GrOptics User's Guide
Version 2.2

Charlie Duke
Grinnell College
Grinnell, Iowa 50112
duke@grinnell.edu

Akira Okumura
Solar-Terrestrial Environment Laboratory
Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan
oxon@mac.com

July 15, 2012

Abstract

GrOptics is a detailed simulation program for ray-tracing Cherenkov photons through large arrays of atmospheric Cherenkov telescopes. Shower packages, such as GrISU [2] and CORSIKA [3], provide Cherenkov photons after conversion to GrISU format. The output to a ROOT file records individual photons striking the camera surface. The package models both VERITAS Davies-Cotton (DC) and Schwarzchild-Coudee SC) telescopes with all telescope parameters taken from input files. There is no limit to the number or type of array telescopes. Adding new telescope types, input and output formats, etc. is straightforward using standard C++ coding techniques with existing base classes. Reference [1] gives the code download site.

1 Introduction

GrOptics is a detailed C++ Monte Carlo ray-tracing program to study the passage of atmospheric Cherenkov photons through telescopes designed to study atmospheric Cherenkov air showers. Photons produced by standard air shower codes enter the telescope; the output ROOT file contains tree records of the photons that strike the telescope cameras. GrOptics provides the input to the CARE telescope electronics code [4].

There are no intrinsic limits to the number or type of telescopes placed in the air Cherenkov telescope (ACT) array. Currently, the code contains two concrete

telescope classes: for VERITAS DC telescopes and for SC telescopes. All array and telescope parameters are placed in input pilot or configuration files.

1.1 Installation

1. GrOptics relies heavily on ROOT[5]. I use the following installation method:
   - Be sure you have installed gsl for use by ROOT
   - Download the ROOT source package [5]
   - Follow package instructions to configure and to make, but do not specify a –prefix directory with configure.
   - Setup all necessary ROOT environmental variables by sourcing *thisroot.sh* or *thisroot.csh* in your <rootDirectory>/bin
2. The SC telescope class uses the ROBAST package [6]. ROBAST (ROot BAsed Simulator for ray Tracing) is a non-sequential ray tracing program which utilizes the 3D geometry library in ROOT. The ROBAST package is automatically downloaded by *curl* within *make* when producing the GrOptics executable.
3. After installing ROOT, download the GrOptics git repository [1]. Go to the GrOptics directory and run *make* to produce the grOptics executable.
4. To test the installation, execute *grOptics* from the GrOptics directory. The code will use the default configuration and pilot files and a test Cherenkov photon file, all within the GrOptics/Config directory, to produce an output root file, *photonLocation.root*. Other test possibilities (see later sections) for using this configuration are telescope drawings and psf camera plots.

1.2 QuickStart

The *grOptics* executable can use the configuration and pilot files and a test Cherenkov photon file, all within the GrOptics/Config directory, downloaded with the git repository. The file, *opticsSimulation.pilot* steers the simulation and specifies the output. The file, *arrayConfig.cfg*, defines the ACT array. You'll see the downloaded file defines a four-telescope DC array that is compatible with the *photon.cph* test input Cherenkov photon file. The parameters in these files are documented both in these files and in following sections in this document.

1. Execute the grOptics code from the GrOptics directory to produce *photonLocation.root* as specified in *opticsSimulation.pilot*. You'll find the output trees with records of the photons on the camera surface for each telescope in this file. You'll also find a history ROOT file for each telescope which documents the history of each photon incident onto the telescope (useful for debugging). You can turn off the creation of these files by removing the leading asterisk of the *PHOTONHIST*

2. To create an *opengl* drawing of a Davies-Cotton telescope using the ROOT geometry classes, add a leading asterisk to the *DRAWTEL* record in *opticsSimulation.pilot* and execute *grOptics*. Note that for DC telescopes, the code does not draw the mirror facets as the ray-tracing for the facets does not use the ROOT geometry classes. Have a look at the *canSpot* figure.

