MARS - CheObs goes Monte Carlo

Thomas Bretz[x], Daniela Dorner[***]

[x] *Universität Würzburg, Am Hubland, 97074 Würzburg, Germany*
[*] *Institute for Particle Physics, ETH Zurich, 8093 Zurich, Switzerland*
[**] *ISDC Data Center for Astrophysics, University of Geneva, 1290 Versoix, Switzerland*

***Abstract*. The concept of a Modular Analysis and Reconstruction Software (MARS - CheObs ed.) has been continuously enhanced in the past years. With its simplicity and the contained automation technology, a powerful package has been designed, best suited to non-interactively analyze the huge amount of data produced daily by the current generation of imaging air-Cherenkov telescopes.**

Since the concept was designed to fit the needs of any kind of event based analysis, it is a consistent step to include also a layer for detector simulations which also processes individual events, i.e. simulated air-showers. Obviously, having the same framework for the simulation, as for the analysis, has the advantage that for the simulation, all tools already contained are provided like logging, I/O, graphical display and naturally all analysis tools.

***Keywords*: IACT, Simulation, MARS - CheObs**

I. Introduction

With recent experiments like MAGIC, H.E.S.S. and Veritas, the number of sources in very high energy astronomy has increased exponentially. This success could be achieved due to a large improvement in light detection technology. Consequently, the amount and rate of recorded data has also increased dramatically in the past years. With better imaging detectors, image analysis could be refined and background suppression improved. Additionally, new small projects, e.g. DWARF [1], will deliver long-term monitoring data for which consistent processing of all data is vital. Since Cherenkov Telescopes lack the possibility for an absolute energy calibration, the analysis of their data relies on the quality of their simulations. As more powerful instruments become more sensitive to not only small changes in the telescope performance, but also to the daily changes in atmospheric conditions, advanced simulations are mandatory.

The easier the framework for such simulations is, and the more simple its appliance is, the easier the implementation and maintenance of improvements will be. This leads to more robust results. The obvious next step is to implement a full Monte Carlo chain into the existing modular and flexible MARS framework.

II. The MARS framework

Having in mind that automation and programs, easy to use for the analyzer, save a lot of work time, hence available to provide a proper interpretation of the obtained observation result, the framework of the MAGIC Analysis and Reconstruction Software (MARS) [2], [3] has been designed from the beginning to support robustness, easy maintenance, easy and simple enhancability, easy automation and especially user-friendliness.

This could be achieved by a modular, multi-layer concept which features separation of data structures and algorithms, plug-ins for easy enhancements, data structure independent I/O, algorithms independent of the detector setup in use, algorithms independent of the detector type, easy reorganization of the needed and available tasks to programs, and a powerful framework for data processing with a well defined interface to the available tools.

Furthermore, there are many tools existing which can be reused in any other environment like redirection of logging stream to several outputs like console, text-file, html-file, etc. (easy enhancable by plug-ins), resource files and an easy to program interface for access from the algorithms, a graphical on-line display in which the progress of an ongoing analysis can be displayed, full I/O of the whole display and easy access to the contents of a stored display, and a powerful class for storage and conversion of high precision timing.

Consistently implemented, this leads to an easy to maintain, robust and powerful analysis setup for the developer, and for the end-user to a well arranged data interface, simple quality verification of the data and finally in an easy production and handling of the end product, like detection, spectrum and light-curve.

Consequently, the existing framework is object oriented and based on C++. As an underlying platform for I/O, analysis tools and graphics, e.g. histograms, the ROOT framework [4] has been chosen.

More details can be found in [5].

III. Achievements

The current setup achieves the processing of more than 100 TB of MAGIC raw data, taken since summer 2004, within less than 20 days. This numbers are obtained with a high-performance storage system and about 40 CPUs installed in Würzburg. By the ability of re-processing the data, it can be ensured that all data are always consistently processed by the latest available software, which gives the end user the most advanced data end product at hand.

A consistent data sample of all MAGIC data is especially remarkable, taking into account continuous changes to the hardware in the past years which had to be followed by the software.

