EVNDISP Manual
IACT event analysis and display
v4.10

Gernot Maier (DESY)
.....

October 26, 2013

Contents

Chapter 1

Documentation

EVNDISP is work-in-progress and the documentation is not in the state it is supposed to be. Apart from the information in this manual, other sources for help are:
A brief description of the analysis methods can be found on the following webpage:
`https://wiki-zeuthen.desy.de/CTA/Eventdisplay%20Software`

README files

INSTALL: information on installation the analysis package, dependencies, environmental variables

README.CTA: short description of a typical CTA analysis

README.VTS: description of a typical VERITAS analysis

AUTHORS: author description

Description and command line options for the different software parts:

README.EVNDISP

README.MSCW_ENERGY

README.ANASUM

README.EFFECTIVEAREA

README.ANALYSISLIBRARY

README.SCRIPTS

README.MACROS

WIKI pages for VERITAS users

The EVNDISP manual for VERITAS users:
`http://veritas.sao.arizona.edu/wiki/index.php/Eventdisplay_Manual`

Chapter 2

Introduction

eventdisplay is a complete package for VERITAS and CTA analysis. The package consists of several analysis steps and tools:

1. evndisp (calibrate and parametrize images, event reconstruction, stereo analysis)
2. evndisp (camera display for data and MC)
3. mscw_energy (use lookup tables to produce mean scaled width and lengths, and energies)
4. anasum (produce maps and calculate analysis results)
5. shared library tools and macros (e.g. produce the energy spectrum and integral fluxes, plot maps, plot sensitivities, etc.)
6. makeEffectiveArea (calculate effective areas)
7. trainTMVAforGammaHadronSeparation (tools to train MVA and optimize gamma/hadron separation cuts)
8. different tools to produce sensitivity file for CTA MC and Physics groups
9. ...

This is a very incomplete manual, started in September 2011. Please help updating it.
The original developers are Gernot Maier (DESY) and Jamie Holder (University of Delaware).
A large number of people contributed, see README/AUTHORS for a list.

Chapter 3

Installation and auxiliary data files

The installation, compilation, and needed or useful environmental variables are described in detail in the file *README/INSTALL* in the EVNDISP source code directory. It is strongly recommended to follow this scheme, as it simplifies the execution of the analysis significantly.

3.1 Auxiliary data files

The auxiliary data files contain information needed for the analysis. This might be files describing the detector geometry, lookup tables, effective areas file, etc. We assume in the following that these files are located in the directory *$OBS_EVNDISP_ANA_DIR* where *OBS* is your observatory (i.e. *VERITAS* or *CTA*).

3.1.1 Detector description

3.1.1.1 VERITAS

The detector description is read from a configuration file in GrIsu style. There are examples for the different detector configurations (e.g. pre or post-upgrade) in the tar ball with all the analysis files. Check the files in *$OBS_EVNDISP_ANA_DIR/*.cfg*.

3.1.1.2 CTA

No detector description is needed, since the telescope and array description is read directly from the DST file (*telconfig* tree). The converter from hessio to EVNDISP DST format needs a subarray file, a simple list of telescopes to be selected from the corresponding hyper array. The subarray files for prod1 can be found in *$OBS_EVNDISP_ANA_DIR/DetectorGeometry/*.lis*.

3.1.2 Analysis parameters files

Detailed description of these files can be found in the corresponding chapters of the different analysis stages.

3.1.3 Gamma/hadron separation parameters

Parameter files describing the details of the gamma/hadron separation can be found here: *$OBS_EVNDISP_ANA_DIR/GammaHadronCutFiles*

Chapter 4

eventdisplay - calibration, image analysis and stereo reconstruction

4.1 Trace integration

Several different trace integration methods are available. All settings should be in *$OBS_EVNDISP_ANA_DIR/ParameterFiles/EVNDISP.reconstruction.runparameter*.
For trace analysis of CTA MC events, the converter *CTA.convert_hessio_to_VDST* should be executed with the option '-f 1'.
Trace integration is done always twice: first with a short summation window to achieve optimal sensitivity, second a wider window to minimize the systematic errors:

```
 trace integration window <window 1> <window 2>
* -1 FADCSUMMATIONWINDOW 6 18
  trace integration start (for window 1) <sample>
  [optional: shift of window start for double pass]
* -1 FADCSUMMATIONSTART 2 -1
```

Note that here and in the following, a '-1' in the first column means that the corresponding values are applicable for all telescopes. Otherwise, values can be set individually for each telescope type.

4.1.1 Fixed window

```
 select trace integration method
* -1 FADCANALYSIS 1
 double pass method <doublepass (on=1, off=0)>
    <integration window length for pass 1>
   <integration method for pass 1 (0/1)>
* -1 FADCDOUBLEPASS 0
```

4.1.2 Doublepass method

To select the double pass method, the following settings are applicable:

```
 select trace integration method
* -1 FADCANALYSIS 1
 double pass method <doublepass (on=1, off=0)>
   <integration window length for pass 1>
   <integration method for pass 1 (0/1)>
* -1 FADCDOUBLEPASS 1 12 0
```

4.1.3 Sliding window method

The sliding window selects the maximum charge for the given integration window length.

```
 select trace integration method
* -1 FADCANALYSIS 1
```

4.1.4 No trace integration

```
 select trace integration method
* -1 FADCANALYSIS 0
```

4.2 Calibration

4.2.1 Pedestal calibration

4.2.1.1 Calculating pedestal and pedestal variations for CTA

These are the steps calculate pedestals and pedestal variations for prod-2 CTA simulations:

- produce a simtel_array file with NSB events only (as described here: ??)
- convert the sim_telarray file with *CTA.convert_hessio_to_VDST* to a dst.root file

```
./bin/evndisp -sourcefile LeoncitoLongTraces.dst.root -runmode=1 \\
-singlepedestalrootfile=1   -donotusepeds -usePedestalsInTimeSlices=0 \\
-calibrationsumwindow=30 -calibrationsumfirst=0 -ignoredstgains \\
-reconstructionparameter EVNDISP.prod2-noDoublepass.reconstruction.runparameter \\
-nevents=20 -nopedestalsintimeslices
```

The pedestal file is then used in the analysis via the settings in converter *CTA.convert_hessio_to_VDST*: '-c pedfile.root'

4.2.2 Low gain calibration

VERITAS only
For the calibration of the low-gain chain two values are important: the low-gain pedestal and the ratio between low and high-gain channel (usually between 5 and 6).

