A new lookup scheme for `hap`

Henning Gast

6th March 2012

Contents

1 Introduction **2**

2 Lookups in the `hap` analysis **3**
2.1 Shape variables 3
2.2 Energy reconstruction 5
2.3 Energy resolution 6
2.4 Effective area 7
2.5 Point spread function 9
2.6 Extracting the PSF 10
2.7 Radial acceptance 11
2.8 Examples and tests 12

3 Technical issues **12**
3.1 Dealing with changing optical efficiencies 12
3.2 Azimuth interpolation 14
3.3 Smoothing 14
3.4 Uncertainties on interpolated values 14

4 The powerhorse: InstrumentResponse **15**

5 Filling lookups **16**

6 Open issues **19**

A Dealing with raw Monte Carlo simulations **20**

B Producing lookups for Heidelberg TMVA configurations **21**

1 Introduction

Lookups play a key role in the analysis of H.E.S.S. data. They are used to extract parameters describing the performance of the instrument that are obtained from Monte Carlo simulations for the most cases. The radial acceptance curves are an exception. They are determined from OFF runs containing no strong γ-ray sources instead.
Internally, lookups are stored as a set of ROOT histograms, with each histogram corresponding to a given set of parameters. When a lookup is queried for a certain parameter set $\vec{x}$, the neighbouring histograms, belonging to the parameter sets closest to $\vec{x}$ are retrieved and a suitable linear interpolation is performed.
Prior to the development of the lookup scheme described in this document, there was no generic structure for lookups in the framework of `hap`, but lookup classes were implemented in a rather inflexible way, only allowing queries with a limited number of parameters. In addition, users of `hap` had to split their run lists by the number of telescopes in a run and manually indicate the correct azimuth (north/south) and optical efficiency configuration (phase 1/1b) for the analysis. The filling of lookups was not integrated in `hap` and was not well documented. The recent refurbishment of H.E.S.S. mirrors lead to quickly changing configurations in terms of optical efficiencies. And with the addition of HESS-II, the energy threshold of the array will be lowered, mandating careful treatment of the effects of the geomagnetic field on low-energy showers and therefore of the dependence of shower parameters on the azimuth angle.
In order to cope with these issues, a new lookup scheme offering much more flexibility was developed and implemented. The core class for storing, retrieving and interpolating information from lookups is the `InstrumentResponse` (IR), and the new lookup scheme is therefore called the IR-based lookup scheme. It comprises `Maker` classes and utility programs for filling lookups, all conveniently controlled by a single script, user-friendly classes for querying lookups, as well as small test scripts for plotting information stored in lookups.
To summarize, the new IR-based lookup scheme was created with the following goals in mind:

- Make `hap` analysis more user-friendly and flexible.
- Cope with quickly changing configurations in terms of optical efficiency.
- Prepare for H.E.S.S.-II analysis.
- Transparent filling of lookups in the framework of `hap`.

This document is intended to be of use for both the casual `hap` user and the `hap` developer. The former will be especially interested in section 2 which contains an overview of all the lookups used in `hap` analysis and explains how to benefit from the new IR-based lookup scheme. Important technical issues are treated in section 3. Section 4 is aimed at the expert user wishing to learn more about the powerhorse of the new lookup scheme, the `InstrumentResponse` class. Filling and creating lookups in the first place

is discussed in section 5. Finally, open issues and points for future developments are the topic of section 6.
More information on lookups in the framework of H.E.S.S. analysis can be found in [1], [2] and [3].

2 Lookups in the `hap` analysis

An overview of the lookups currently in use in the `hap` analysis is given in table 1 which lists the filenames, the lookup histogram names, the parameters that each response function depends on and the meaning of the lookup histogram axes for each lookup. More details are given in the following subsections. The response functions accessible by the lookups typically depend on a source position of interest on the sky, specifically on the zenith and azimuth angles as well as the offset from the observation position. In addition, all lookups except for the radial acceptance are filled for different configurations in terms of optical efficiencies of the four telescopes. Such a configuration will be referred to as a "phase" in the following. Lookups for array-specific parameters such as effective area or point-spread function also depend on the telescope pattern (eq. (5)) of an observation.

The new lookup scheme has been implemented such that the `hap` user can choose between the old and new schemes, the new one now being the default. Once lookups prepared for the IR-based classes have been filled (sec. 5), using the new scheme is as simple as adding the name of the desired cut configuration to the `hap` configuration file:

```
[OPTIONS]
        config = std
```

Note that the names of the cut configurations have been simplified, reflecting the greater flexibility of the new scheme. Make sure that the `$HESSCONFIG` variable is set to the top-level directory containing the new lookups. For the casual `hap` user, the biggest immediate effect of the new scheme will be the fact that she no longer has to split her run list according to 3-telescope or 4-telescope runs and that she does not have to care whether a source is located towards the north or the south. The old lookup scheme can be used by setting the `Analysis/UseOldLookups` flag in the `hap` configuration file.