This was rendered possible by the modular concept of the underlying framework which makes the development of workarounds for hardware failures, and new algorithms for hardware changes, easy and fast to be implemented without further complication of the existing structures.

Within the same framework, existing algorithms can easily be replaced by new algorithms, for example new background suppression methods, which can then be used as an improved replacement or even in parallel. Processing the data in different independent ways gives a strong handle on data quality assurance.

Since the hardware setup of the MAGIC experiment has been improved quite fast in the past years, sometimes changes in the software had to be implemented quickly. Therefore, often new implementations entered the software, bypassing the framework. For this reason, the further development of the framework, with its own implementation of an analysis, has been split from the main development of a MAGIC analysis. This keeps the framework consistent, and ensures further development independent from urgent experimental requirements.

Aiming at a robust, consistent and powerful framework and not at following urgent needs of the experiment, it has been given the name Modular Analysis and Reconstruction Software (MARS) - Cherenkov Observatory edition (CheObs) for better distinction.

IV. Design goals

To profit from these achievements also in the important step of simulation, a new program has been included, called *ceres* (Camera Electronics and REflector Simulation), which follows the successful concept of the MARS framework.

It has turned out in the past that simulations written for special purposes, not modular and flexible, ran into problems when minor or major hardware upgrades were installed. Furthermore, they have the problem that simulation of future projects is always strongly bound to existing setups. The implementation of new ideas, e.g. new triggers, always implies the knowledge and understanding of large fraction of the analysis code, while MARS aims for splitting the code into function modules. Thus, the developer only has to understand the piece of code which implements the task he is interested in.

Together with the existing automation concept implemented in the MARS framework [6], this will allow for fast and easy studies based on a large fraction of the parameter space available.

V. Overview of the elements of the simulation

A. Input

The input are files, written by the air-shower simulation CORSIKA [7] either in its own format or in the optional EventIO-format [**?**]. Assuming logically identical contents, the modularity would also allow an easy implementation of basically any other format written by a different air-shower simulation.

The setup of the simulation is given in a resource file as in any other MARS program. These resource files have an easy structure and the meaning of there contents are defined by the programs. The interpretation is done by root's TEnv class. In this case the resource file contains mainly the detector setup but also details like histogram binning (for histograms for which a good choice for the binning is not automatically available from header informations).

B. Reflector

Cherenkov telescopes usually have a segmented reflector. The implemented simulation consists of all individual mirrors and is optimized for execution speed. Since the shape of individual mirrors is described by dedicated classes, any mirror shape can be implemented. This approach has been chosen in favor of a more general one to optimize the CPU hungry check which of the hundred mirrors has been hit by a photon. The different classes are addressed by its name from a resource file describing the whole reflector. Consequently, new mirror shapes can be added without applying changes to the existing code. The structure of the file allows to assign a variable number of parameters to each mirror type, e.g., for a hexagonal mirror only the diagonal must be known while for a rectangular mirror both side lengths are needed.

The point spread function of each mirror is simulated smearing the normal vector of the mirror surface at the photon incident point by a two dimensional Gaussian.

The calculation of the incident point is done analytically, for accuracy and speed reasons. Currently, all mirror surfaces are assumed to be spherical with user-defined focal lengths. Enhancing the code to support also parabolic and aspherical mirrors would be simple, but is currently not needed.

C. Pointing

The pointing, i.e. the orientation of the telescope w.r.t. the shower orientation, is as flexible as possible. The orientation of the magnetic field, as stored in the run-header of the CORSIKA output, is taken correctly into account.

Depending on the particle type and user request, several pointing options can be chosen. This includes pointing parallel, or with a fixed offset on the local sky, to the shower axis, but also along the main direction around which the showers have been simulated (*view cone* option). Also simulation randomly distributed

around the shower axis with a fixed offset is possible. As soon as the implementation of shower production along a star's trajectory [8] is finished in CORSIKA, also fixed offsets on the celestial sky will be available.

D. Atmosphere

Since the shower simulation in use, CORSIKA, does not simulate the absorption of the atmosphere, although the refraction is simulated, this must be implemented in the detector simulation. For this purpose, the well tested code from the MAGIC detector simulation has been chosen. It implements ozone and aerosol absorption depending on the production height, i.e. mean free path, and the wavelength of the photons. By pre-calculating the mean free path in a discrete grid of height and emission angle, a fast atmosphere simulation is carried out. Accuracy is maintained by interpolation.

