average spectrum from the spectra corresponding to the list of spaxels `i_spax` in the `filename`, as a fits (outtype=0) or an ascii (outtype=1) table.

- `save_spectra_sel i_spax`

  Stores the list of selected spaxels `i_spax` in the memory, as the currently selected spaxels.

- `set_first_last first last`

  Defines the range of spectra to be plotted in the raw-stacked spectra representation of the data (see `draw_raw`). The range is defined between the spaxel number `first` and the spaxel number `last`.

- `set_flag_spec flag_spec`

  Defines the kind of statistics used to define the average spectrum from a list of selected spectra: flag_spec=0 for mean, flag_spec=1 for 2σ cleaping.

- `set_grid grid_func grid_opt pixel_size`

  Defines the algorithm to be used for the maps interpolation, from the list included in E3D (`grid_func`): (1) Cubical Spiline (CSA, default), (2) Linear Delaunay, (3) Natural Neighbour

(NN), (4) kth Inverse Weight Nearest Neighbour (NN IDW), and (5) 3th Linear Weight Nearest Neighbour (NNL). The `grid_opt` is a parameter needed for each function, that has no use for the 1, 2 and 5. For 3 defines the acurracy of the slope, and it should be a very low number, and for 4 defines the last neighbour to take into acount.

The pixel scale of the interpolated map is defined by `pixel_size`, in the units of the position table (e.g., arcsec).

- `set_type_spec_rep type`

  Defines the `type` of spectral representation for the different routines that plots spectra: `plot_spectra` and `plot_spectra_mem` . The different representations are: (0) RSS image plus 2D plot of the average spectrum, (1) RSS image, (2) 2D plot of all the spectra together, (3) RSS image plus 2D plot of all the spectra together and (4) 2D plot of the average spectrum.

- `simple_dar index delta_index`

  Determines the empirical DAR correction, based on the comparison of the peak position of the interpolated slices (maps). It requires the spectral `index` of the slice used as reference, and the spectral indeces width of the datacube slices, `delta_index`.

- `skysub n_spec`

  Subtracts to all the spectra in the cube the stored spectrum number `n_spec`. (Note: This routine is useful to perform an rough sky subtraction).

- `slicearith n1 operator n2 n3`

  Performs the arithmetic operation defines by the `operator` (+,-,*,/) between the slices stored in the buffer `n1` and `n2`, saving the result in the buffer `n3`.

- `specarith n1 operator n2`

  Performs the arithmetic operation defines by the `operator` (+,-,*,/) between the spectra stored in the buffer `n1` and `n2`, saving the result in the currently in use buffer.

- `start_server`

  Starts the Shared Memory Server (experimental).

- `stats_spax`

  Performs basic statistics over the fluxes of the currently selected spaxels (previously stored using `save_spectra_sel i_spax`).

- `stop_server`

  Stops the Shared Memory Server (experimental).

- `sub_spec n1 n2`

  Deprecated. Subtract the spectrum in the buffer `n2` to the spectrum in the buffer `n2`, saving the result in the currently in use buffer.

- `tcl_pgptxt x y angle font text size color width`

  Plots the `text` in the position `x` and `y`, rotated an angle `angle` with a font type `font`, and the size `size`. The color of the text is defined by the color index `color`, and the width of the line by `width`

- `theoretical_dar`

  Determines the theoretical DAR correction, based on the formulae presented by Filippenko (1995), and the igroup information stored in the Euro3D format.

- `zoom_raw pix1 pix2 dev_id`

  Select the wavelength range between the spectral pixels `pix1` and `pix2`, for cutting the row-stacked spectra. It also wipe out the device `dev_id`.

- `zoom_spax dev_id x_min x_max y_min y_max bordersize [aspect]`

  Defines the zooming area for map plotting in the device `dev_id`, between the values x_min x_max y_min y_max, with a certain bordersize. The `aspect` flag defines whether to mantain the real aspect ratio of the spaxels (1) or not (0).

- `unzoom_spax dev_id`

  Redefines the zooming area for map plotting in the device `dev_id` to the default values.

4.4 Examples of E3D Tcl/Tk scripts

We list a number of simple E3D/tcl scripts as examples of the use of the scripting capabilities of E3D. All of them have to be executed using `tk_e3d`, the E3D interpreter.