3. To replace an DC telescope with an SC telescope, open the *arrayConfig.cfg* file and activate the *TELFAC SC* SC telescope factory record. Then, in one of the *ARRAYTEL* records replace the *DC* with *SC*. You can then run the same tests as described above. Note that the ROOT geometry classes will draw the complete SC telescope (in contrast to the DC telescope drawing)

4. To produce a series of spot patterns on the camera, activate the *TESTTEL* record in *opticsSimulation.pilot* by adding an asterisk. Since the code produces the spot photons internally, change the *NSHOWER* from -1 and -1 to 1 and 1 (fixing this requirement is on my to do list). Execute *grOptics*.

2 Code Overview

See *DevelopersGuideGrOptics.pdf* for more details. This section only introduces standard telescopes, telescope arrays, and the various coordinate systems used in GrOptics.

GrOptics uses standard C++ plus ROOT.

2.1 Standard Telescopes

The GrOptics package currently produces telescopes from each of two telescope factories, one for DC telescopes and one for SC telescopes. All telescope parameters are in configuration files to maintain maximum flexibility. However, having separate configuration files for each array telescope leads to large, unmanagable file sizes. Thus, the factories can produce a limited number of standard telescopes based on configuration files in the *GrOptics/Config* directory. The factories then use edit records taken from the configuration files to change specific telescope parameters, e.g. the mirror reflectivity, thus maintaining complete flexibility for parameter selection for individual telescopes. The edit records use matlab colon notation so that a single record may change a parameter for a range of telescopes and elements within each telescope, e.g., facet reflectivities. Edit records for additional parameters can easily be added.

None of the telescope dimensions or super structures may be edited. Changing these dimensions requires a separate standard telescope.

2.2 Telescope Arrays

The telescope factories provide individual standard telescopes, after editing, to the ACT array as defined by telescope type, standard ID number, ground location, and pointing offset. Telescope numbering starts at 1, not 0.

2.3 Coordinate Systems

Understanding the coordinate systems used in our simulation codes is always difficult, especially if your memory is as bad as mine. So, these descriptions should help.

2.3.1 GrISU Ground Coordinate System

The GrISU Cherenkov files either from GrISU or from the CORSIKA I/O Reader use the following coordinate system:

x: East
y: South
z: Down

to form a right-handed system

2.3.2 GrOptics ground coordinate system

The incoming GrISU produced incoming photons are immediately transformed by GrOptics to the system:

x: East
y: North
z: up

forming a right-handed system. Note that the telescope locations in the VERITAS GrISU configuration file use this coordinate system.

2.3.3 Telescope Coordinate System

Each telescope has a coordinate system with origin at the telescope rotation point and with z axis pointed toward the sky along the optic axis of the telescope. In stow position (DC telescopes), the z axis is horizontal and pointed toward the North, the y axis is down, and the x axis is toward the East. To point the telescope to a given location on the sky, GrOptics first rotates the telescope about the vertical through the azimuthal angle. Then, it raises the telescope about its new x axis through the elevation angle. This new coordinate system is the "telescope coordinate system". Prior to injection into the GTelescope class, the photons undergo a coordinate transformation to this telescope coordinate system.

2.3.4 Array Coordinate System

Prior to added pointing misalignments, all telescopes point to the same location on the sky with coordinate systems previously defined. The array coordinate system is similarly defined, but its origin is at the origin of the array.

2.3.5 Camera Coordinate System

For historical reasons, the camera coordinate system's y axis is a reflection of the y-axis of the telescope coordinate system. For the VERITAS telescope, stand in front of the camera with the telescope in stow position. The camera y axis is up and the camera x axis is to your right (to the East). The telescope y-axis is down; its x axis is to the right.

In the output tree of grOptics, the photon locations on the camera are in camera coordinates, both for DC and for SC telescopes. I'll add an input flag later to select photon camera locations in either telescope or camera coordinates.

3 Input Files

The git repository contains input files for steering the simulation, setting up the array, and defining the standard telescopes. These files have unique record flags and can be self-referential for combining into single files.

3.1 Simulation Steering

The default input steering file is *GrOptics/Config/opticsSimulation.pilot*. Execute *grOptics -h* so obtain command line options to use other filenames. The *opticsSimulation.pilot* file is fully documented. Much of the documentation is reproduced here for completeness.

The *opticsSimulation.pilot* file contains the following records. Each record has a leading asterisk (not reproduced here) when active. In the following list, the bolded name is the record flag with the adjustable parameter listing following.

