Astronomical Data Analysis Software and Systems X
ASP Conference Series, Vol. 238, 2001
F. R. Harnden Jr., F. A. Primini, and H. E. Payne, eds.

The Evolution of GIPSY—or the Survival of an Image Processing System

M. G. R. Vogelaar and J. P. Terlouw

Kapteyn Astronomical Institute, Postbus 800, 9700 AV Groningen, The Netherlands

Abstract. Since its introduction in the early seventies, GIPSY has constantly evolved. We present an overview of the developments over the last few years. These include the introduction of event-driven user interaction and the addition of a set of highly interactive graphical user interface (GUI) components. The GUI has been built on top of the existing user interface with which it is completely compatible. We also present examples of applications based on these developments.

1. Introduction

GIPSY[1], the Groningen Image Processing SYstem (Allen et al. 1985; Van der Hulst et al. 1992), has its roots in the early seventies. Since then it has constantly evolved. Stimulated by close contact with its users, the exploration of many new ideas in image analysis and user interaction has been possible. GIPSY's user group is modest in size but is still growing and has extended beyond those interested in the analysis of spectral line synthesis data.

The system is organized as a set of independent application programs, 'tasks', which are controlled by the user interface program Hermes (Allen & Terlouw 1981).

2. Event-driven Tasks and Graphical User Interfaces

Originally, all tasks within GIPSY were programmed in a procedural fashion. For many tasks this is still the case. Operation is enhanced by the user interface program Hermes. One of Hermes' most important functions is maintaining the user inputs for every active task. These are stored in the form of keyword-value pairs. The user can supply input to Hermes at any time. When a task needs input, it presents the keyword to Hermes, which then returns the associated value. If no value is available, the user may be prompted.

2.1. The Event Mechanism

Event-driven tasks are fundamentally different from procedural tasks. Procedural tasks are in control of the order in which their different parts are executed

[1] `http://www.astro.rug.nl/~gipsy`

and consequently of the required order of their inputs. An event-driven task, by contrast, is programmed so that it ideally can handle any input at any moment. In this way the user, not the task, is in control.

To support the event-driven operation of tasks, some features had to be added, but the original scheme was kept intact. Hermes was modified so that it sends a message to the task whenever there is a change in its set of inputs. The task, in turn, must be prepared to receive this message. For this purpose an event-dispatching routine was written. When a task is to be notified of changes related to particular keywords, it can register one or more functions with the event-dispatcher for every such keyword. Whenever a change occurs for any of these keywords, the dispatcher will call all functions registered for that keyword. Each function can then obtain the associated value in the usual way.

2.2. The Graphical User Interface

GIPSY's graphical user interface has been implemented as a collection of 'elements', each consisting of an X Toolkit widget, such as a button, a menu, a text input field, a plot canvas, etc., and code which connects the widget to the event mechanism. Most elements are associated with a user input keyword. The widget always reflects the keyword's value, regardless of how it was set. When the widget is manipulated by the user, e.g., by typing text into an input field, the keyword will be updated. This update then causes an event that can be received and handled by the application code. Implemented in this way, there is a strict separation between form and meaning. A benefit of this is that the application programmer need not know anything about the details of the GUI, but only has to deal with a uniform, abstract, event mechanism. It also allows the separate development of application code and user interface.

2.3. Plot Windows and PostScript Driver

In earlier versions of GIPSY, the PGPLOT library (by T. J. Pearson, California Institute of Technology) was only used for vector graphics. Image display was done by GIPSY's image display server GIDS. In the current version PGPLOT's image capability has become an important utility.

GIPSY's graphical user interface contains a PGPLOT device driver supporting up to 16 independent plot windows. The color maps for these windows can be partially or completely shared, or can be separate. Because PGPLOT's procedural cursor interaction cannot be used in the event-driven environment, a special mouse interface routine is provided instead.

Associated with each window, off-screen images can be stored which can quickly be brought back to the screen. These can be used for backup purposes, e.g., when it is too time consuming to regenerate the window while the application regularly needs to overwrite parts of it. They can also be used to implement movie loops, etc.

For images with a small number of pixels, which usually have a 'blocky' appearance when displayed, interpolation between pixels has been implemented. This feature is available both in the display windows described above and in GIPSY's PostScript driver. To prevent the PostScript output from becoming excessively large, in this case the interpolation was implemented as an algorithm written in the PostScript language.