4.2.2.1 Low-gain pedestal calibration

Pedestal values of the low-gain chain are not sensitive to changes in the NSB as the high-gain chain. Therefore new calibration is only necessary after changes in the electronics, e.g. the after swapping a FADC board. For a list of good low-gain calibration runs and the valid time ranges look the VERITAS wiki page *Low gain calibration*. There is a list of documents linked to this page, valuable for background information.
Low-gain calibration runs are taken by raising the HV in part of the cameras - therefore a pair of flasher runs is needed for the calibration (exception is the first run, 36862). The calculation of the low-gain pedestal values consists of two steps: i) the calculation of the pedestal values and ii) merging these files into single pedestal files.
Step 1 (script is in $EVNDISPSYS/scripts/VTS)

```
VTS.EVNDISP.analyse_pedestal_events <telescope number> \\
                                <sourcefile> [runnumber] [lowgain]

use parameter lowgain=1 to calculate low-gain pedestals

    VTS.EVNDISP.analyse_pedestal_events -1 /d20110213/55083.cvbf 55083 1
```

Results are written into the calibration directory in $VERITAS_EVNDISP_ANA_DIR, files with suffixes *.lped* and *.lped.root*.
Step 2: merge files into a single low-gain pedestal file
Use the EVNDISP shared library and the class *VPedestalCombineLowGainFiles*:

```
root [0] .L $EVNDISPSYS/lib/libVAnaSum.so
root [1] VPedestalLowGain a;
root [2] a.combineLowGainPedestalFileForAllTelescopes( unsigned int iNTel, \\
string iCalibrationDirectory, \\
string iRun1, string iRun2, string iOutRun );
```

The list of run numbers and the corresponding valid ranges are listen in the file *calibrationlist.LowGain.dat* in the calibration direction in $VERITAS_EVNDISP_ANA_DIR:

```

LOWGAINPED <TELESCOPE> <RUNSTART> <RUNSTOP> <LOW GAIN PEDESTAL RUN NUMBER>
(all comparisons are >= and <=)

RUNS 55975 and 55978 (128 samples)
* LOWGAINPED -1 46642 60000 5597578
....
....
```

4.2.2.2 Low-gain multiplicator

4.3 Image cleaning

There are different methods available for image cleaning. All settings should be in
$OBS_EVNDISP_ANA_DIR/ParameterFiles/EVNDISP.reconstruction.runparameter.

4.3.1 Signal-to-noise cleaning

This is the default cleaning method for VERITAS analysis.

```
 image cleaning <method> <FIXED/VARIABLE>
* -1 IMAGECLEANINGMETHOD TWOLEVELCLEANING VARIABLE
 image cleaning thresholds <image> <border>
* -1 IMAGECLEANINGTHRESHOLDS 5. 2.5
```

4.3.2 Fixed threshold cleaning

4.3.3 Time cluster cleaning

4.3.4 Time next-neighbor cleaning

4.4 Image analysis

4.4.1 Image parameters

The image parameterization is based on the so-called Hillas parameters [Hillas(1985)]. They are derived by the zeroth order (amplitude or size), first order (center of gravity) and second order (length, width, orientation) moments of the elliptical image of the shower. In general, they are calculated using the formulas given in [Fegan(1997)].
The image parameters are stored per telescope in the *tpars* tree of the evndisp output file. For more information see:

- *inc/VImageParameter.h* - definition of the image parameters
- *src/VImageParameterCalculation.cpp* - calculation of the parameters

4.4.2 Log-likelihood fitting of images

To recover truncated images at the edge of the camera, a simple log-likelihood fitting algorithm is applied. The underlying assumption of the fitting method is that the image of a gamma-ray shower can be described by a two-dimensional normal distribution. Asymmetry is ignored to limit the number of fit parameters which are image centroid position, image width, image length and image size. Noise is estimated by assuming Poisson fluctuations. The starting values for the fit parameters are obtained from the standard image analysis. The following expression is minimized using the MIGRAD method of MINUIT [James(1998)]:

$$L = -\sum (n_i \ln S_i - S_i - n_i \ln n_i + n_i) \tag{4.1}$$

where n_i is the measured integrated charge for pixel i, and S_i is the estimated charge from the fit function.
The settings for the log-likelyhood fitting should be in
$OBS_EVNDISP_ANA_DIR/ParameterFiles/EVNDISP.reconstruction.runparameter.

```
 image calculation at the edge of the camera
              <minimun loss value> <minimum number of pixels>
* -1 LLEDGEFIT 0.1 2
```

The status of the covariance matrix is returned and saved to the image parameter tree *tpars* in the variable *Fitstat*. No fitting has been applied to images with $Fitstat = -1$

4.5 Stereo analysis

Chapter 5

Displaying events

eventdisplay can be used to display events. Camera images of integrated charges, timing and calibration values, image cleaning and parameterization, core and direction reconstruction results can be displayed. There are example scripts for display files, see

- for CTA: $EVNDISP/scripts/CTA/CTA.EVNDISP.display
- for VERITAS: $EVNDISP/scripts/VTS/VTS.EVNDISP.display

Note that the display can be a bit slow for arrays with a large number of telescopes.

Chapter 6

mscw_energy - using lookup tables

6.1 Energy reconstruction

Chapter 7

Gamma/hadron separation

7.1 Cut parameters

Option	Number of parameters	Allowed value(s)	Default value(s)	Description
cutselection	2			type of gamma/hadron and direction cut
	Parameter #1	?	0	gamma/hadron cut id
	Parameter #2	0-5	0	direction cut id: fixed Θ^2 cut (0, needs parameter *theta2cut*), energy dependent Θ^2 from a function read from a IRF file (1, needs parameter *theta2file*), from a IRF graph (2, needs parameter *theta2file* and option *IRF*), all other values: experimental (3-5, TMVA)
angres	1	$]0, 100]$	0	containment probability for energy dependent Θ^2 cut (in %)

Table 7.1: *Parameter definition and range for gamma/hadron cut files. This is used for example in the effective area calculation or for data analysis.*

