FULL VOLUME TITLE
*ASP Conference Series, Vol. *, © 2008*
Editor 1, Editor 2, and Editor 3, eds.

Chapter 1: Mirage: A Visualization Tool for VO Data

Tin Kam Ho

Introduction

A fundamental need in analyzing data from large archives is to find correlations among different objects or events, Mirage is a tool that addresses this need. Specifically, it offers intuitive graphical operations to track correlations across different attributes and views of the same object, or correlations among different levels of analysis. It provides a means to display data together with the results of automatic clustering, offering a way to facilitate interactions between pattern recognition capabilities of humans and machines.

Early trials have been made to connect Mirage to VO archives, external web services, dynamic data streams, and statistical analysis libraries. A broader goal is to provide a platform for integrating several modern technologies into the scientific inquiry process. In this chapter we describe the basic functionalities of Mirage, its connections to the VO, and ongoing developments.

1. Overview

Mirage has been developed with design goals to make it

- intuitive, easy to learn and use,
- supportive of multiple modes of usage (online with or without graphics, offline without graphics, etc.),
- conforming to a minimalist approach, and
- flexible and scalable.

Central to the software is an interpreter that accepts commands like loading a data file, sourcing a script, and launching a cluster analysis. A graphics library contains subclasses of Java Swing widgets and provides the exploration commands. The dataset is managed as a shared list of objects representing the entries and the points to it is passed to all graphics and analysis classes. Peripheral facilities are included to access external databases via JDBC, and to link external graphics panels via a shared set of selecting and highlighting operations. An auxiliary package is called that provides basic file I/O, data manipulation, and graphics functions. A "pattern recognition" package is included that contains the cluster analysis code.

Mirage was first developed in response to a need to analyze results of large-scale photonics simulations that involve numerical vectors and scalar attributes, clus-

ters, and shape profiles. Later it was adapted to include analysis of astronomical images and derived catalogs [Ho 2004]. Designs were motivated to address needs in these contexts, which also happen to be important concerns in the VO.

2. Basic Functionalities of Mirage

The core of Mirage is a Java program that supports file based I/O, the basic exploratory operations, and some simple clustering algorithms. Access to external data archives, interaction with local or remote analysis services, and linkage to dynamic updating facilities are supported through extensions. Trial interfaces have been written for these extensions and are expected to be further enhanced in future releases. We first describe some simple and typical operations with a data matrix and associated mixed-type inputusing the current v0.3 release of Mirage.

Starting Mirage with a Data Matrix

A simple way to use Mirage is to start with a data matrix, as in:

```
> java –Xmx750M –jar Mirage0.3.jar –data data/IRASpsc300.dat
```

where the flag `–X` asks for a larger heap size than the default, `–jar` passes the packaged Mirage code, and `–data` gives the location of the data matrix. Using the version of Mirage included in the NVOSS software CD, and assuming the NVOSS environment has been set up, the same data file could be displayed using the command:

```
> mirage data/IRASpsc300.dat
```

Note that in both cases Mirage assumes the current working directory is the one containing the jar file and so the location of the data file is specified relative to that (in this case, in the 'data' subdirectory).

The data file is expected to be a whitespace-delimited ASCII file with one row of the data matrix per line. If no format header is given, Mirage will assign variable names X1, X2, etc, to the columns. A slightly more sophisticated way to prepare a data file is to precede the matrix with a "format var" statement that gives the column names:

```
format var ObjectName RA Dec Nobs J12 J25 …
```

One can further designate a particular column to be used as the "id" string. An "id" string does not participate in the graphical plots, and is put as the row header in table displays. For very wide tables, one can scroll horizontally to see the rightermost columns or move the columns around by dragging their header, but the id string stays at the head of the row to identify the entries. For example, the following command specifies the variable "ObjectName" to be used as an "id" field:

```
format id ObjectName
```

A few other format statements can be added to describe the data. These include "format vec" statements to define a vector that consists of selected columns in the data matrix, "format text" statements to force the interpretation of one particular column to be a text (instead of numeric) variable, and "format extent" statements to label and restrict the ranges of the axes in a vector plot. The details of these are described in the "Overview" page of the help menu in the software. When additional format statements are to be added, those should also precede the data matrix.