2.1 Shape variables

Lookup class: `HillasReco::ShapeLookupIR`

Shape variables are used in the suppression of the hadronic background. The mean reduced scaled width is defined as

$$MRSW = \frac{1}{N_{\mathrm{tel}}} \sum_{i=1}^{N_{\mathrm{tel}}} \frac{W_i - \langle W \rangle}{\sigma_W}$$

lookup file (.root)	histogram	parameters	x-axis	y-axis	z-axis
ScaleInfo	avg_length	opt,azm,zen,off	$\ln(\text{size}/p.e.)$	d/m	$\langle L\rangle$/mrad
	avg_width	opt,azm,zen,off	$\ln(\text{size}/p.e.)$	d/m	$\langle W\rangle$/mrad
	sigma_length	opt,azm,zen,off	$\ln(\text{size}/p.e.)$	d/m	σ_L/mrad
	sigma_width	opt,azm,zen,off	$\ln(\text{size}/p.e.)$	d/m	σ_W/mrad
EnergyInfo	MeanTrueEnergy	opt,azm,zen,off,tel	$\ln(\text{size}/p.e.)$	d/m	E/TeV
	SigmaTrueEnergy	opt,azm,zen,off,tel	$\ln(\text{size}/p.e.)$	d/m	$\sigma(E)$/TeV
EffectiveAreas	EffArea_TrueEnergy	opt,telp,azm,zen,off	E_{true}/TeV	$A_{\text{eff}}/\text{m}^2$	
	EffArea_RecoEnergy	opt,telp,azm,zen,off	E_{reco}/TeV	$A_{\text{eff}}/\text{m}^2$	
	EnergyBias	opt,telp,azm,zen,off	$\log_{10}(E/\text{TeV})$	$(E_{\text{reco}} - E_{\text{true}})/E_{\text{true}}$	
PSF	ThetaSq	opt,telp,azm,zen,off	$\log_{10}(E/\text{TeV})$	θ^2/deg^2	p.d.f. value
EnergyReconstruction	EnergyReconstructionPDF	opt,telp,azm,zen,off	$\log_{10}(E/\text{TeV})$	$(E_{\text{reco}} - E_{\text{true}})/E_{\text{true}}$	p.d.f. value
RadialAcceptance	RadialLookup	zen	$(\Delta\Psi)^2/\text{deg}^2$	acc/a.u.	

Table 1: Lookups used in `hap`. Here, d is the impact distance, L and W are the length and width of the shower in the camera, respectively. $(\Delta\Psi)^2$ is the square of the angular distance to the observation position. θ^2 is the square of the angular distance to the centre of a source. Azm, zen, off, opt,tel, and telp are used to abbreviate azimuth, zenith and offset angles, optical efficiencies, telescope ID, and telescope pattern (eq. (5)), respectively.

where W_i is the width of the shower image in camera i. The mean reduced scaled length $MRSL$ is defined accordingly. The mean width $\langle W \rangle$ and length $\langle L \rangle$ of a photon-induced shower and their respective fluctuations σ_W and σ_L are calculated from Monte Carlo simulations and stored in lookups. As an example, the shape lookups for one set of observation parameters are shown in figure 1.

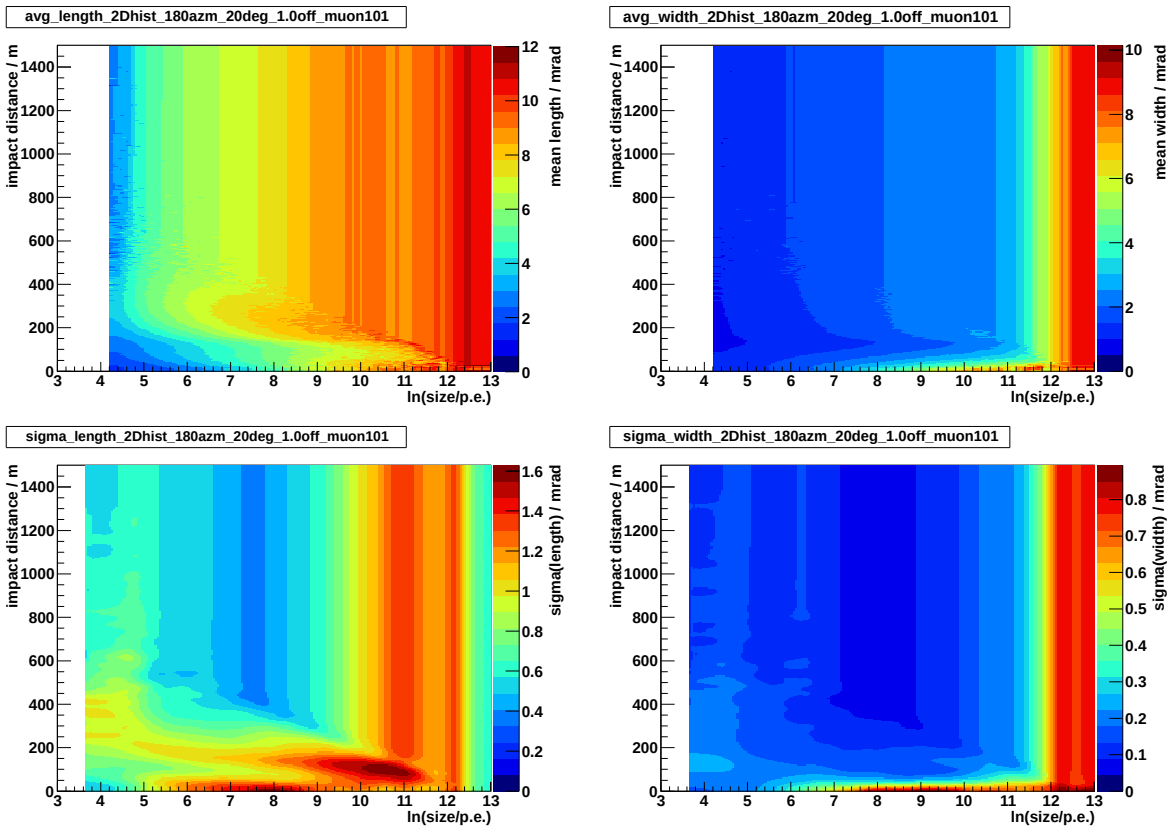

Figure 1: Shape parameters $\langle L \rangle$, $\langle W \rangle$, σ_L and σ_W, for a shower arriving from the south, at a zenith angle of 20° and an offset of 1.0°.