Table 7.2: *Gamma/hadron cut selector values. They consist of two digits: ID1+ID2*10*

gamma/hadron cut selector	Description
ID2	
0	apply gamma/hadron cuts on parameters in given data tree
1	apply gamma/hadron cuts on probabilities given by a friend to the data tree (e.g. random forest analysis)
2	same as 1
3	apply cuts on probabilities given by a friend to the data tree already at the level of the event quality level (e.g. of use for analysis of certain binary phases only)
4	TMVA gamma/hadron separation
5	apply FROGS based gamma/hadron separation
ID1	
0	apply cuts on MSCW/MSCL (mean reduced scaled width/length)
1	apply cuts on mean width/length (no lookup tables necessary)
2	no cut applied (always passed)
3	apply cuts on MWR/MLR (mean scaled width/length)
Example:	
0	apply MSCW/MSCL cuts (default)
22	apply event probability cuts
10	apply cuts from a tree AND apply MSCW/MSCL cuts
40	use TMVA AND apply MSCW/MSCL cuts

7.2 Cut variables

7.2.0.1 nnimages: Minimum number of images (telescope type independent)

Name nnimages

Type Integer

Comparator *greater equal* ($\geq$)

Range 0 to maximum number of telescopes

Default 2

Example *nnimages 2* (at least 2 images)

Note that the *teltype_nnimages* cut can overwrite this cut (if *nnimages* < *teltype_nnimages*).

7.2.0.2 teltype_nnimages: Minimum number of images (telescope type dependent)

Name teltype_nnimages

Type Integer

Comparator *greater equal* ($\geq$)

Range 0 to number of telescope for the given telescope type

Default off

Example *teltype_nnimages 4 2 3* (at least 4 images for telescope type 2 and 3)

Note that the *nnimages* cut can overwrite this cut (if *nnimages* > *teltype_nnimages*).

7.2.0.3 corearea: cut on location of core position

```
corearea <distance to array center X_min> <distance to array center X_max>
         <distance to array center Y_min> <distance to array center Y_max>
         [optional: max allowed distance to edge of array]
```

Name corearea

Type Floats

Unit meter

Comparator *greater* (>)

Default off

Example (box cut on core): *corearea -300 300. -400. 400*

Example (edge cut only): *corearea -1.e10 e.1e10 -1.e10 1.e10 250.*

Chapter 8

makeEffectiveArea - instrument response functions

Instrument response functions (IRF), i.e. effective areas and angular, core and energy resolution and bias curve can be calculated using *makeEffectiveArea*.

Option	Number of parameters	Allowed value(s)	Default value(s)	Description
FILLINGMODE	1	0,1,2,3	0	filling of IRFs: fill all IRFs (0), resolution plots only (1), angular resolution plot only (2), effective areas only (3)
ENERGYRECONSTRUCTIONMETHOD	1	0,1	0	energy reconstruction method (see 6.1)
ENERGYAXISBINS	1	> 0	60	number of bins on $\log_{10}$ energy axis
ENERGYRECONSTRUCTIONQUALITY	1	0,1	0	
AZIMUTHBINS	1	0,1	1	define azimuth bins and calculate IRFs in each azimuth bin. Bins are hardwired with a bin width of 22.5° (16 bins), bin 17 contains the full azimuth range
ISOTROPICARRIVALDIRECTIONS	1	0,1	0	input MC are simulated with random direction (wobble) offsets (use for gamma rays only)
TELESCOPETYPECUTS	1	0,1	0	apply telescope type dependent cuts CHECK! STILL USEFUL?
FILLMONTECARLOHISTOS	1	0,1	0	fill histograms with MC spectra only (no IRF calculation)

ENERGYSPECTRUMINDEX	3			reweight events to this set of spectral indexes
	#1	> 0	1	number of different spectral indexes
	#2	$> 0.$	2.0	lower value
	#3	$> 0.$	0.1	step size
CUTFILE	1			cut file (full path, see 7.1)
SIMULATIONFILE_DATA	1			simulation data file (mscw file)
SIMULATIONFILE_MCHISTO	1			data file with thrown events. This can be either a full mscw file (slow) or a file with the filled MC histograms

Table 8.1: *Parameters in the run parameter file for the IRF calculation.*

Chapter 9

CTA analysis

9.1 General concept

We assume in the following that CTA simulations are produced and available in the *hessio* format. Eventdisplay does not depend on *hessio* and the reading of any future CTA data format could easily be implemented.
To use Eventdisplay for CTA analyses, the *simtel.gz* files have first to be converted into ROOT format (called in the following DST format) using the converter tool described below. The analysis then proceed in several steps. Note that the analysis must be done for each subarray separately, even if there are more telescopes in the simtel files (concept of hyper arrays). The following list shows the exectuables and macros needed to produce CTA sensitivity files:

CTA.convert_hessio_to_VDST convert simtel.gz files into ROOT format

eventdisplay FADC trace integration (if needed); image cleaning; calculation of image parameters; reconstruction of the direction and core; display

mscw_energy train and use lookup tables to estimate energies and mean scaled parameters (note that these are two separate steps)

trainTMVAforGammaHadronSeparation train MVA for gamma/hadron suppression (e.g. using boosted decision trees or box cuts)

makeEffectiveArea calculate effective areas and instrument response functions (e.g. angular and energy resolution; migration matrix, etc)

writeParticleRateFilesFromEffectiveAreas calculate event number spectra from gamma or background effective areas. This is used in the cut optimization

sensitivity.C ROOT macro to calculate signal and background events after pre-cuts and to plot sensitivity curves

writeCTAWPPhysSensitivityFiles write sensitivity files (CTA WP-PHYS style root file)

There are shell scripts to simplify these steps in the *$EVNDISP/scripts/CTA* directory (see description in section 9.2. They allow analyze a large amount of simulations files on a computing cluster[1]. It is easy with their help to analyze many simtel.gz files for several subarrays, offsets, cuts, and so on. The scripts expect several environmental variables to be set during execution time, see section 3 and the readme file *README/INSTALL* for details.