The first Mirage screen is shown in Figure 1:

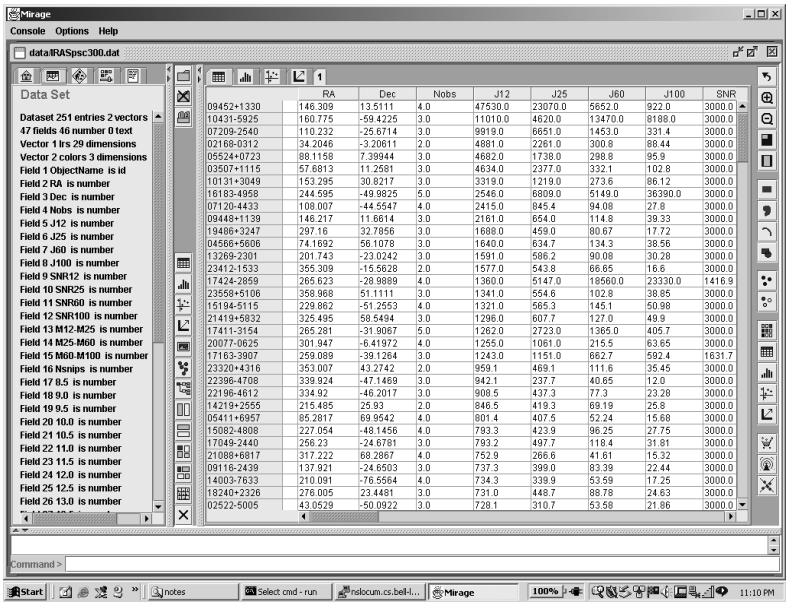


Figure 1. The Mirage startup window showing a loaded data matrix.

The Mirage display contains a stack of frames each divided into several major panels. Each data matrix will be in its own frame (the topmost box marked with 'data/IRASpsc300.dat' in Figure 1), and each frame contains the *console panel* on the left, a *graphics panel* on the right, and a *command window* at the bottom. Each of these panels subdivides into other components like pages and toolbars. For a detailed graphical illustration of the layout and the meanings of various icons, please refer to a set of annotated screen shots in "*A Graphical Guide to the Usage of Mirage"* that is available from the project web site.

The console panel gives a number of tabbed pages. These contain a description of the dataset as read, an import/export control panel, a work area where one can launch a clustering analysis, a configuration page where one can set graphical properties like sizes of various markers, and a Log area that echoes the commands being executed. The last statement requires some explanation: at the heart of Mirage there

is a command interpreter. Except for the operations initialized from the graphics panel, most of the other I/O and analysis procedures are executed as commands passed to the interpreter. Many of these commands are directly available to the user and can be entered at the "Command" prompt at the bottom panel. The "Help" menu at the top contains a list of such commands.

Explorations in the Graphics Panel

The majority of the exploratory utilities are offered in the graphics panel. This panel can be expanded to use all of the screen space by pushing the split dividers all the way to the left to close the console panel and to the bottom to close the command panel. Five tabbed pages are offered as the program starts: a table view (see Figure 1) is the default tab shown; a histogram view, a scatter plot, a parallel-axis display, and a "combo" page (under the number '1' tab) that contains four tiles for these other four views. Additional pages can be added by clicking the top button (i.e. the folder icon) on the tool bar to the left of the graphics panel, the two buttons below it are for deleting and renumbering these combo pages.

The lower group of buttons on the left toolbar controls how the current combo page is subdivided and what display goes into each subdivision. Resting the mouse over a button for a second or two brings up a textual tool tip for that button. To create a new view of the data or to further split the display, drag the appropriate button to the desired place in the current combo page. The pane splitters can be used recursively to tile a page arbitrarily. This can yield a display as complicated as the one shown in Figure 2.