2.2 Energy reconstruction

Lookup class: `Reco::EnergyLookupIR`

To determine the energy of the primary photon, the shower size, i.e. the amount of light detected in a camera and measured in photo-electrons, has to be converted to an energy value for each telescope, depending on the impact distance of the shower to the telescope. The energy reconstruction lookups are filled from Monte Carlo simulations. One such raw energy lookup histogram is shown as an example in fig. 2 (left). Figure 2 (right) shows an example of interpolated values for a certain set of query parameters.

The energy lookups are filled for each telescope and for each phase, using the optical efficiency at the beginning of each phase, as determined for each telescope from the analysis of muon rings in the data [4]. In a `hap` analysis, the lookups are then queried for the appropriate phase of an observation run, in an effort to minimize the necessary muon correction. See section 3.1 for more details.

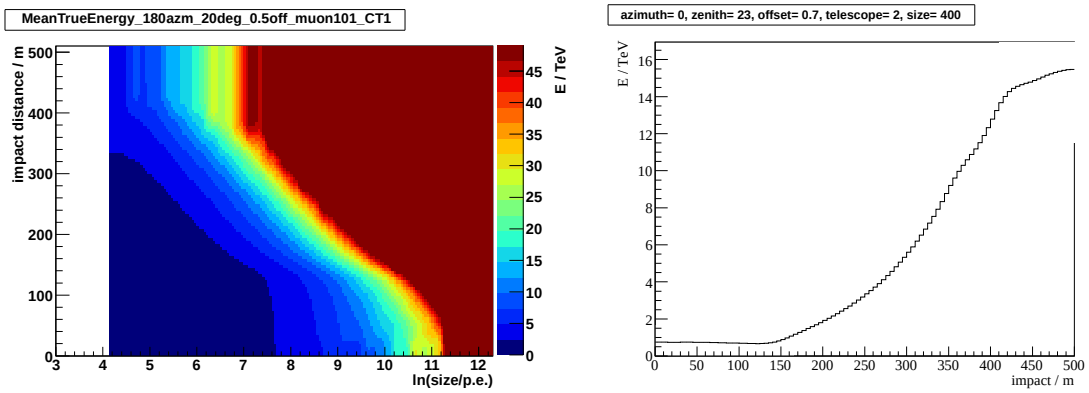

Figure 2: Left: Example for an energy lookup histogram: Energy as a function of size and impact parameter, for showers arrving from south, at a zenith angle of 20°, an offset of 0.5°, and for the phase1b optical configuration. Right: Dependence of interpolated reconstructed energy on impact distance, for the set of query parameters quoted in the plot title. This plot was generated with the script presented in sec. 2.8.

2.3 Energy resolution

Lookup class: `Flux::EnergyResolutionLookupIR`

Knowledge of the energy resolution is needed, for example, for a spectral analysis using the forward-folding method. The lookup class offers access to the probability density function (p.d.f.) of reconstructing a primary photon of energy E_{true} at an energy of E_{reco}. For convenience, following the approach of Hoppe [5], the p.d.f. is stored and returned as a function of $(E_{\mathrm{reco}} - E_{\mathrm{true}})/E_{\mathrm{true}}$, given that the true energy is E_{true},

$$p\left(\frac{E_{\mathrm{reco}} - E_{\mathrm{true}}}{E_{\mathrm{true}}}, E_{\mathrm{true}}\right)$$

For each bin in $\log_{10}(E_{\mathrm{true}})$, the p.d.f. is normalized to unit integral, so that interpolated values will also correspond to a normalized p.d.f. An example of an energy resolution lookup histogram is shown in fig. 3.

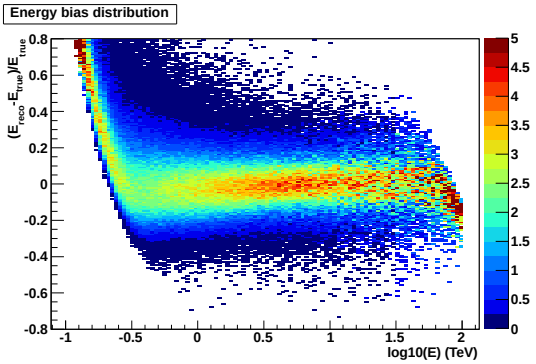

Figure 3: Energy reconstruction p.d.f. for the same parameters, for an observation with all four telescopes. The z-scale gives the probability $p(E_{\mathrm{reco}} - E_{\mathrm{true}})/E_{\mathrm{true}}, E_{\mathrm{true}})$, with each projection at fixed energy being normalized to unit integral.

2.4 Effective area

Lookup class: `Response::EffectiveAreaLookupIR`

The effective area as a function of energy is needed to convert the numbers of γ-ray candidates that have been binned in energy into fluxes. It is determined from Monte Carlo simulations according to

$$A_{\mathrm{eff}}(E) = \frac{N_{\mathrm{sel}}(E)}{N_{\mathrm{MC}}(E)} A_{\mathrm{MC}}$$

where A_{MC} is the area used in the Monte Carlo generation and $N_{\mathrm{MC}}(E)$ is the number of Monte Carlo photons generated at an energy E. $N_{\mathrm{sel}}(E)$ denotes the number of γ-ray candidates surviving all selection cuts applied. The effective area therefore depends on the selection cuts, and on the θ_{max} cut in particular. A_{eff} is available as a function of both the true and the reconstructed energies. Figure 4 contains two example effective area curves, one as a function of energy and the other one illustrating the zenith dependence of effective areas.
In addition to querying effective areas, the lookup class also allows the user to calculate several important quantities related to spectral analysis:

- The energy threshold as defined in section 5.2.2 of reference [2] limits the energy range usable for flux determination. The trigger threshold (defined by the peak in the anticipated count rate) and the safe threshold (defined as the energy where the bias in energy reconstruction drops below 10 %) can also be calculated separately.
- The mean and RMS of the relative energy resolution $(E_{\mathrm{reco}} - E_{\mathrm{true}})/E_{\mathrm{true}}$ can be used for a quick Gaussian approximation of the energy reconstruction p.d.f.