3. New Applications

3.1. Applying the New Input Facilities

The developments described above allowed us to write more user-friendly programs. The graphical interface is used to facilitate input for applications, especially those where it is important to understand the relations between variables used for fine tuning. Examples are the fitting parameters in the GIPSY tasks GAUFIT2D and XGAUFIT. The first fits the parameters of a two-dimensional Gaussian distribution to data in an image and uses a robust moments calculation to find initial estimates for a least-squares fit. The second task fits the parameters of a one-dimensional Gaussian distribution to profiles, usually used to derive a velocity field from data in an H I data cube. It also measures deviations from the Gaussian shape using higher order terms of the so called Gauss-Hermite series. (Van der Marel & Franx 1993)

Another example is program RENDER which renders 2-D and 3-D GIPSY data and displays 2-D slices at any angle. It is based on the subroutine library PGXTAL[2] by D. S. Sivia. The sky coordinate transformation task SKYTOOL is an example of matured functionality with a new interface. It is based on intensively used and tested transformation routines. One can enter a position in any of the GUI fields, either formatted or in degrees, and the program will immediately update the other fields with the corresponding coordinates.

3.2. Enhanced Task Interactivity

We also used the new techniques to create highly interactive applications. Examples are the interactive profile fitter XGAUPROF, based on the same algorithms as used in XGAUFIT, and ROTMAS, which fits the mass components contributing to a rotation curve to an 'observed' rotation curve. All components can be interactively varied to explore parameter space. ROTMAS uses the (modified) values of the variable parameters as initial estimates for a least squares fit. The composition of the curve can be either a function selected from a menu or a user defined expression. ROTMAS also includes special functions such as a Hernquist or Isotherm halo, exponential functions or expressions entered by the user.

3.3. New Procedures for Image Analysis

Inspired by the new applications, users proposed new functionality which could not be implemented before. One example is INSPECTOR. This task displays slices through a three-dimensional GIPSY data set (channel maps) and overlays radial H I velocities (i.e., 'tilted ring' circular velocities) on velocities in position-velocity diagrams extracted from the data and allows the user to adjust the tilted ring parameters interactively in order to get a better fit. A full description of how to derive rotation curves with INSPECTOR and its advantages and disadvantages compared to other methods is given in Swaters (1999). A second example is SLICEVIEW. This is a multi-purpose inspection tool for GIPSY data sets. It has an easy to use movie facility for the display of subsets, e.g., channel maps.

---

[2] `http://www.isis.rl.ac.uk/dataanalysis/dsplot/pgxtal.htm`

It also has a flexible way of creating images of slices through a 3-D data set. Slice types are a straight line, an ellipse or a cubic spline with control points positioned by the mouse.

3.4. Tools for Education

Some new GIPSY applications are also suitable for education. The task FUNPLOT is an example. It is a utility to explore mathematical expressions. The user enters an expression containing functions and variables. For each variable a slider is created which enables the user to interactively change the value of that variable. The function, as well as its derivative and Fourier transform, can also be plotted. The interactive change of the variables' values can reveal unexpected behaviour of the Fourier transform which can be very instructive.

3.5. Re-use of Procedural Tasks

The majority of tasks within GIPSY is still of the procedural kind. Of course these tasks can still be used the way they were originally designed, but with GIPSY's standard facility that allows one task to run another, some of these tasks can be re-used in novel ways. For example the task ROTMAS can delegate the calculation of a rotation curve to the old task ROTMOD. It is worth mentioning that some tasks are quite old (e.g., ROTMOD dates from 1984), but this is an advantage rather than a disadvantage. Many tasks have matured over time and are in excellent condition.

4. Conclusion

Evolution, rather than redesign, has been the main mechanism by which GIPSY has been able to keep up with modern requirements. This has enabled our focus to remain on the development of applications that implement new ideas in data analysis.

References

Allen, R. J. & Terlouw, J. P. 1981, in Proceedings of the Workshop on IUE Data Reduction, ed. W. Weis. (Vienna Observatory), 193

Allen, R. J., Ekers, R. D. & Terlouw, J. P. 1985, in Data Analysis in Astronomy, ed. V. di Gesù, L. Scarsi, P. Crane, J. H. Friedman & S. Levialdi, (Plenum Press, N.Y.), 271

Swaters, R. A. 1999, Ph.D. thesis Rijksuniversiteit Groningen

Van der Hulst, J. M., Terlouw, J. P., Begeman, K., Zwitser, W. & Roelfsema, P. R. 1992, in ASP Conf. Ser., Vol. 25, Astronomical Data Analysis Software and Systems I, ed. D. M. Worrall, C. Biemesderfer, & J. Barnes (San Francisco: ASP), 131

Van der Marel, R. P. & Franx, M. 1993, ApJ, 407, 525