EVNDISP Manual
IACT event analysis and display
v4.00

Gernot Maier (DESY)

.....

SVN $Rev : 660$

SVN $Date : 2012 - 03 - 2112 : 31 : 20 + 0100 (Wed, 21 Mar 2012)$

SVN $Author : maierg$

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:

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

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 (whatever 'complete' means...) The package consists of several analysis steps and tools:

1. evndisp (calibrate and parametrize images, event reconstruction, stereo analysis)
2. mscw_energy (use lookup tables to produce msw, msl, and energies)
3. anasum (produce maps and calculate analysis results)
4. shared library tools and macros (produce the energy spectrum and integral fluxes, plot maps, plot sensitivities, etc.)
5. makeEffectiveArea (calculate effective areas)
6. trainTMVAforGammaHadronSeparation (tools to train MVA and optimize cuts)
7. ...

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, among others: ..

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.

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

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*.

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.

Chapter 4

eventdisplay - calibration, image analysis and stereo reconstruction

4.1 Trace integration

4.2 Image analysis

4.2.1 Log-likelihood fitting of images

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 $Fitstat = -1$

4.3 Image cleaning

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.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 2
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
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

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

To use Eventdisplay for CTA analyses, the *simtel.gz* files have first to be converted into ROOT format (called in the following DST format). This must be done for each subarray separately. The following list shows the binaries and macros needed to produce CTA sensitivity files:

CTA.convert_hessio_to_VDST convert simtel.gz files into ROOT format

eventdisplay image cleaning & calculation of telescope parameters & reconstruction of the direction and core; display

mscw_energy train and use lookup tables to estimate the energy and mean scaled parameters

trainTMVAforGammaHadronSeparation optimize cuts or train MVA

makeEffectiveArea make effective areas

$EVNDISPSYS/macros/sensitivity.C calculate and plot sensitivity curves

writeCTAWPPhysSensitivityFiles write sensitivity files (WP-Phys style root file)

There are scripts to simplify these steps in the $EVNDISP/scripts/CTA directory. It is easy with their help to analyze many simtel.gz files for several subarrays, offsets, and so on. The scripts expect several environmental variables to be set (see 3). Most of the scripts work fine on the DESY batch system, they might need some adjustment for other computing environments.

9.2 Analysis steps for CTA IRF and sensitivity calculation

9.2.1 Step 1: Converter

In this step a simulation data file in hessio format 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

c_DST: 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 place scaling for DC telescopes; default=1)
```

The following options are necessary: -a and -o. Note the limitations of the DST format (10.2.1).
Many of the CTA simulation files contain data for a so called hyper array. To select the actual array, the corresponding subarray has to be specified in a ASCII text file (called *subarray* file in the following). The following example file selects an array consisting of telescopes 63, 19, 67, and 33, each telescope with a field of view of 8 degrees (the FOV can be reduced compared to the simulated FOV by setting the corresponding pixels dead):

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

Subarray files for most of the typical arrays used in the CTA sensitivities studies are part of the analysis file package (see 3.1). Note that a subarray file is always needed, even if all telescopes from the hessio file are read out and analyzed.
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).

9.2.2 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
```

9.2.3 Step 3, Eventdisplay: Calibration, image analysis and stereo reconstruction

9.2.4 Step 1 & 3 combined [USE]

Run the converter and Eventdisplay for a specific subarray: use script (in $EVNDISPSYS/scripts/CTA)

```
./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, ...
```

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 seperately (see section XXXX).

9.2.5 Step 4: mscw_energy

If you use standard configurations maybe some lookup tables already exist (ask Gernot or Heike where you could find them). If not, you have to create them from gamma-ray simulations yourself with $EVNDISPSYS/scripts/CTA/CTA.MSCW_ENERGY.sub_make_tables.sh
Copy or move the created tables to your directory $CTA_EVNDISP_ANA_DIR/Tables/
Now you have to run mscw_energy for estimating the energy of each event:
$EVNDISPSYS/scripts/CTA/CTA.MSCW_ENERGY.sub_analyse_MC.sh
For all different particle types, use $EVNDISPSYS/scripts/CTA/CTA.subAllParticles_analyse_MC.sh

9.2.6 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):

```
$EVNDISPSYS/scripts/CTA/CTA.TMVA.sub_train.sh subArray.list \
TMVA.BDT.runparameter BDT.20120321 BDT onSource ISDC3700m
```

9.2.7 Step 6: Effective Areas

For using TMVA based gamma-hadron cuts, you have to run through several steps now. The reason for this is that the cuts on the MVA variable is 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.
All scripts needed can be found in
$EVNDISPSYS/scripts/CTA/CTA.EFFAREA.sub_analyse.sh

Angular resolution:

Calculate the angular resolution from gamma-ray simulations

```
$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/<DataSet>/AngularResolution \
<DataSet> 2
```

Quality cuts:

Apply quality cuts and calculate the particle numbers using a root macro.

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

```
% 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 );
```

Make effective areas:

Make effective area while applying the TMVA-based gamma-hadron cuts trained before.

```
$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>
```

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...
```

9.2.8 Step 7: Sensitivity curves & WP-Phys files

Writing the IRF and sensitivity curves in the agreed 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>
```

This can then be used as well to plot all the sensitivities using the root macro $EVNDISPSYS/macros/sensitivity.C

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

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

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.

12.1 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

12.2 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.

12.3 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);
```

12.4 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

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