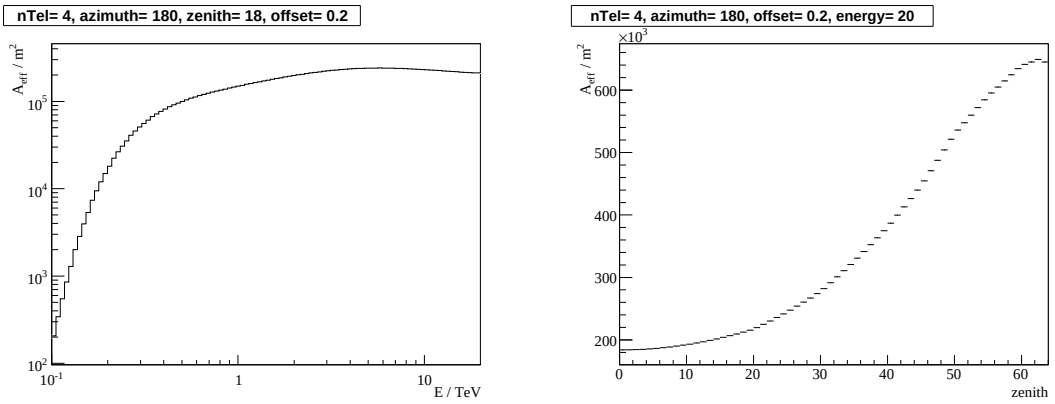

Figure 4: Effective area for two different sets of fixed parameters, as a function of energy (left) and zenith angle (right, in degrees), for std cuts.

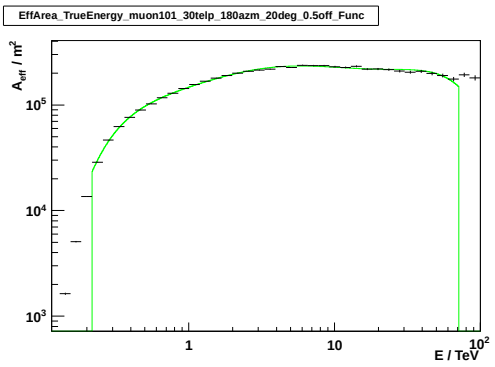

Figure 5: Example of a smoothed (green line) and unsmoothed (black histogram) effective area curve.

- The number of expected counts for a given reference flux can be calculated for a given energy range, and is used, e.g. in the generation of flux maps [6] and light curves.

Effective area lookup histograms usually have to be smoothed using an ad-hoc parameterization that involves high-order polynomials for non-TMVA configurations and a series of low-order polynomials for TMVA configurations, to account for the steps in γ-ray efficiency due to the energy bands used in the training of the classifier. The effect of smoothing is shown for one example in figure 5.

Effective areas are available for 3- and 4-telescope configurations, in an effort to keep the number of configurations and thereby the time spent on filling the lookups and file sizes limited. The number of usable 2-telescope runs only amounts to a few percent of the total.

2.5 Point spread function

Lookup class: `Flux::PSFLookupIR`

The PSF describes the angular spread of reconstructed photon positions due to the limited resolution of the instrument and is determined from Monte Carlo simulations. It can be calculated for a fixed energy or averaged over a given energy range with an assumed spectral index or averaged over an energy range and a given distribution in zenith and offset. PSF information is stored in raw lookup histograms like the one depicted in figure 6, as the probability density function $f(\theta^2, E)$ for reconstructing a primary photon of energy E at an offset θ from its true location. Each energy bin is normalized to unity,

$$\int f(\theta^2, E)\,\mathrm{d}\theta^2 = 1 \tag{1}$$

This means that we can interpolate distributions, to match any set of parameters in a lookup query. For generic parameters x_1 and x_2 with corresponding PSF distributions f_1 and f_2, the interpolated PSF at parameter x is then

$$f(\theta^2, E) = \frac{x - x_1}{x_2 - x_1} f_2(\theta^2, E) + \frac{x_2 - x}{x_2 - x_1} f_1(\theta^2, E) \tag{2}$$

and is found to be normalized to unity as well.
In the most general case, the calculation of the averaged PSF for a given set of observation runs proceeds as follows. Given the distribution of zenith angle ζ and offset angle δ as well as the mean azimuth angle φ, telescope pattern p and optical configuration ϵ_{opt} for each run, the (unnormalized) PSF can be calculated if a source spectrum $\phi(E)$ is assumed:

$$f(\theta^2) = \sum_{\mathrm{runs}} \sum_{E} \sum_{x} N_{\mathrm{exp}}(x, E) \cdot f(\theta^2, x, E) \tag{3}$$

where the number of expected counts is defined as

$$N_{\mathrm{exp}}(x, E) = \int_{E-\Delta E/2}^{E+\Delta E/2} \phi(E') A_{\mathrm{eff}}(x, E') w(x) T\,\mathrm{d}E \tag{4}$$

Here, T is the livetime of the run, w is the fraction of livetime spent in each zenith/offset bin, $x \equiv (\zeta, \varphi, \delta, p, \epsilon_{\mathrm{opt}})$ and the effective area A_{eff} is taken from the effective area lookup. Equation (3) means that the PSF is calculated as a weighted sum, with the number of expected signal counts for each contributing set of parameters taken as the weight. The individual contributions $f(\theta^2, x, E)$ can be accessed via the lookup class `Flux::PSFLookupIR`. The final $f(\theta^2)$ can then be normalized to unity and used in the analysis.