9.1.1 Array configuration - subarray file

Many of the CTA simulation files contain data for a so called hyper array (a large number of telescopes have been simulated, a sub-set of the telescope data is analyzed). To select the actual array, the corresponding telescopes have to be specified in a ASCII text file (called *subarray* file in the following). The example file below selects an array consisting of telescopes 63, 19, 67, and 33, each telescope with a field of view of 8 degrees (the FOV is reduced with respect to the simulated FOV by setting the corresponding pixels dead; for no FOV cut, set this value to a very large number):

```
63 8
19 8
67 8
33 8
```

Subarray files for most of the arrays used in the CTA sensitivities studies are part of the analysis file package (see 3.1). The subarray files for prod1 and prod2 can be found in *$OBS_EVNDISP_ANA_DIR/DetectorGeometry/*.lis*.
Note that a subarray file is always needed, even if all telescopes from the hessio file are read out.

9.1.2 Step 1: Converter

In this step a simulation data file in *hessio* format (*simtel.gz*) is converted to the EVNDISP DST format (see 10.2). These are the possible options to run the converter:

```
$EVNDISPSYS/bin/CTA.convert_hessio_to_VDST

CTA.convert_hessio_to_VDST: A program to convert hessio data to EVNDISP DST files (v.4.00)
==========================================================================================


Syntax: ./bin/CTA.convert_hessio_to_VDST [ options ] [ - | input_fname ... ]
Options:
   -v                 (More verbose output)
   -q                 (Much more quiet output)
   -s                 (Show data explained)
   -S                 (Show data explained, including raw data)
   --history (-h)     (Show contents of history data block)
   -i                 (Ignore unknown data block types)
   --max-events n     (Skip remaining data after so many triggered events.)
   -a subarray file   (list of telescopes to read with FOV.)
   -o dst filename    (name of dst output file)
   -f on=1/off=0      (write FADC samples to DST file;default=0)
   -r on=1/off=0      (apply camera plate scaling for DC telescopes; default=1)
   -d <nbits dyn.>    (dynamic range of readout (e.g. 12 for 12 bit. Switch to low gain)
```

[1]Most of these scripts work fine on the DESY batch system, they might need some adjustment for other computing environments

The following options are mandatory: -a (subarray file, see next paragraph) and -o (DST output file). Note the limitations of the DST format (10.2.1).
For a typical run, the following command line should be used:

```
./CTA.convert_hessio_to_VDST -a subArray.list -o dstfile.dst.root \
          gamma_run12241.simhess.gz
```

For FADC analysis, add the option *-f 1*. Note again the limitations of the DST format (10.2.1).

9.1.3 Step 2: Display (event-by-event)

It is always useful to look at events in the display. To do this, it is best to select a small subarray with the *-teltoana* option in evndisp. The display might otherwise be quite slow in responding to your input due to the large number of objects to be drawn.
A typical command line to look at events might be:

```
$EVNDISPSYS/bin/evndisp -display=1 \
-reconstructionparameter ./EVNDISP.reconstruction.runparameter \
-l2setspecialchannels nofile \
-sourcefile tt.v2.root
```

9.1.4 Step 3, Eventdisplay: FADC integration, calibration, image analysis and stereo reconstruction

section missing

9.1.5 Plotting effective areas in instrument response functions

Plotting the effective areas and instrument response functions using the shared library from evndisplay:

```
% root
root [0] .L $EVNDISPSYS/lib/libVAnaSum.so
root [1] VPlotInstrumentResponseFunction a;
root [2] a.addInstrumentResponseData("gamma_onSource.E_ID0.eff-0.root")
root [3] a.plotEffectiveArea()
root [4] a.plotAngularResolution()
root [5] a.plot...
```

9.2 Automized CTA analysis using scripts

All bash scripts described in the following can be found in *$EVNDISPSYS/scripts/CTA*.

9.2.1 Directory structure of analysis products and log files

The following scripts expect a certain directory structure for data products and log files. Following this structure simplifies the analysis significantly.

- different simulations sets are distinguished by their name. This name has to be chosen at the beginning of the analysis, and then used consistently through all steps (e.g. *cta-ultra3* or *ISDC3700m*). In the following, the placeholder **SIMDATASET** is used in all directory names and examples of command lines.
- analysis results (in ROOT format) are written to *$CTA_DATA_DIR/analysis/SIMDATASET*
- analysis log files to *$CTA_LOG_DIR/analysis/SIMDATASET*
- results for individual subarrays written in sub directories (e.g. *$CTA_DATA_DIR/analysis/SIMDATASET/E/*)
- the configuration files for the different subarrays are expected to be in *$OBS_EVNDISP_ANA_DIR/DetectorGeometry/*.

9.2.1.1 Runparameter files for scripts

The information on directory names, reconstruction methods etc are used in several scripts. To simplify this, some parameters are read from a run parameter file. An example can be found in *$OBS_EVNDISP_ANA_DIR/ParameterFiles/scriptsInput.runparameter*. Note that the values are read out by a simple *grep* command, so no repetitions are allowed.
This is work in progress

MSCWSUBDIRECTORY [string]: name of sub directory with products from the lookup table analysis

TMVASUBDIR [string]: name of sub directory where results from TMVA training are written

EFFAREASUBDIR [string]: name of sub directory for results from effective area calculation

RECID [integer]: quality cuts selector used in lookup table analysis (default is 0)

ENERGYRECONSTRUCTIONMETHOD [integer]: ID of energy reconstructed method

NIMAGESMIN [integer]: cut on minimum number of images applied in analysis (default is 2)

Example:

```
MSCWSUBDIRECTORY Analysis-NM2-ID0-d20120507
TMVASUBDIR BDT-NM2-ID0-ErecS-d20120510
EFFAREASUBDIR EffectiveArea-NM2-ID0-ErecS-d20120510

RECID 0
ENERGYRECONSTRUCTIONMETHOD 1
NIMAGESMIN 2
```

9.2.2 Off-axis analysis

The off-axis analysis (creating lookup tables and training the gamma-hadron separator at different camera offsets) is called *cone10*. The offset bins are hardwired in the bash scripts to the following values (mean and intervals):

```
OFFMIN=( 0.0 1.0 2.0 3.00 3.50 4.00 4.50 5.00 5.50 )
OFFMAX=( 1.0 2.0 3.0 3.50 4.00 4.50 5.00 5.50 6.00 )
OFFMEA=( 0.5 1.5 2.5 3.25 3.75 4.25 4.75 5.25 5.75 )    (mean value)
```

9.2.3 Step 1 & 3 combined: convert simtel file & run eventdisplay

Run the converter (simtel to DST) and eventdisplay analysis for a list of subarrays.

```
./CTA.EVNDISP.sub_convert_and_analyse_MC_VDST.sh

 <sub array list>          text file with list of subarray IDs

 <list of simtelarray files>

 <particle>                gamma_onSource , gamma_cone10, proton , ...

 <data set>                e.g. cta-ultra3, ISDC3700m, ...

<0/1>  [1 = keep DST.root files (default off=0=delete DST.root files after analysis)]
```

NOTE: The image cleaning thresholds can be specified in the file
$OBS_EVNDISP_ANA_DIR/ParameterFiles/EVNDISP.reconstruction.runparameter
file either for all telescopes to the same values or for each telescope type separately (see section XXXX).