The coefficients describing the atmosphere are read from the CORSIKA output, so that it is guaranteed that the same atmosphere is used as in the shower simulation.

Since these coefficients are the result of a fit done internally in CORSIKA, also the layer boundaries are usually not fixed. For the simulation these layer boundaries must either be given manually or set fixed in CORSIKA. Newer CORSIKA versions write these boundaries into their output files, hence the correct boundaries are automatically taken.

E. Additional photon losses

Additional photon losses in general are implemented by spline interpolation of a wavelength dependant absorption curve. To make sure that the spline behaves well in the region of interest, the interpolated curve is plotted in the output. Where possible, absorption effects are simulated as early as possible, i.e. before the photons hit the mirror surface, to gain execution speed. Absorption effects are, e.g. mirror reflectivity, transmission of the camera protection window, reflectivity of the light guides or acceptance of the photon detectors. From all absorption curves, a single absorption curve can be calculated to reduce execution time.

F. Winston Cones

The Winston cone acceptance is simulated by a self made ray-tracing program [9]. Since light guides are non-imaging, i.e. the image on the photon detector has no information about the incident point of the incoming photons, their acceptance can be simulated just by an incident angle dependant absorption curve. This curve is calculated by the program and used as input in ceres.

Furthermore, the geometry of the cones, i.e. which cone is hit by which photon, is implement in the same way as pixel geometries in MARS in general and similar to the implementation of the reflector geometry. In principle, it allows the usage of any kind of geometry, although currently only variable geometries using hexagonal and rectangular shaped cones are in use. This routine just tags each photon with the index of the cone it has hit.

G. Diffuse night-sky background

To simulate the diffuse night-sky background (NSB), randomly distributed photons are created for each cone separately. The distance between two consecutive photons is simulated as an exponential distribution. Its rate is calculated by ceres from the measured rate on the observation site (user input), the absolute collection area and angular acceptance of the cones (or the reflective surface), and the wavelength dependent acceptance of the photon detectors.

In addition to this variable rate, a fixed rate of photons hitting each photon detector can be added, which can, for example, be used to simulate the dark current of semi-conducting detectors.

H. Photon Detectors

In the case, G-APDs (Geiger-mode avalanche photo diodes, [10]) are used as photon detectors, the conversion from photons hitting the detector to the output signal is more complicated than for photo-multiplier tubes. G-APDs are made of a number of individual cells which can only detect one photon at a time and they need some time for recovery before the next photon can be converted to a signal. Thus, the response is non-linear. Furthermore, the so called optical crosstalk can induce signals higher than expected from a single photon hit. All these effects, as well as the saturation due to the continuous hits from photons of the diffuse night-sky background, are simulated. Since a typical G-APD has hundreds to thousands of cells this is a rather time-consuming step and therefore work is carried out to replace the full simulation by a more analytical approach.

In addition, the signal of each converted photon has a typical width in amplitude, simulated as a Gaussian.

I. Pulse shape

For every photon detector and signal, amplitudes and their arrival times are now known. The signals hitting one photon detector (primary analog channel) are "digitized" calculating the amplitude at discrete times by superposition of the induced pulses. These pulses are copied from a master-pulse, described by a table of amplitude and time, interpolated by a spline.

To make sure that the region of interest, i.e. the later readout window, will be realistic, the whole history of each channel is simulated including all effects like NSB, since the earliest time a pulse can influence the region of interest.

J. Analog noise

In addition to the pulses, analog noise, i.e. Gaussian noise on top of a positive base voltage, is added to the primary analog channels.

K. Signal summing

These primary channels (photon detectors) can now be summed into the final analog channels (pixels) which later trigger or get readout. The summing is implemented by a look-up table read from a file.

L. Trigger

The current trigger implementation aims to be as flexible as possible to cover a wide variety of different possible trigger setups. Of course, the modular design of the whole software, and in particular also the trigger simulation, allows to exchange parts of the trigger or the trigger as a whole easily.