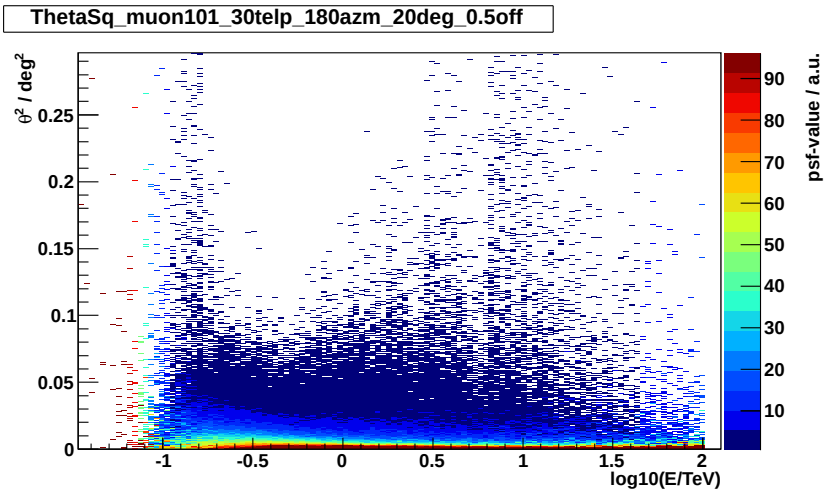

Figure 6: Example of a raw lookup histogram used in the calculation of the PSF. The PSF is normalized to unit integral in each energy bin.

2.6 Extracting the PSF

- Run a `hap` analysis to produce run-wise `BgStats` objects, e.g. using the ring background method.
- Run a `hap` analysis to extract a spectrum, using the reflected background method. The spectral index is needed for the calculation of the PSF (eqs. (3) and (4)).
- Use the `ExtractPSF` tool to calculate the PSF and store it in a ROOT file.

```
ExtractPSF --inputfile hap.root --psffile psf.root
           --spectralIndex 2.5
```

- Use the `Flux::PSFInfo` class to access information on the PSF.

```
Flux::PSFInfo* psf = Flux::PSFInfo::ReadFromFile("psf.root");
```

See the `hddst/scripts/lookups/test_PSF.C` script for some examples on how to use the `Flux::PSFInfo` object. Note that there are now convenience functions in `Plotters::SkyHist` for plotting the PSF into a corner of a sky map.

As the binning of the PSF lookup histograms is rather coarse, a suitable parametrization should be fitted to the calculated PSF histogram and used in the analysis. By default, the `PSFInfo` class uses a sum of three exponentials in θ^2 for this, which has been found to work well in general.

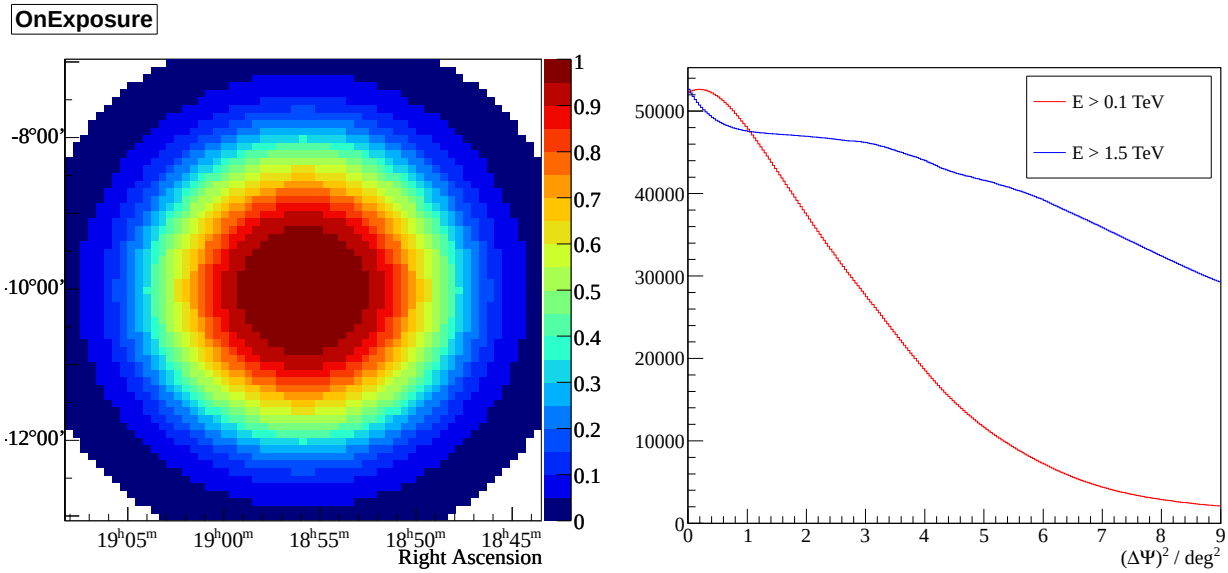

Figure 7: Left: Model for the radial acceptance for the zenith angle range from 0° to 20°. Right: Acceptance curves for two different thresholds, for the same zenith angle range.