Input Simulation file from simtel.
Parameter files for image and array analysis:
$OBS_EVNDISP_ANA_DIR/ParameterFiles/EVNDISP.reconstruction.runparameter

Output One ROOT and one log file for each simtel file with results from image and array analysis in e.g.
$CTA_DATA_DIR/analysis/SIMDATASET/gamma_onSource/ and
$CTA_LOG_DIR/analysis/SIMDATASET/gamma_onSource/

9.2.4 Step 4: mscw_energy

9.2.4.1 Creating lookup tables

Lookup tables are used for mean scaled with, length and energy reconstruction.

```
CTA.MSCW_ENERGY.sub_make_tables.sh <table file name> <recid> <subarray list> \
<onSource/cone10> <data set>

  <table file name>  name of the table file (to be written; without .root)
```

```
  <recid>            reconstruction ID according to EVNDISP.reconstruction.parameter
  <subarray list>    text file with list of subarray IDs
  <onSource/cone10>    calculate tables for on source or different wobble offsets
  <data set>         e.g. cta-ultra3, ISDC3700m, ...
```

Input eventdisplay ROOT files from previous analysis step.

Output One lookup table files (ROOT format) to be found in
$CTA_DATA_DIR/analysis/SIMDATASET/Tables/.
One log file in
$CTA_LOG_DIR/analysis/SIMDATASET/Tables/.

9.2.4.2 Off-axis lookup tables

The off-axis tables produced in different wobble offset bins have to be combined into a single file before any further steps:

```
CTA.MSCW_ENERGY.combine_tables.sh <combined table file name> <subarray list> \
<input table file name> <output directory> <data set>

  <combined table file name>  name of the table combined file (without .root)

  <subarray list>             text file with list of subarray IDs

  <input table file name>     name of the input table name (beginning of...)

  <output directory>          directory for combined tables
```

Copy or move the created tables to your directory $OBS_EVNDISP_ANA_DIR/Tables/

9.2.4.3 Using lookup tables

Run mscw_energy for estimating the energy of each event:

```
CTA.MSCW_ENERGY.sub_analyse_MC.sh <tablefile> <recid> <subarray list> <particle> \
<data set> [wildcard]

  <tablefile>      table file name (without .root)
                   expected file name: xxxxxx-SUBARRAY.root; \
                     SUBARRAY is added by this script
  <recid>          reconstruction ID
  <subarraylist >  text file with list of subarray IDs
  <particle>       gamma_onSource / gamma_cone10 / electron / proton / helium
  <data set>       e.g. ultra, ISDC3700m, ...

optional (for a huge amount of MC files):
  [wildcard]     used in the < CTA.MSCW_ENERGY.subParallel_analyse_MC.sh > script
```

To submit jobs for different particle types, use
$EVNDISPSYS/scripts/CTA/CTA.subAllParticles_analyse_MC.sh.

Input Lookup tables
eventdisplay ROOT files from previous analysis step.

Output One or several ROOT files for each particle type in
$CTA_DATA_DIR/analysis/SIMDATASET/Analysis/.
Log files will be in
$CTA_LOG_DIR/analysis/SIMDATASET/Analysis/.
You can combine many ROOT files from the first analysis step (converter+eventdisplay) into a single large ROOT file.

9.2.5 Step 5: Optimize cuts or train MVA

Gamma-hadron separation is based on TMVA and needs to be trained for each subarray due to their different layouts. Here an example how to do this for Boosted Decicion Trees (BDT). Input to the training are the files produced during the lookup table analysis (containg shower direction and core, MSCW, MSCL, reconstructed energy, etc.).

```
CTA.TMVA.sub_train.sh <subarray list> <onSource/cone10> <data set> <analysis parameter file>

  <subarray list>   text file with list of subarray IDs

  <onSource/cone10>    calculate tables for on source or different wobble offsets

  <data set>          e.g. cta-ultra3, ISDC3700, ...

   note 1: keywords ENERGYBINS and OUTPUTFILE are ignored in the runparameter file

   note 2: energy and wobble offset bins are hardwired in this scripts

   note 3: adjust h_cpu depending on your MVA method

   note 4: default TMVA parameter file is
   \$CTA\_EVNDISP\_ANA\_DIR/ParameterFiles/TMVA.BDT.runparameter
```

Note that for off-axis training (parameter *cone10*), one MVA is produced per energy and off-axis bin.

Input Products from lookup analysis for gamma rays, protons and electrons.
TMVA run parameter from e.g.
$OBS_EVNDISP_ANA_DIR/ParameterFiles/TMVA.BDT.runparameter
run parameter file for scripts (see section 9.2.1.1)

Output TMVA output files (ROOT and XML format) and log files are all written to
$CTA_DATA_DIR/analysis/SIMDATASET/TMVA/subDir/.
Use the TMVA macros delivered with ROOT to check the gamma/hadron separation and training results (can be found in *$ROOTSYS/tmva/test/TMVAGui.C*.