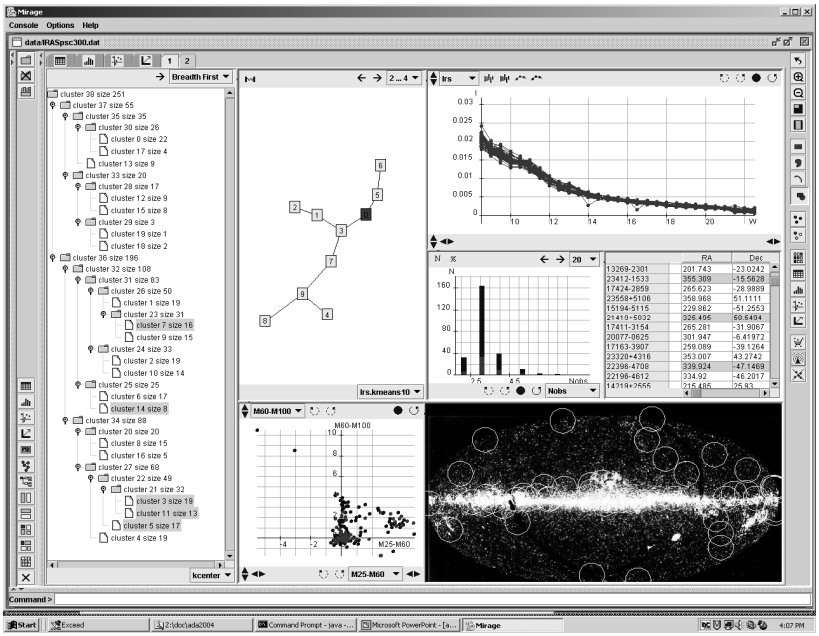


Figure 2. A "combo page" divided into multiple display areas. Each display area shows a different view of the same data. Note the Console panel to the left has been closed to allow full use of the window for the Graphics panel.

The power of Mirage comes from the linkage between these various types of displays. While each display offers its own set of commands for exploring the data, it is the combined use of these commands and the capability for one to go back and forth between the displays that are most valuable. The buttons on the right toolbar control what is to be done for the display or objects pointed to by the cursor: The top group controls panning, zooming, repositioning, and resizing images and scatter plots so that they fit entirely within the panel. The second group are "selectors" that allow the user to highlight items with the cursor, followed by the third group for toggling between a monochrome and a colored display. The bottom two groups are for actions to perform with the selected items. These can be coloring the selection, bringing up a restricted display for the selected items only, saving the selection in a "cart", and broadcasting the selection to other views. Note that hidden pages (i.e., those under tabs with numbers different from the one in the top view) require a re-broadcast when the page is brought into view. The last button deletes a current selection.

Linking Images with Data Matrices

An important use of Mirage is to relate images, the features extracted from them and analysis of such features. In this case, the entities of analysis are expected to be objects or regions in the image. These are represented as rows in the data matrix, and the attributes (features) as columns. To establish the relationship, two columns are needed to specify the center location of this object or region in the image. Also, the name of the image needs to be passed to Mirage with the following (example) format statement:

```
format image data/IRASpscallsky.jpg row y col x
```

where the path of the image is given relative to the Mirage startup directory, and two variables "x" and "y" are identified to be containing the row and column indices of the center pixel. The data matrix is then expected to contain a column called "x" and another column called "y". There is no limit on the number or size of images that can be associated with a data matrix, however, in practice such limits exist according to the amount of available memory and processor speed. Multiple images are listed for selection when the "image" icon is dragged to an empty panel from the left toolbar.