2.7 Radial acceptance

Lookup class: `Background::AcceptanceLookupIR`

The relative acceptance of the instrument for cosmic-ray events over its field-of-view is important for the calculation of α and thus for the background estimation. It is usually determined from OFF data, i.e. from dedicated OFF runs or from observation runs of sources that do not show a substantial γ-ray signal. The acceptance is assumed to be a function of $(\Delta\Psi)^2$, i.e. to be radially symmetric around the observation position. It is normalized to have unit value at the peak. Figure 7 (left) contains an example of a radial acceptance model.

The radial acceptance varies strongly with energy, as illustrated in figure 7 (right). It is sometimes desirable or necessary to restrict the energy range of an analysis. For example, in order to create an RGB map of a source, three analyses with different energy ranges have to be performed. For the calculation of a flux map, only photons with energies above the safe energy threshold will usually be considered. In these cases, the correct acceptance curves have to be used. The new acceptance lookup offers convenient access to acceptance curves for any energy range set by the user. As statistics can be an issue for high energies and/or high zenith angles, acceptance curves can be smoothed using a sliding-window polynomial fit. The smoothing parameters can be set in the `hap` configuration file and should be tested, e.g. with the methods provided in the test script `test_radialAcceptance.C` (sec. 2.8).

An alternative way is to determine the radial acceptance from the data on a run-by-run basis. See [1] for an extensive discussion of the radial acceptance in the context of H.E.S.S. analysis.

2.8 Examples and tests

It has been checked that the old and new lookup schemes give identical results for lookup queries where expected. Small ROOT scripts showing how to query each lookup and plot information from it are available:

```
$HESSROOT/hddst/scripts/lookups/test_effArea.C
$HESSROOT/hddst/scripts/lookups/test_energy.C
$HESSROOT/hddst/scripts/lookups/test_energyPDF.C
$HESSROOT/hddst/scripts/lookups/test_energyResolution.C
$HESSROOT/hddst/scripts/lookups/test_PSF.C
$HESSROOT/hddst/scripts/lookups/test_radialAcceptance.C
$HESSROOT/hddst/scripts/lookups/test_shape.C
```

3 Technical issues

3.1 Dealing with changing optical efficiencies

The optical quality of the telescope system degrades over the years. Mirror refurbishments on individual telescopes are conducted to counteract this effect, but additionally complicate the situation from the analyzer's point of view as they cause discontinuities in the optical efficiency of the system. Figure 8 shows the optical efficiency reconstructed for CT3 over time, illustrating both the slow degradation in optical efficiency and the sudden improvement after the first mirror refurbishment. The optical efficiency ϵ of the system is derived from an analysis of muon rings in the data and expressed in units of p.e./photon. Monte Carlo simulations are done with a fixed set of optical efficiencies for the telescopes, and different sets are available, labeled *phase1* for the start of H.E.S.S., *phase1b* for the situation after a few years of running, roughly around the year 2007, *phase1c* for the state immediately before the first mirror refurbishment, and *phase1c1* and *phase1c2* for the state after the mirror refurbishments on CT3, CT2, respectively. Optical efficiencies corresponding to the various MC sets are stored in the `Monitor_Run_Muon` table in the H.E.S.S. database, with run 100 corresponding to *phase1*, run 101 corresponding to *phase1b*, and so on.
As a consequence of the complicated situation with vastly different optical configurations for all telescopes, the affected lookups (effective areas and PSF) now store information as a function of the telescope pattern p,

$$p = \sum_{\mathrm{CT}\, n \,\mathrm{in\,run}} 2^n \tag{5}$$

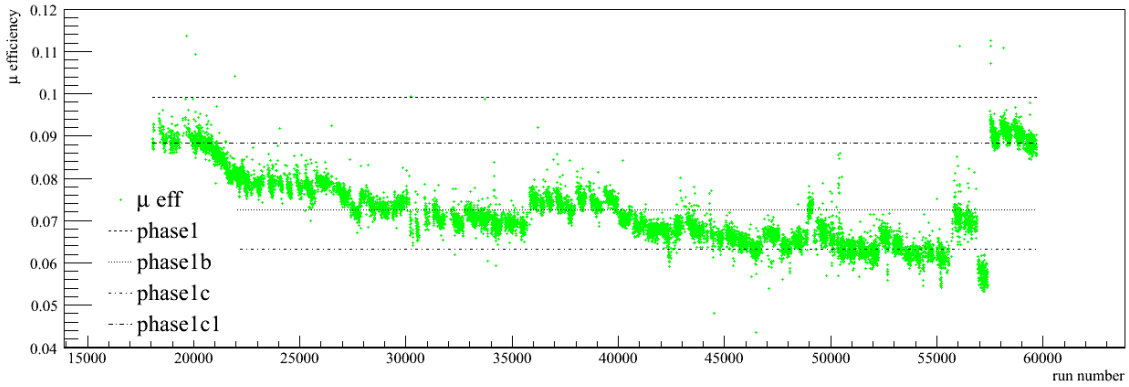

Figure 8: Optical efficiencies as reconstructed by the calibration tools for CT3, as a function of run number. The horizontal lines correspond to available sets of Monte Carlo data. This plot can be produced for all telescopes with the `hddst/PlotMuonEfficiencies` tool.