FILEIN <filename of GrISU-type file>

FILEOUT <root filename><TreeName><telBaseTreeName><photonDirCosBranchFlag>

<root filename>: name of output data root file
<TreeName>: name of ROOT tree containing parameters common to all photons
<telBaseTreeName>: telescope number to be appended to this base tree name
<photonDirCosBranchFlag>: if 1, add dirCosineCamera branches.

LOGFILE <name of logfile>//
NSHOWER <number Showers><number Photons>

<number Showers>: <0, no limit
<number Photons>: <0, no limit

ARRAYCONFIG <filename: default ./Config/arrayConfig.cfg>

SEED <TRandom3 seed, default 0 : set by machine clock >

WOBBLE <xSource><ySource><source Extension><latitude>

<xSource><ySource>: coordinates (deg) of source in field of view <source Extension>: source extension radius (deg) <latitude>: latitude of observatory
If the latitude is less than 90 degrees, the source position in x corresponds to an offset in the east west direction while the y position corresponds to north south.
Example:
wobble North: WOBBLE 0.0 0.5 0.0 31.675
wobble East : WOBBLE 0.5 0.0 0.0 31.675

DRAWTEL <¡telescope number to draw¿, default 0 (no drawing)>

TESTTEL <telescope number><baseName for histograms >

PHOTONHISTORY <ROOT filename, tel.number to be appended ><tree name>

3.2 Array Configuration

The default array configuration filename, set in *GrOptics/Config/opticsSimulation.pilot*, is *GrOptics/Config/arrayConfig.cfg*. This file specifies telescope locations, types, and configuration files for standard telescopes. Edit records for individual telescopes usually appear in this file. The *arrayConfig.cfg* file is fully documented. Much of the documentation is reproduced here for completeness.

The array defined here may be a subset of the array used to create the photon file.

TELFAC telescope factory type and parameters

<factory type: DC or SC >
<photon reader type: GRISU (only option) >
<configuration filename >
<telescope edit filename >

ARRAYTEL parameters listed below

<telescope number, >0 >

The array numbering eed not be sequential and can be a subset of the array used to create the photon file.

<x telescope location (meters)>
<y telescope location (meters)>
<z telescope location (meters)>
<pointing offset x >: >0 is left on tangent plane in degrees
<pointing offset y >: >0 is down on tangent plane in degrees
<telescope print mode >: fully implemented for DC telescopes only

0: no printing
1: print summary information
2: add geometry details
3: add facet details

3.3 Telescope Parameter Editing

The telescope editing records in the git repository are contained in the *arrayConfig.cfg* file. The editing flags are *EDITDCTEL* for DC telescopes and *EDITSCTEL* for SC telescopes. Only the *EDITDCTEL* are currently implemented.

3.3.1 Colon Numbering Notation

The colon numbering notation (used in Matlab and Octave) is useful for containing multiple entries in a single record. It is easily explained with several examples.

$[1:3] = [1\ \ 2\ \ 3]$ and
$[1:3\ \ 5] = [1\ \ 2\ \ 3\ \ 5]$

Thus, if the telescope number in an EDITDCTEL record is [1 : 3], the telescope parameter changes defined on the remainer of the record will apply to telescope numbers 1, 2, and 3. Similarly, if the telescope number is [4 5 : 8 10 11], the changes will apply to telescopes 4, 5, 6, 7, 8, 10,and 11.

3.3.2 DC Telescope Editing

The edit records normally are placed in the *arrayConfigure.cfg* file. These edit records apply to specific telescopes, not to standard telescopes. The telescope factories create the telescopes and then look for edit records specific to that telescope.

EDITDCTEL DC telescope edit record

<telescope number (matlab notation)>
<edit Flag1 >*FACET* only current option
<facet number (matlab notation) >
<edit Flag2 >*align* or *reflect* are current options

Parameter following *align*
<maximum misalignment (degrees)>

Parameters following *reflect*
<blur radius >
<mirror degradation factor >
<reflective curve (int) >

Examples:
EDITDCTEL [1:2] FACET [1:50] align 0.5
EDITDCTEL [5:10] FACET [10:100] reflect 0.2 0.95 2

3.3.3 SC Telescope Editing

The edit records normally are placed in the *arrayConfigure.cfg* file. These edit records apply to specific telescopes, not to standard telescopes. The telescope factories create the telescopes and then look for edit records specific to that telescope.