Launching a Cluster Analysis

Mirage supports some simple cluster analysis through the work panel that is accessible via the first ab (i.e. the yellow-diamond construction icon) in the console panel. This includes a partitional clustering algorithm known as the "k-means" method, and a hierarchical algorithm that can use one of three typical "linkages" (simple, average, or complete-linkage). There is also a "composite" algorithm that first finds the k-means partitional clusters and then computes a hierarchy of the cluster centers. Details of these algorithms can be found in [Duda et al. 2001]. A chosen algorithm is run when the "Run" button at the bottom is clicked. The results are visible when one

opens a "cluster display" panel by dragging the button labeled 'graph plot' in the tool tip for partitional clustering, or the one labeled 'tree plot' in the tool tip for hierarchical clustering to an empty tile.

The main reason for including these cluster analysis procedures is that very often structures in large, high-dimensional datasets cannot easily be seen, even with many different ways of projecting the points. Cluster analysis can also serve as a data compression method. This is especially true if the dataset contains attributes of several categories that are comparable only within each category. When the attributes are comparable only within each category, a useful thing to do is to construct clusters using only those attributes, and then compare the cluster structures arising from different groups of attributes [Ho 2002].

More sophisticated clustering and classification procedures are available in other analysis packages such as Weka, R, or PRTools in Matlab (see "Useful Links" in the references). Mirage does not attempt to reproduce facilities available in those packages. Nevertheless, one can relate the results of external classification to views in Mirage by including, as an additional column in the data table, the class index of each data entry. The class index can come from a file written by an external analysis package. It can be added to Mirage as part of the input data matrix, or in real time after the data matrix is loaded, with the command "addcolfrom *newFile newAttributeName*", assuming that the rows are ordered in the same way as the original data matrix. To look into the class effects, one can select points of a certain class (e.g., using a histogram on the class index) and broadcast it to other views.

Scripting and Offline Processing

Scripting is natural to Mirage because there is a command interpreter at its core. Scripts are plain ASCII files that contain a list of commands, one per line. At the command prompt in the bottom panel of each frame, the command "`source` *`script_name`*" would load the script into Mirage and execute the commands sequentially.

A script can be used to produce an animation that performs selection and highlighting of different interesting subsets, with desirable pausing in between. This is most useful when sharing the results of an analysis with another user. For example, if a particular trend is found to be interesting enough for others to follow, the commands used in selecting consecutive segments of the trendcan be put into a script file and replayed as an animation.

3. VO Interfaces

An extension has been developed to connect Mirage to existing VO standards and archives [Carliles 2003]. This uses the IVOA Client Package to load VOTables, and the IVOA FITS Package for FITS image support. To use this extension, copy the .jar file containing the extension (e.g. sloanNoMirage-3.5.jar) into in the directory where the Mirage package (e.g. Mirage0.3.jar) is installed, and add the "Property.dat" file from the VO-enabled Mirage distribution site. This file tells Mirage of an external

graphics panel that provides FITS image support. Then, start Mirage with switches that pass in the location of the VO registry and CAS servers as below. If the switches are omitted, default servers are used (`http://nvo.stsci.edu/voregistry/registry.asmx`, `http://voservices.org/registry/registry.asmx`).

```
➢ java -Xmx1024M \      -
  Dregistry="http://nvo.stsci.edu/voregistry/registry.asmx" \
➢ -DCAS_URL=http://casserver.edu/CasService.asmx/GetVoTable? \
➢ -jar sloanNoMirage-3.5.jar [datafile1 datafile2 …..]
```