The strategy for dealing with varying optical efficiencies can be summarized as follows.
Energy is calculated on a per-telescope basis. Energy lookups for the best-guess optical configuration are used, chosen such that the necessary muon corrections are minimized (see `Muon::OpticalConfiguration` in `muon/src/MuonTimes.C`). The corrected energy is then calculated using the muon correction factors (one for each telescope), determined for the given run,

$$c_i = \frac{\epsilon_{\mathrm{MC},i}}{\epsilon_{\mathrm{data},i}} \tag{6}$$

For the **shape** lookups, the optical configuration is an additional parameter in the lookup, and again the best-guess configuration is used. The dependence of shape parameters on optical efficiency is weak though because they are stored as a function of size, not energy.
Effective areas are interpolated based on the averaged correction factors,

$$C = \frac{1}{N_{\mathrm{tel}}} \sum c_i \tag{7}$$

as

$$A_{\mathrm{eff}} = \frac{C - C_1}{C_2 - C_1} A_2 + \frac{C_2 - C}{C_2 - C_1} A_1 \tag{8}$$

where A_1 and A_2 are the effective areas evaluated for the two MC configurations bracketing the observation, $C_1 = 1$ by construction, and C_2 is the muon correction factor for the second MC configuration with respect to the first. Thresholds are computed for fixed optical configurations and then interpolated for optical efficiencies as in eq. (8) as well.
For observations during the mirror refurbishment campaign, effective areas are not interpolated but assumed to be constant over the six months between mirror upgrades.

3.2 Azimuth interpolation

As shown in [7], the effect of the geomagnetic field on the shower development is best taken into account by interpolating response functions based on the angle between the shower direction and the direction of the geomagnetic field. Linear interpolation is sufficient to a very good approximation. As a consequence, the dependence of quantities on azimuth looks like that shown for the example of an effective area in figure 9.

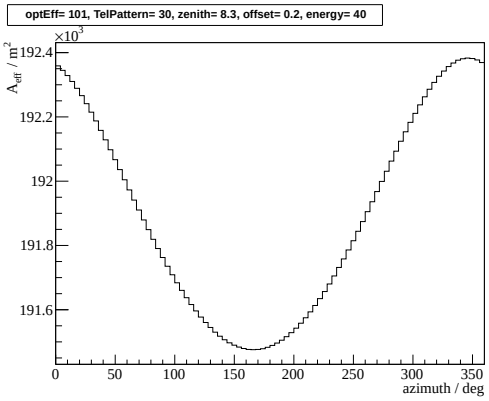

Figure 9: Dependence of effective area on the azimuth angle of a source.

3.3 Smoothing

As Monte Carlo statistics are limited and raw lookup histograms filled from simulations do not cover the parameter space in size and impact distance that corresponds to extremely high energies, shape and energy lookups are smoothed and extended. The algorithms have been copied from the old lookup scheme. The effect of smoothing and extending is visible, e.g. in the lookup histograms presented in figures 1 and 2.

3.4 Uncertainties on interpolated values

The statistical uncertainty on an entry in a lookup histogram is determined by the available Monte Carlo statistics used for filling the lookup. Let $a_1 \pm \sigma_1$ and $a_2 \pm \sigma_2$ be two lookup values with their respective uncertainties, and c_1 and c_2 be the corresponding parameters. Then, assuming a Gaussian distribution with mean a_i and variance σ_i^2 for the two values, we can define corresponding random variables A_1 and A_2 and calculate the variance for the interpolated variable

$$A \equiv \frac{c - c_1}{c_2 - c_1} A_2 + \frac{c_2 - c}{c_2 - c_1} A_1$$

at parameter c. The result is found to be

$$\sigma_A^2 = \frac{\sigma_1^2(c_2 - c)^2 + \sigma_2^2(c - c_1)^2}{(c_2 - c_1)^2} \tag{9}$$

4 The powerhorse: InstrumentResponse

The core class responsible for storing and accessing lookup histograms as well as interpolating for a given query is the `Response::InstrumentResponse`. Internally, it stores pointers to lookup histograms, that can be spread across several ROOT files, and loads the histograms into memory as needed. The storage structure is a tree-like hierarchical structure, as visualized in figure 10, where each level corresponds to a different lookup parameter. As a consequence of this data structure, accessing and interpolating lookups

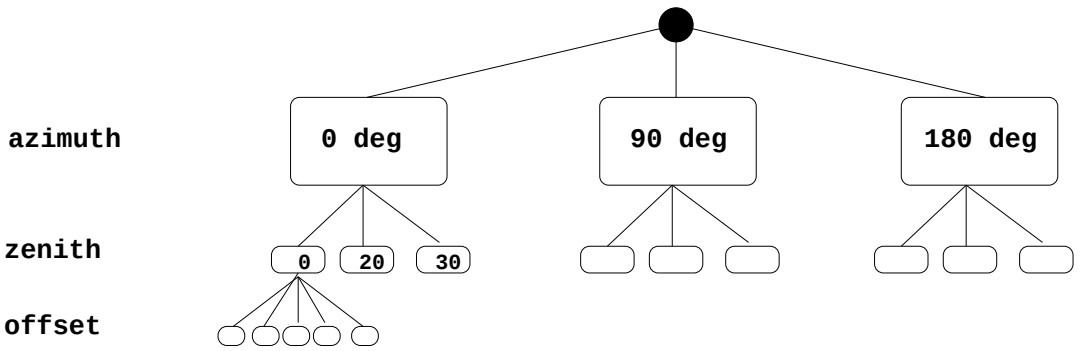

Figure 10: Internal structure of the `InstrumentResponse` class. The bottom node contains ROOT `TH1`, `TH2` or `TProfile` histograms.

of (almost) arbitrary depth (=number of parameters) is greatly simplified by using recursive function calls. Lookup classes store pointers to one or more `InstrumentResponse` objects as data members and offer wrapper functions used to query a lookup.

In the following, some of the internal workings of the IR-class are detailed. The casual `hap` user does not need to worry about these details.