9.2.6 Step 6: Effective area calculation

Effective areas after optimal gamma-hadrons cuts are calculated in the following in three consecutive steps. Cuts on the MVA variable are optimized 'on the fly', depending on the energy spectrum of the source, the source strength and the observation time. First the typical angular resolution for the given array is calculated, then the signal and background rates after quality cuts, and finally, the optimal MVA cut value.
The Θ^2-cut (cut on angular direction) is always the 80% containment radius of the reconstructed with respect to the true direction. This might not the optimal choice, but it is probably reasonable for most point-source analysis.
Note that the effective area calculation includes the determination of all instrument response functions (i.e. angular and energy resolution, migration matrix, etc).

```
CTA.EFFAREA.sub_analyse.sh <subarray> <recid> <particle> <cutfile template>
<scripts input parameter file> <outputsubdirectory> <data set> [filling mode]


 calculate effective areas and instrument response functions for CTA


<subarray>
     subarray for analysis (e.g. array E)

<recid>
     reconstruction ID from array reconstruction (mscw stage)

<particle>
     allowed particle types are:
     gamma_onSource / gamma_cone10 / electron / electron_onSource / proton / proton_onSource

<cutfile template>
     template for gamma/hadron cut file (full path and file name)
     (examples can be found in /Users/maierg/Experiments/VERITAS/Data/AnalysisData-VTS-v400//

<scripts parameter file>
     file with analysis parameter
     see e.g. CTA_EVNDISP_ANA_DIR/ParameterFiles/scriptsInput.runparameter

<outputsubdirectory>
     directory with all result and log files (full path)

  <data set>         e.g. cta-ultra3, ISDC3700m, ...

[filling mode]
     effective area filling mode (use 2 to calculate angular resolution only)
     (optional, for default full calculation no option is needed)
```

Use *./CTA.EFFAREA.subAllParticle_analyse.sh* to submit the jobs for all particle types.

9.2.6.1 Directory structure for effective area calculation

The steps of calculating effective areas and optimizing cuts depend on the following directory structure:

- Results from angular resolution calculation:
  $CTA_USER_DATA_DIR/analysis/SIMDATASET/EffectiveArea/AngularResolution/

- Effective areas after quality ctus:
  $CTA_USER_DATA_DIR/analysis/EffectiveArea/SIMDATASET/QuallityCuts

9.2.6.2 Determination of angular resolution (80% containment)

The angular resolution is a weak function of the MVA cut parameter. Therefore a MVA cut value corresponding to a fixed signal efficiency (20%) in the following as applied.

```
$EVNDISPSYS/scripts/CTA/CTA.EFFAREA.subAllParticles_analyse.sh \
subArray.list $CTA_EVNDISP_ANA_DIR/ParameterFiles \
ANASUM.GammaHadron.TMVAFixedSignal.gamma.dat \
$CTA_USER_DATA_DIR/analysis/EffectiveArea/SIMDATASET/AngularResolution \
<DataSet> 2
```

Input Products from lookup analysis for gamma rays, protons and electrons.
TMVA XML and root files with classifier data.
Cut parameter file:
$OBS_EVNDISP_ANA_DIR/ParameterFiles/ANASUM.GammaHadron.TMVAFixedSignal
run parameter file for scripts (see section 9.2.1.1)

Output One ROOT file per subarray in the directory
$CTA_USER_DATA_DIR/analysis/EffectiveArea/SIMDATASET/AngularResolution
To plot the angular resolution vs energy, follow the instructions described here in Section 9.1.5.

9.2.6.3 Effective areas after quality cuts

This step is necessary for the determination of number of events after quality cuts. The effective areas after some loose quality cuts are determined, then folded with given spectra (e.g. Crab Nebula spectrum and CR proton and electron spectrum) followed by the calculation of the number of events expected for a certain observation period.
These are typical loose cuts applied in this step:

- successful reconstruction
- Θ^2 cut as described earlier (80% containment radius, energy dependent)
- $-2 < MSCW < 2$ and $-2 < MSCL < 5$

The cuts can be changed in the cut file, the default file is
$OBS_EVNDISP_ANA_DIR/ParameterFiles/ANASUM.GammaHadron.QC

To calculate effective areas after quality cuts do:

```
$EVNDISPSYS/scripts/CTA/CTA.EFFAREA.subAllParticles_analyse.sh \
subArray.list $OBS_EVNDISP_ANA_DIR/ParameterFiles \
ANASUM.GammaHadron.QC \
$CTA_USER_DATA_DIR/analysis/EffectiveArea/<DataSet>/QualityCuts \
<DataSet>
```

Input Products from lookup analysis for gamma rays, protons and electrons.
Cut parameter file:
$OBS_EVNDISP_ANA_DIR/ParameterFiles/ANASUM.GammaHadron.QC
run parameter file for scripts (see section 9.2.1.1)

Output One ROOT file per subarray in the directory
$CTA_USER_DATA_DIR/analysis/EffectiveArea/SIMDATASET/QualityCuts
To plot the effective areas vs energy, follow the instructions described here in Section 9.1.5.

9.2.6.4 Calculation of number of signal and background events

To calculate the number of signal and background events, which is needed to find the optimal cuts:

```
./CTA.ParticleRateWriter.sub.sh <sub array list> <directory with effective areas>
                                 <offset=onSource/cone10> <recid>

  write particles files needed for TMVA cut optimization

  <sub array list>          text file with list of subarray IDs

  <directory with effective areas>  (full) path to effective areas

  <recid>                   reconstruction ID from mscw stage
```

This script is using the binary *bin/writeParticleRateFilesFromEffectiveAreas*.
Alternatively, this can be done with the following commands by hand:

```
% cd $CTA_USER_DATA_DIR/analysis/EffectiveArea/<DataSet>/QualityCuts
% root
root [0] .L $EVNDISPSYS/lib/libVAnaSum.so
root [1] .L $EVNDISPSYS/macros/sensitivity.C
root [2] writeAllParticleNumberFiles( "subArray.list", 0 );
// or for several (e.g. 8) offsets:
root [2] writeAllParticleNumberFiles( "subArray.list", 8 );
```

Input Effective area files after quality cuts (one ROOT file per sub array).
Energy spectra for the Crab Nebula and proton and electron cosmic rays spectra are read from
$OBS_EVNDISP_ANA_DIR/AstroData/TeV_data/EnergySpectrum_literatureValues_CrabNebula.dat
$OBS_EVNDISP_ANA_DIR/AstroData/TeV_data/EnergySpectrum_literatureValues_CR.dat
run parameter file for scripts (see section 9.2.1.1)

Output One ROOT file per subarray with two graphs: number of signal/number of background events per energy bin vs energy.