4.4.1 Extract a single spectra of an Euro3D cube

The following script extracts a single spectrum, corresponding to the spaxel number 100, from the Euro3D file `e3d_file.fits` and save it as an ASCII table, in the file `spectrum.txt`.

```
create_env euro3d /null
euro3d load_file e3d_file.fits
set selected_temp
lappend selected_temp 100
euro3d save_spec_table spectrum.txt Flux 1 $selected_temp
exit
```

4.4.2 Extract a single slice of an Euro3D cube

The following scripts extracts a single slice, corresponding to the spectral pixel number 1, from the Euro3D file `e3d_file.fits` and save it as an ASCII table, in the file `slice.txt`.

```
create_env euro3d /null
euro3d load_file e3d_file.fits
euro3d create_slice 0 1 1 0
euro3d save_slice 1 slice.txt Flux 1
close
exit
```

4.4.3 Extract a single slice of an Euro3D cube

```
create_env euro3d "/null"
euro3d load_file e3d_file.fits
set id_ps [euro3d open_device "file.ps/CPS"]
euro3d plot_spaxels 1.0 100.0 $id_ps 100 300 heat 0.6 0.7 1 1 1 0 2
euro3d close_device $id_ps
```

```
06 exit
```

5 Connection E3D with external packages

5.1 Presentation

In this section we describe how to interact with E3D using external packages. There are two ways to interact with E3D, once based on FILES, using the E3D tk/tcl routines, and another using SHM, through the E3D Share Memory Server, and low-level C-routines.

For the majority of the cases the use of FILES is highly recommended, and more easy to handle. The SHM is operative, but it requires larger knowlodges on C-programming and the hardware architecture of your computer. Even more, it can interact with another programs using SHM, with unpredictible consequences. We recommend to use only for high-experience coders.

5.2 Interaction through FILES

E3D can understand direct commands calls (see REF.), and scripting files (see REF.). Using this method it is easy to interact with E3D, once knowing the nomenclature of E3D tcl/tk routines (REF).

We have added a simple system to send orders to E3D through the file `~/.E3D/input.e3d`. It will execute any secuence of tcl/tk (+E3D) commands included in this file, whatever extend. As a simple test, execute the following command line with the E3D open:

```
echo "puts test" > ~/.E3D/input.e3d
```

you should have an output on the xterm where you have opened E3D on the form:

```
test
```

Once you sent commands to E3D you can recive answers from it, just in form of ASCII and FITs files. By convention we have defined three different output files, for different kind of outputs:

- `~/.E3D/output.e3d` ASCII file. It is intended for small outputs, both combining numerical and strings information.
- `~/.E3D/output_e3d.txt` ASCII file. It is intended for large outputs of numerical data, like position tables, or slices.
- `~/.E3D/output_e3d.fits` FITs file. It is intended for large outputs of numerical data, like spectra or maps.

It is recomended to delete the output file once you have readed it, not to allow possible confusions.

An example of `~/.E3D/input.e3d` file should looks like:

```
cd DIRECTORY

load_e3d E3D_FILENAME

euro3d draw_raw 0 300 1 1 heat 0.7 0.7 1

set data [euro3d ask_e3d_info]

set fid [open ~/.E3D/output.e3d RDWR 0777]

puts $fid $data
```

```
close $fid
```

This script will load an E3D file (`E3D_FILENAME`) into the E3D, plot the row staked spectra, and read the common information of the file, and stores it in the standard outputfile (`~/.E3D/output.e3d`).

We have included two different sets of examples, interacting with E3D through IDL and PYTHON (by Sébastien Foucaud).

5.2.1 E3D through IDL

On the directory `scripts/idl` under the E3D architecture, you can find three files containing three IDL procedures:

`e3d_display.pro (e3d_display)`

`e3d_read_spec.pro (e3d_read_spec)`

`e3d_read_slice.pro (e3d_read_slice)`

- `e3d_display,filename,data`

  Loads an E3D file, `filename`, into the visualization tool and display the Row-staked spectra image on it. It gives you back an array containing the basic common information of the E3D file: number of pixels on each spectra, number of spaxels, starting wavelength, final wavelength, wavelength increase per pixel, number of groups, maximun and minimun value within the data.

- `e3d_read_spec,nspec,spectrum`

  Reads the spectrum `npsec` from a previous loaded E3D file, and save its result in the array `spectrum`.

- `e3d_read_slice,i1,i2,data,slice`

  Creates a broad-band SLICE, from the wavelenght index `i1` to the wavelength index `i2` and save it in the matrix `slice`. It requires the `data` from the file, output of the `e3d_display` routine, as an input parameter.

  `slice` is a 4×`NSPAXELS` matrix, that contains, in each row: the identification of the spaxel, the `x` and `y` position of the spaxel and the flux associated with this spaxel. This flux is the average of the fluxes through the spectra from the wavelength index `i1` to `i2`.

5.2.2 E3D through PYTHON

On the directory `scripts/python` under the E3D architecture, you can find a PYTHON library, `e3d.py` cotaining the following functions:

- `e3d.display("filename",min,max)`

  Loads an E3D file, `filename`, into the visualization tool and display the Row-staked spectra image on it.

