The MAGIC Analysis And Reconstruction Software

Thomas Bretz,[1] and Robert Wagner[2] for the MAGIC Collaboration
(1) Universität Würzburg, Institut für theoretische Physik und Astrophysik, Am Hubland, D–97074 Würzburg, Germany
(2) Max–Planck–Institut für Physik, Föhringer Ring 6, D–80805 München, Germany

Abstract

The robust and versatile MAGIC Analysis and Reconstruction Software ("MARS") has been developed for use with the MAGIC Telescope, to convert raw data into physics from FADC output to fluxes. MARS will be used for online analysis, standard analysis, and for the in-depth analysis of data from cosmic sources as different as gamma ray bursts, blazars, and supernova remnants. Moreover, MARS is a powerful tool for monitoring the data quality. The software is a ROOT-based [3] collection of C++ classes driven by a central event loop. It comprises a general base package for the analysis of event-based experiments, add-ons like background rejection algorithms, and tools specifically developed for imaging air-Čerenkov telescope data analysis, such as enhanced image cleaning algorithms. A software release is documented and available on the web [4].

1. Introduction

The phase I, 17m diameter MAGIC imaging air-Čerenkov telescope (IACT), currently becoming operational on the island of La Palma, has been designed to search the sky for gamma-ray sources above 30 GeV, of which there should be at least an order of magitude more than were observed with previous IACTs of higher threshold. Prime task for the analysis software MARS therefore was to provide a robust and flexible platform for dealing with the data from a variety of sources. Flexibility from the developer's point of view means that the software should be structured such that enhancements and add-ons are readily incorporated into the software. From the users' point of view, the software should be flexible and should have implemented a large number of tools to deal with a wide range of analysis requirements. We therefore choose the ROOT package as the platform for the design of MARS. ROOT implements a large number of basic features and a wide range of tools. MARS is a collection of definitions for basic interfaces (base classes), pre-defined algorithms, and data structures. The software is composed of the kernel which carries the implementation of general event-based data analysis algorithms, and a special analysis technique part for IACT applications. Interfacing to the telescope geometry allows for using MARS with all existing IACTs. Owing to ROOT's C++ interpreter, specific user-defined analysis tasks can be implemented without the need to recompile MARS, Besides this open strategy, MARS contains a standard analysis program for MAGIC data. Its main design aim is to be independent of the measured source.

2. The kernel

The kernel provides the event-loop driven core of MARS. Its I/O stream is structured by sharply separating the data (parameter containers) from the algorithms and their setup paramaters (called tasks). Parameter containers are collected in a list (parameter list), which is available to all classes operating on the data (tasks) and thus serves as a dictionary for all algorithms accessing the data. Within MARS, an event consists of all data provided by one or more of the experiment's subsystems and a corresponding timestamp.

A sequence of tasks defined in one object (tasklist) is applied to each event in an eventloop. Before the eventloop is actually started, the prerequisities of all tasks connected to it are first checked. One example is checking the availability of the necessary parameter containers which is done by calling a dedicated interface to each task (virtual method). This allows tasks to create their own output containers. Processing the eventloop, each task can decide wether an event should be skipped or the eventloop should be stopped. After the eventloop is finished, another dedicated method of each task is called which finalizes each task's job. By setting up different tasklists one can completely change the analysis performed. While the first task in a tasklist typically produces the events (e.g., reads the events from a file/memory or could be used to produce a random Monte Carlo event), the task at the end of the tasklist writes the data to an output stream (file).

The processing of each task can be made dependent on the return value of a filter. Because all filters have access to the parameter list, complex cutting algorithms such as background reduction algorithms can be implemented. A base class defines the standard interface for such a filter. As tasklists are tasks themself, they can be used in combination with filters to set up complex event processing trees in the analysis.