In a first step, the current implementation allows, again by a look-up table, to sum several analog channels to new trigger channels.

A second step discriminates each trigger channel. The discriminator output is calculated analytically by spline interpolation of the trigger channels. The output of each discriminator is a digital signal of either a fixed length, or a length corresponding to the time the signal is above threshold. If a signal is less than a predefined time above threshold, it is omitted.

A last step now checks for coincidences in patterns. The patterns are also defined in look-up tables. A gate length, i.e. a minimal length the signals must overlap, can be defined. The easiest coincidence patterns is a list of all channels which would, consequently, trigger if a single pixel would be above threshold.

M. Optimization of the trigger threshold

The ideal trigger threshold mainly depends on the number of accidental coincidences, i.e. triggers due to photons from the night-sky.

Letting the simulated electronics trigger on pure night-sky events without simulated showers, the trigger rate can be estimated from the distribution of two consecutive triggers. Since this distribution is assumed to be exponential also the distance from the first trigger after any artificial point in time should be exponential. As the artificial point in time, the beginning of the region of interest in the simulation can be chosen. Consequently, from an exponential fit to the distribution of the moment, when the first trigger in a simulation window takes place, the trigger rate can be concluded. Doing this with several trigger thresholds allows to extrapolate to the point at which the trigger rate falls below a rate acceptable by the electronics, or clearly below the expected trigger rate induced by air showers.

N. Readout

In the readout, the analog channels, stored already discrete in time, are also discretized in amplitude with the properties of the readout system. This can also include saturation effects and non-linearities.

Currently, just a linear discretization with variable number of bits, and a cut-off at saturation is implemented.

O. Visualization

To understand the performance and correctness of each step of the simulation, histograms showing the result of each individual step are filled. This allows the user to detect problems with the setup already at the beginning. In addition to photon, arrival time, and incident angle distributions for every step, energy, photon number, impact radius and incident angle distributions are available as well as energy threshold and acceptance curves.

P. Output

The ceres output, apart from the histograms written in MARS' display format, consists of an output stream basically identical to raw data format. Two additional streams contain the photon distribution on the camera plane and image parameters calculated from pure photon events, i.e. without noise and the simulation of the readout electronics.

VI. Conclusion

The MARS framework has, once more, proven that it is suited for several different data processing chains as needed in modern event based analysis. Furthermore, it now implements a simulation framework suited for detector simulation of imaging air Cherenkov telescopes. With this approach a flexible and modular simulation platform is available which fulfills the needs for planning new projects but also the extreme requirements in case of long-term observations. It is particularly relevant to have as less bias by non-ideal simulation of the detector performance or the observation conditions.

Currently, studies on the performance of different designs for the DWARF project are ongoing.

References

[1] T. Bretz *et al.*, *Status of the DWARF project for long-term monitoring of bright blazars*, in these proceedings
[2] T. Bretz *et al.*, *Roadmap to a standard analysis*, in Proc. 2^{nd} International Symposium on High Energy Gamma-Ray Astronomy, 745, 730 (2005)
[3] T. Bretz *et al.*, *Standard analysis for the MAGIC Telescope*, in Proc. 29^{th} ICRC, 4, 315 (2005)
[4] ROOT System home page, http://root.cern.ch
[5] T. Bretz, D. Dorner, *MARS - The Cherenkov Observatory edition*, in Proc. 4^{th} International Symposium on High Energy Gamma-Ray Astronomy, 1085, 664 (2008)
[6] D. Dorner *et al.*, *Automatic Monte Carlo Production for Imaging Air Cherenkov Telescope*, in these proceedings
[7] D. Heck *et al.*, *CORSIKA: A Monte Carlo Code to Simulate Extensive Air Showers*, FZK report 6019 (1998)
[8] M. Doert, diploma thesis, TU Dortmund University, 2009, in preperation
[9] I. Braun *et al.*, *Solid Light Concentrators for Cherenkov Astronomy*, in these proceedings
[10] T. Krähenbühl *et al.*, *Geiger-mode Avalanche Photodiodes as Photodetectors in Cherenkov Astronomy*, in these proceedings