- `spectrum=e3d.rdspec(number)`

  Reads the spectrum `number` from a previous loaded E3D file, and save its result in the array `spectrum`.

- `slice=e3d.rdslice(i1,i2)`

Creates a broad-band SLICE, from the wavelenght index `i1` to the wavelength index `i2` and save it in the matrix `slice`.

`slice` is a 4×`NSPAXELS` matrix, that contains, in each row: the identification of the spaxel, the `x` and `y` position of the spaxel and the flux associated with this spaxel. This flux is the average of the fluxes through the spectra from the wavelength index `i1` to `i2`.

Warning: To use the python scripts described above, some PYTHON modules have to be previously installed: `pyFITS`[a], `NumArray`[b], `Numeric`[c], and `TableIO`[d].

[a] http://stsdas.stsci.edu/pyfits
[b] http://sourceforge.net/project/showfiles.php?group_id=1369
[c] http://sourceforge.net/project/showfiles.php?group_id=1369
[d] http://php.iupui.edu/ mmiller3/python

5.3 Interaction through SHM

E3D can handle direct memory access using a Share Memory Server (SHM) implemented on its C-Core. This comunication is actually only on one-way, letting access from any external routine to the data storaged on the E3D memory. Some high-level C-routines have been coded to simplify the access to the data, all of them included in the `Euro3D.o` file. Any C-program intended to use these routines should be compiled including this file.

To allow E3D to cominicate through the SHM, you have to start the Server using the SHM buttom on the GUI or with the Tcl/TK routine `start_server` (REF??).

The SHM routines in the `Euro3D.o` file are:

- `int ask_e3d_info(E3D_file *e3d_frame, int **n_spaxels, SPAXEL ***spaxels, int **index_start, int **npts, GROUP **groups)` This function allows to access to the basic descriptors of the current Euro3D file loaded on the E3D through the SHM. This descriptors contains the E3D_file structure (`e3d_frame`), the array of number of spaxels (`*n_spaxels`), the SPAXELs matrix (`**spaxels`), the array starting wavelength index for each spectra (`*index_start`), the array of number of wavelength pixels contained in each spectra (`*npts`), and the array of GROUPs (`*groups`).

  All these descriptors have to be passed as pointers, in the form:

  `ask_e3d_info(&e3d_image,&n_spaxels,&spaxels,&index_start,&npts,&groups);`

- `int ask_raw_data(float **out_raw_data, int nb_spec, int npix, int delta_n)`

  This function allows to access to the spectra contained in the E3D file, storing them in a staked spectra array, ie, an array of `npix` ×`nb_spec`, where `npix` is the number of wavelength pixels and `nb_spec` is the number of spaxels.

  `delta_n` is the number of spectra that will be transfered in a row for each request to the SHM. The SHM has a hardware/kernel limitation that can not be avoid, so you should know how big are your spectra in terms of memory, no to send too much in a row. For a spectrum of 1024 float spectral pixels, a typical value of `delta_n = 256` would be enough.

  The staked spectra is stored in the array `*out_raw_data`. The function must be called in the form:

  `ask_raw_data(&out_raw_data,nb_spec,npix,delta_n)`

- `int ask_slice(float **out_data, int nb_spec, int index_w)`

This function allows to access to a single slice of the E3D file, with wavelength index `index_w`, storing it in a `nb_spec`-length array `*out_data`.

The function must be called in the form:

`ask_slice(&out_data,nb_spec,index_w)`

- `int ask_spectrum(float **out_spectrum, int specId, int npix, int index_start)`

  This function allows to access to a single spectrum of the E3D file, with spectral identifier `specId`, storing it in a `npix`-length array `**out_spectrum` . It is needed to provide the `index_start` of the the spectrum as input parameter.

  The function must be called in the form:

  `ask_spectrum(&out_spectrum,specId,npix,index_start)`

- `int ask_load_e3d(char *input_filename)`

  This function send a command to the visualization tool SHM to upload a new E3D file (`input_filename`).

- `int ask_delete_shm()`

  This function send a command to the visualization tool SHM server to wipe out the Share Memory.

An example of the interaction with the visualization tool through the SHM can be found in the `user/bin` directory, under the name `shm_client2`. To test the SHM server you can open the E3D, and click the SHM button. Once this done, you load a file in the standard way (REF??). The file will the uploaded both to the E3D memory and the SHM server memory. It is possible to access to the data on the file through the SHM running `shm_client2`. This program will *ask* for the data to the SHM and plot them in the same way as the `protoEuro3D` program (REF??).

The source code of this program can be found on `pkg/v3d/source/shm_client2.c` .

6 FAQ

—oOo—