EDITSCTEL SC telescope edit record, no options implemented at present

3.4 Standard Telescopes

3.4.1 Davies-Cotton Telescopes

3.4.2 Schwarzchild-Coudee Telescopes

4 Data Files

4.1 Input Data Files

The photon Cherenkov file begins with a header followed by photon lines (see *GrOptics/Config/photon.cph*. The header file contains information passed from shower and photon production code. All header information must occur between the initial HEADF flag and the final DATAF flag which indicates the start of data records. The following data records are from the beginning of the *GrOptics/Config/photon.cph* file:

R 1.000000
H 1307.645000
S 0.80000 50.0 -50.0 -0.2500 0.4330 1307.6 -1000 -10777 -32656
P -1.4 79.9 -0.2519 0.4403 13509.0 239.0 534 3 2

P -0.2 79.6 -0.2518 0.4402 13531.4 237.5 498 3 2
The R record carries the photon thinning fraction.
The H record carries the observatory height in meters
The S record indicates the start of a new shower and contains information about the primary in this order:

1. Primary energy in TeV
2. x coordinate of the core (meters)
3. y coordinate of the core (meters)
4. x-direction cosine of the core
5. y-direction cosine of the core
6. observatory height (meters)
7. three negative random-number seeds

The P record contains the Cherenkov photon details as follows:

1. x-coordinate (meters) on the ground relative to an individual telescope at telescope level for CORSIKA or at ground level for KASCADE.
2. y-coordinate (meters) on the ground relative to an individual telescope at telescope level for CORSIKA or at ground level for KASCADE.
3. x-direction cosine in ground coordinates
4. y-direction cosine in ground coordinates
5. height (meters) of emission
6. relative time (nsecs) of emission
7. wavelength (nanometers)
8. particle type
9. telescope id number intercepting the photon

4.2 Output Data Trees

4.2.1 allTel Tree

The allTel tree contains information that is constant for all showers in the file. The tree branches are as follows:

1. fileHeader string
2. globalEffic globalEffic/D
3. obsHgt obsHgt/D
4. telIDVector vector¡int¿
5. telLocXVector vector¡float¿
6. telLocYVector vector¡float¿
7. telLocZVector vector¡float¿
8. transitTimeVector vector¡float¿

4.2.2 Individual Telescope Tree

Each telescope has its own tree. The branches are as follows:

1. eventNumber eventNumber/i

2. primaryType primaryType/i
3. primaryEnergy primaryEnergy/F
4. Xcore Xcore/F
5. Ycore Ycore/F
6. Xcos Xcos/F
7. Ycos Ycos/F
8. Xsource Xsource/F
9. Ysource Ysource/F
10. delay delay/F
11. photonX vector¡float¿
12. photonY vector¡float¿
13. time vector¡float¿
14. wavelength vector¡float¿

5 Graphical Output Options

References

[1] GrOptics git repository (read only)
git clone http://gtlib.gatech.edu/pub/IACT/GrOptics.git

[2] GrISU download site
http://www.physics.utah.edu/gammaray/GrISU/

[3] CORSIKA: A Monte Carlo Code to Simulate Extensive Air Showers
D. Heck, J. Knapp, J.N. Capdevielle, G. Schatz, T. Thouw
Forschungszentrum Karlsruhe Report FZKA 6019 (1998)

[4] CARE git repository (read only)
git clone http://gtlib.gatech.edu/pub/IACT/CARE.git

[5] Rene Brun and Fons Rademakers,
ROOT - An Object Oriented Data Analysis Framework,
Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996,
Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86.
See also http://root.cern.ch/drupal/

[6] Development of Non-sequential Ray-tracing Software for Cosmic-ray Telescopes Authors: Akira Okumura, Masaaki Hayashida, Hideaki Katagiri, Takayuki Saito, Vladimir Vassiliev. http://arxiv.org/abs/1110.4448 Download site http://sourceforge.net/projects/robast/