Loading a VO Table

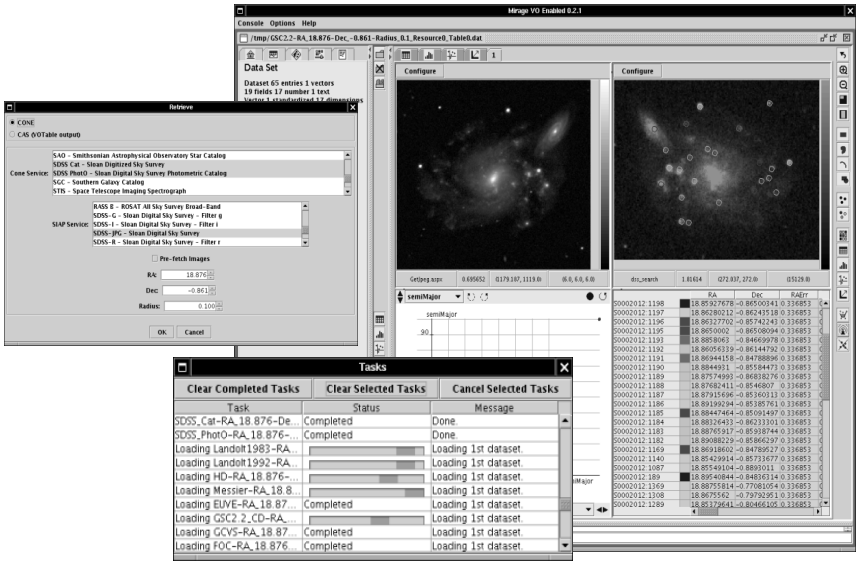
With this extension, a VO table can be loaded just as a regular data file under the "Console" menu. Notice that since the VO table specifications are evolving, older releases of VO-enabled Mirage may not be compatible with the latest versions. If that happens, one should obtain the latest release from the extension's distribution web site.

Accessing the VO Archives

Access to the VO archives is provided under the "Console" menu, through the menu item "New Dataset from VO Source …". This throws open a dialog box for "CONE/SIAP" search in the registry, or "CAS" search via a query to the SDSS Skyserver. The VO tables returned by these services are automatically loaded into Mirage, and the image URLs returned by the SIAP services are added to a list presented to the FITS image support module. More details can be found at the extension's distribution web site.

FITS Image Displays and Operations

The VO extension includes a FITS image support module that contains many sophisticated operations. The module uses WCS support and color maps from the JSky package. Beyond basic operations like zooming and panning, brightness and contrast adjustment, and intensity scaling in a single image, one can do synchronized zooming and panning when multiple FITS images with overlapping WCS ranges are displayed simultaneously in separated panels (See Figure 3).



4. Future Development

Future releases are expected to further isolate the core data manipulation and analysis functions from the view-specific exploratory operations. Interfaces are to be made public so that custom views or analysis code can be developed and linked to the core. Besides the VO enabling extension, other experiments have been attempted to link Mirage to network views, commercial relational databases, remote data analysis via SOAP, and dynamic data streams.

Mirage has been experimental software for proof of concepts in integrating human and machine pattern recognition techniques. Like many other experimental software, throughout the development, new features are conceived and added, and some of these caused revisions of the code organization. By now it is becoming clear that a certain set of features are especially desirable, and a few others are in high priority for further experimentation. Future releases are expected to incorporate such designs. An overriding theme is to provide capabilities for inquiring large scientific data sets in a way that is fun, painless, stimulating, and fruitful.

References

Carliles, S., Ho, T.K., O'Mullane, W., in "Proc. of Astronomical Data Analysis Software & Systems XIII," Strasbourg, France, Oct 12-15, 2003.

Duda, R.O., Hart, P.E., Stork, D.,G., Pattern Classification, John-Wiley & Sons, 2001.

Ho, T.K., in "Proc. of the 16th Int'l. Conf. on Pattern Recognition," Quebec City, Canada, Aug 11-15, 2002.

Ho, T.K., in "Proc. of the 17th International Conference on Pattern Recognition," Cambridge, U.K., August 22-26, 2004.

Useful Links

Mirage project page. http://cm.bell-labs.com/who/tkh/mirage

VO-Enabled Mirage. http://skyservice.pha.jhu.edu/develop/vo/mirage

Weka for machine learning. http://www.cs.waikato.ac.nz/~ml

The R project for statistical computing. http://www.r-project.org

PRTools, pattern recognition toolbox in Matlab. http://www.prtools.org