9.2.6.5 Effective areas after gamma/hadron separation cuts

Calculate effective areas while applying the TMVA-based gamma-hadron cuts trained before. Example:

```
$EVNDISPSYS/scripts/CTA/CTA.EFFAREA.subAllParticles_analyse.sh \
subArray.list $CTA_EVNDISP_ANA_DIR/ParameterFiles \
ANASUM.GammaHadron.TMVA \
$CTA_USER_DATA_DIR/analysis/EffectiveArea/<DataSet>/TMVA/BDT.20120321 \
<DataSet>
```

Input Products from lookup analysis for gamma rays, protons and electrons.
TMVA XML and root files with classifier data.
Cut parameter file:
$OBS_EVNDISP_ANA_DIR/ParameterFiles/ANASUM.GammaHadron.TMVA
run parameter file for scripts (see section 9.2.1.1)

Output One effective area ROOT file per subarray with effective areas, angular and energy resolution, migration matrixes, etc.

9.2.7 Step 7: Sensitivity curves & WP-Phys files

Calculates sensitivity curves and writes these together with instrument response histograms into ROOT files following the WP Phys format.

```
$EVNDISPSYS/scripts/CTA/CTA.WPPhysWriter.sh <sub array list> \
<directory with effective areas> <observation time [h]> \
<output file name> <offset=0/1>
```

Some sensitivity related plotting functions can be found in the root macro $EVNDISPSYS/-macros/sensitivity.C

Input Effective area file after gamma/hadron separation cuts

Output One root file per subarray with sensitivities and instrument response function histograms.

Chapter 10

Input data format

10.1 VBF - VERITAS Bank Format

10.2 DST - data summary tree

The DST format is a simple ROOT tree containing standard C++ variables only (no class data).

10.2.1 Limitations

The implementation requires the hardwiring of the maximum number of telescopes, channels, etc. These values can be found in `inc`/VGlobalRunParameter, for example:

```
// HARDWIRED MAXIMUM NUMBER OF TELESCOPES AND CHANNELS, etc.
// maximum number of telescopes
#define VDST_MAXTELESCOPES   100
// maximum number of channels per telescopes
#define VDST_MAXCHANNELS     12000
// maximum number of summation windows
// (=maximum number of samples per FADC trace)
#define VDST_MAXSUMWINDOW    64
// maximum number of time slices for pedestal calculation
#define VDST_PEDTIMESLICES 5000
// maximum number of arrayreconstruction method
#define VDST_MAXRECMETHODS   100
// maximum number of timing levels
#define VDST_MAXTIMINGLEVELS 10
```

NOTE: These numbers determine the memory requirements of *evndisp* and *CTA.convert_hessio_to_VDST*.
NOTE: *evndisp* must be compiled with the same settings as the writing program.

Chapter 11

Detector Setup

Detector setup is written into a ROOT tree name *telconfig*, which is written to the *eventdisplay* and *mscw_energy* output files.

11.1 Telescope types

Different telescope types (e.g. mid-size and small-size telescopes, telescopes with different FOV, etc) are assigned a telescope type number in the code, this number is as well written to the data trees. The telescope type contains the mirror shape (DC, Parabolic, SC), the mirror area (m^2), the field of view ([deg]) and the pixel size ([deg]). For VERITAS, the telescope type correspond simply to the different telescope numbers (and are therefore 0,1,2,3).
For clarification, this is the corresponding code bit from src/CTA.convert_hessio_to_VDST.cpp:

```
fTelescope_type  = TMath::Nint(pix_size*100.);
fTelescope_type += TMath::Nint(fFOV*10.)*100;
fTelescope_type += TMath::Nint(fMirrorArea)*100*10*100;
// all large telescopes are parabolic, all others are Davies-Cotton (hardwired)
if( fMirrorArea > fParabolic_mirrorArea )      fTelescope_type += 100000000;
// Schwarzschild-Couder: check number of mirrors
else if( fNMirrors == fSC_number_of_mirrors )  fTelescope_type += 200000000;
```

Note: There is currently no way to determine the mirror/telescope shape (parabolic, Davies-Cotton, etc) from the hessio file. This is why the mirror area and the number of mirrors is used. The parabolic shape is assigned to all telescopes with a mirror area > 400 m^2. Schwarzschild-Couder Design are all telescopes with 2 mirrors only.

Chapter 12

Tools for VERITAS analysis

12.1 Exposure maps and run lists for any point in the sky

This small tools allows you to

- plot exposure maps in Galactic coordinates for VERITAS (with and without radial acceptance)
- get a list of data runs for a given direction in the sky
- get a list of data runs for objects from a catalogue
- print a LaTex-style table for the selected list of objects with exposure, zenith angles, observing angle, etc.

Usage example:

```
// load shared library
.L lib/libVAnaSum.so
VExposure f;
// read VERITAS db for a given period
// (only necessary if no root file with
// exposures is available)
f.setTimeRange("20070901", "20071231" );
f.readFromDB();
// the results can be saved to a root file (fast access)
f.writeRootFile( "myrootfile.root" );
// and later retrieved with
f.readRootFile("myrootfile.root" );
// for radial acceptance correction an acceptance curve is needed
f.readAcceptanceCurveFromFile("myacceptance.root" );
// fill maps: (this may take a while)
f.fillExposureMap();
// plot the maps for a given l, b range
f.plot(-10., 10., 60, 100., "" );
// plot the maps with some objects from a catalogue:
// (there are some examples of catalogues in the directory
// with the auxiliary files:
// AstroData/Catalogues/: tevcat.dat, ...
f.plot(-10., 10., 60, 100., "mycatalogue.dat" );
```

```
// print list of runs of data taken e.g. around the Galactic centre (5 deg circle)
f.printListOfRuns( 0., 0., 5. );
// print list of runs around the objects in the given catalogue
f.printListOfRuns( "mycatalogue.dat" );
```

12.2 Radiosonde Atmospheric Data

Download radiosonde balloon data from Tucson airport using a wget script and create a root file of the data.
Use the download script, for example get data for the entire year of 2010:

```
cd $EVNDISPSYS/scripts/VTS/
./VTS.downloadSoundingDatafromUWYO.sh 2010
```

Combine the monthly data into one file and create a list of files (in this case just the total file):

```
cat sounding_2010* > all_2010.dat
ls all_2010.dat > list_2010.dat
```

Use a root file of radiosonde data and plot results

```
root -l
.L ../../../../sharedLib/libVAnaSum.so
VAtmosphereSoundings a;
a.readSoundingsFromTextFile("list_2010.dat");
a.writeRootFile("all_2010.root");
.q
```