3. Additional basic features

In this structure, many helpful basic features have already been implemented, such as tasks for printing a parameter container, skipping an event (usefull in combination with a filter), writing parameter containers (containing experimental data, calculated results or setup information) into a ROOT file, and reading a file with an automatic structure detection. A general interface to parameter containers containing histograms is also available. The interface allows for filling of all histograms by the same task. For displaying results and for online checks (while the eventloop is still running) several enhancements of the ROOT GUI classes (Graphical User Interface) are available, such as popup menus and an online display (displaying and updating data structures – such as histograms – as they are built up during the eventloop).

For easy access to data member variables from outside in the precompiled code (such as a setup file) so-called rules have been implemented. Rules are based on the ROOT dictionary giving access to data members of classes by their names. Thus variables (used e.g. for signal/background reduction) can be setup with a string (e.g., "sqrt(ContainerName.fValue*2)"). This can be used for simple testing of the behaviour of variables on algorithms. Similar to this, simple filter cuts (filters) can be setup (e.g. "sqrt(ContainerName.fValue*2)<5").

Using a ROOT parser and interface for setup files, the contents of a setup file are distributed to all tasks in an eventloop before the eventloop is performed. For debugging and logging a logging stream based on the C++ streaming classes has been implemented. This logging system has several debug levels (all, informal, warning and error) which can be switched off by user request and which are displayed in different colors on the console. The stream can – at the same time – be redirected to standard out, standard error, a file and a graphical interface.

4. General analysis methods

Based on the general structure, a number of analysis methods have been implemented. The most interesting methods may be methods for signal/background separation, for example: one dimensional composite probabilities, kernel, nearest neighbors, and random forest [5]. The results of these methods are displayed in several plots, including a Nearman-Pearson diagram.

5. Imaging air-Čerenkov telescope specific algorithms

For IACTs one needs a number of specific algorithms. Using the base classes, storage containers for the storage of data have been implemented. In combination with a task dealing with Monte Carlo data (calculating thresholds, collection areas, etc.), an image analysis for IACTs has been implemented. The image analysis starts first with the creation of the image from the FADC data recorded by the data acquisition system, then treats blind pixels (broken pixels, or pixels lightened by a bright star) and applies different image cleaning algorithms, to finally perform a calculation of Hillas and advanced image parameters (based on HEGRA experience). For displaying the images a camera (event) display is available which is not bound to a fixed camera geometry. Also a graphical interface to display raw data and to display data check plots (e.g., distributions) is available. After signal/background reduction the number of gamma and background events is computed, and by comparing with the Monte Carlo data flux and corresponding light curve can be determined.

6. Telescope specific part of the software

The part specific to the telescope mainly implements the geometry of the MAGIC telescope (focal length, camera layout, etc.). Currently, the geometry of the MAGIC telescope, HEGRA CT1 and the planned ECO1000 telescope has been implemented. Since a standard interface for the geometry has been defined by a base class, one can easily enhance MARS to a data analysis software for any other telescope as well.

7. Conclusion

MARS is a robust and flexible standard analysis software for event-based experiments, such as MAGIC with its unprecedented sensitivity and low energy threshold. Consistency of the MARS standard analysis with previous analysis methods employed for the analysis of HEGRA CT1 data has been successfully demonstrated using data from Mkn-421 observations. Special analysis techniques for low-energy gamma ray events amd future changes in the experimental setup (e.g., high quantum-efficiency camera, conincidence telescope) can readily be implemented into MARS.

8. References

1. M.Martinez for the MAGIC collaboration, these proceedings
2. J.A. Barrio et al, MAGIC design study, Preprint MPI-Phe/18-5, 1998
3. ROOT Web Pages, http://root.cern.ch/
4. MARS Web Pages, http://magic.astro.uni-wuerzburg.de/mars/
5. Leo Breiman, http://stat-www.berkeley.edu/users/breiman/

9. Acknowledgment

We gratefully acknowledge support by the German BMBF through the grant 05 CM0MG1.