Use the VAtmosphereSoundings class to plot the data:

```
root -l
gROOT->SetStyle("Plain");
gStyle->SetCanvasBorderMode(0);
gStyle->SetPadBorderMode(0);
gStyle->SetPalette(1);
.L ../../../../sharedLib/libVAnaSum.so
VAtmosphereSoundings a( "all_2010.root");
a.plotAverages(2010,1,2010,12);
a.plot2DProfiles(2010,1,2010,12);
.q
```

- please read through $EVNDISPSYS/src/VAtmosphereSoundings.cpp for full analysis details
- for a detailed overview of the results please see: Atmosphere
- radiosonde data is from http://weather.uwyo.edu/upperair/sounding.html

There is a root macro in *$EVNDISPSYS/macros/VTS/atmosphericSounding.C* to plot monthly/yearly average and compare with atmospheric profiles from simulations.

Chapter 13

Light curve analysis

Several methods for light curve analysis are currently under development. We describe in the following the existing tools, for further details please check the different sections in the code. Light curve data can be read from evndisplay result files (*anasum* files) or from ascii files.

13.1 Code organization

- *VLightCurveData:* data class (contains a single point of the light curve)
- *VLightCurveUtilities:* basic functions to read ascii files, print light curves, and get light curve properties (e.g. mean, variance)
- *VLightCurve:* light curve reader and plotter for ascii and evndisplay result files (*anasum* files)
- *VLombScargle:* discrete Fourier transform for unevenly spaced data after Lomb and Scargle
- *VZDCF:* plotting class for Z-transformed discrete correlation functions

13.2 Light curve plotting

Example for plotting light curves using an evndisplay result files (*anasum* file:

```
.L $EVNDISPSYS/lib/libVAnaSum.so
VLightCurve *iLightCurve = new VLightCurve();
iLightCurve->setName( "VHE light curve" );
// iDayInterval = 15. e.g. 15 day binning
// iMJDmin and iMJDmax to restrict light curve range in MJD
iLightCurve->initializeTeVLightCurve( "myanasumfile.root",
                          iDayInterval, iMJDmin, iMJDmax );
// plot points above 2 sigma or at least 3 on events as flux points,
// all others as upper flux limits
iLightCurve->setSignificanceParameters( 2., 3. );
// for binaries: phase folding
iLightCurve->setPhaseFoldingValues( PhaseMJD0, PhaseOrbitdays );
// fill light curve with fluxes above 500 GeV
iLightCurve->fill( 0.5 );
```

```
// plot light curve
iLightCurve->plotLightCurve();
```

Text file example (observation date (MJD), length of observation (in days), flux and flux error):

```
54857.173780    0.074748    1.27275    0.17745
54858.178508    0.074864    1.27704    0.17042
54859.139628    0.101489    1.30742    0.18451
54860.154963    0.105847    1.31362    0.17415
54867.274058    0.262490    1.09824    0.15957
```

Note the first two columns can be as well: begin and end of observations in MJD.
Example for plotting light curves using a simple text file:

```
.L $EVNDISPSYS/lib/libVAnaSum.so
VLightCurve b;
// plot 95\% upper flux limits for points with
// significances $<2 \sigma$:
b.setSignificanceParameters( 2., -9999., 0.95 );
// set spectral parameters assumed in the flux calculation:
b.setSpectralParameters( 0., 1., -2.5 );
b.initializeXRTLightCurve("mylightcurve.txt");
b.setPlottingStyle( 2, 1, 1., 20, 1. );
b.setLightCurveAxis( 0., 2., "counting rate" );
b.plotLightCurve();
```

Additionally to the described functions, there are several functions to fill gaps in light curves. This is work in progress and should be used only after carefully reading of the code.
VLightCurve and *VLightCurveUtillities* provide several methods to print details of the calculations to the screen and fill Latex and Wiki tables.

13.3 Lomb Scargle analysis

The discrete Fourier transform for unevenly spaced data after Lomb and Scargle is implemented in the *VLombScargle* class [Scargle(1982)]. The basic light curve reader classes can be used to read in a light curve from a text or eventdisplay result file.
There are two different implementations for the calculation of the significances of the peaks:

1. the calculation provided by Lomb & Scargle taking into account the number of independent frequencies scanned and Poissonian errors

2. a toy MC based method: the light curve is randomly shuffled N times with the flux points changed randomly according to their errors. For each of the resulting light curve the periodigram is calculated. The probability/significance is derived from the distribution of powers at each frequency bin (note: a large number of toy light curves have to be produced for larger values of significance).

Example:

```
.L $EVNDISPSYS/lib/libVAnaSum.so
 VLombScargle g;
g.readASCIIFile("mylightcurve.txt");
// scan 1000 frequencies between 50. and 1000. days
g.setFrequencyRange(1000., 1./1000., 1./50. );
g.plotPeriodigram();
// plot a line at the give frequency
g.plotFrequencyLine(1./315. );
// plot probability levels using the Lomb \& Scargle calculation
g.plotProbabilityLevels();
// plot probability levels using a toy MC with 500 MC light curve realisations
g.plotProbabilityLevelsFromToyMC( 500);
```

13.4 ZDCF Autocorrelation analysis

No autocorrelation analysis is implemented yet. We used until now the Z-transformed discrete correlation functions and the code provided by the authors of ZDCF[1].
There is a small class in eventdisplay provided to plot the ZDCF results, see the following example:

```
.L $EVNDISPSYS/lib/libVAnaSum.so
VZDCF a;
a.readZDCF( "XRT20120216.dcf");
a.setMLinterval( 315., 315.+6., 315.-3.86 ); // error provided by plike
a.plotZDCFoverError( 0, 45., 405., 12. );
a.plotZDCF( 0, 45., 405. );
```

[1] http://www.weizmann.ac.il/weizsites/tal/research/software/

Bibliography

[Fegan(1997)] Fegan, D. J., 1997, Journal of Physics G Nuclear Physics, 23, 1013

[Hillas(1985)] Hillas, A. M., 1985, in Proc 19th ICRC, La Jolla

[James(1998)] James, F., 1998, CERN Program Library Long Writeup D506

[Scargle(1982)] Scargle, J. 1982, ApJ 263, 835