For a typical interpolation, a `vector<double>` of parameters of the appropriate length is passed to the `Interpolate` function. Histograms are then accessed by name. The naming scheme of the lookup histograms is given to the respective instance of `InstrumentResponse` at construction time. Let us look at the commands for creating a lookup class to extract effective areas (see the static convenience functions in `InstrumentRepsonse` and the constructors of the lookup classes): The following assumes that the effective area histograms have names like

```
EffArea_TrueEnergy_muon101_30telp_0azm_60deg_1.0off_Func
```

```
unsigned int nparam = 5;
int ndim = 1;

std::vector<Response::ParameterType> types(nparam,Response::UNDEF);
types[0] = Response::DISCRETE;
types[1] = Response::DISCRETE;
types[2] = Response::AZIMUTH;
types[3] = Response::ZENITH;
types[4] = Response::OFFSET;

std::string names =
      std::string("optEff,TelPattern,azimuth,zenith,offset,energy");
std::string prefixes = std::string("muon,,,,");
std::string suffixes = std::string(",telp,azm,deg,off");

Response::InstrumentResponse* ir =
      new Response::InstrumentResponse(nparam,ndim);
ir->SetBaseName("EffArea_TrueEnergy");
ir->SetGlobalSuffix("Func");
ir->SetParameterNames(names.c_str());
ir->SetParameterPrefixes(prefixes.c_str());
ir->SetParameterSuffixes(suffixes.c_str());
ir->SetParameterTypes(types);

ir->Read(filename);
```

In the code snippet, we first define the number of lookup parameters and the dimension of the lookup histograms. The parameter types defined next determine how interpolation for a given parameter is handled and how a parameter value is converted to a string for the purpose of getting a lookup from a file. Table 2 contains the details. Then, the naming scheme is set up, by defining a base name which is the first string in each histogram name, a global suffix terminating the histogram names, and prefixes and suffixes for each parameter value, given as comma-separated lists. Using this information, the internal data structure can then be built by reading the keys of a ROOT file containing the lookup histograms.
The special case of the energy lookup interpolating between values of $\log_{10}(E)$ is taken care of by calling `SetInterpolateBetweenLogValues(true)`.
Energy and shape lookups are a special case in that we do not want to reject events with reconstructed offsets or zeniths that are little bit out of bounds. This is taken care of by calling `SetAllowSettingValueToOuterBound(true)`.
The internal structure of a lookup can be revealed by calling `PrintLookupArray()`.
During filling, histograms can be stored in the lookup structure by the `Store` function.

5 Filling lookups

In the new lookup scheme, the process of filling lookups is implemented completely within the framework of `hap`. All the necessary steps are conveniently controlled by the script

parameter	behaviour
`Response::UNDEF`	default behaviour, most importantly, simple linear interpolation between values is done.
`Response::ZENITH`	Linear interpolation is done between $\cos(z_1)$ and $\cos(z_2)$, z_i being the zenith angles.
`Response::OFFSET`	Values are converted to strings using a precision of one digit behind the decimal point.
`Response::AZIMUTH`	Interpolation is done based on the angle between the shower direction and the geomagentic field. Interpolation over the full circle of azimuth is possible after calling the `DuplicateAzimuth` function once (e.g. in the constructor of the lookup class). Pointers to the histograms belonging to the lowest available azimuth value ϕ are duplicated for $\phi + 2\pi$ so that neither CPU time nor memory are wasted.
`Response::DISCRETE`	No interpolation is performed for this parameter. Queries have to match one of the available parameters exactly.
`Response::INTERVAL`	As `DISCRETE`, but available parameter values are treated as interval boundaries during filling. This is needed for the radial acceptance lookup, which is filled for bands in zenith angle.

Table 2: Parameter types used in `InstrumentResponse`.

`fillLookups.pl`, located in `hddst/scripts/lookups`. It is assumed that Monte Carlo DSTs have been produced and are stored in a directory structure like that in Heidelberg (as encoded in the `get_mc_runlist` subroutine of `fillLookups.pl`). Section A contains hints on how to produce Monte Carlo DSTs and deal with them in the context of the lookup scheme.
Assuming that the `$HESSROOT/hddst/scripts/lookups directory` is in your `$PATH`, a typical procedure for filling lookups for a given cut configuration (std cuts in the examples below) is as follows: First, change directory to the desired top-level directory where lookups will be stored. Note that the θ^2 cut has to be fixed before filling lookups and then quoted using the `--thetaSqr` option. Start by creating the necessary directories and copying the analysis config file:

```
fillLookups.pl --config std --task Prepare --thetaSqr 0.0125
```

Produce the energy and shape lookups first because they are needed for the generation of the effective area lookups:

```
fillLookups.pl --config std --task EnergyShape
```

```
fillLookups.pl --config std --task MergeEnergyShape
```

Then, produce the effective area, PSF, and energy reconstruction lookups:

```
fillLookups.pl --config std --task EffArea --thetaSqr 0.0125
fillLookups.pl --config std --task MergeEffArea
```

The last step is the production of the radial acceptance lookups

```
fillLookups.pl --config std --task RadAcc
fillLookups.pl --config std --task MergeRadAcc
```

and the radial acceptance lookups for OFF data as needed by the template background maker:

```
fillLookups.pl --config std --task RadAcc --off
fillLookups.pl --config std --task MergeRadAcc --off
```

The experienced user can use a single task for the entire process:

```
fillLookups.pl --config std --task complete --thetaSqr 0.0125
```

For cut configurations involving the Heidelberg TMVA ("zeta") analysis, the procedure is a bit more complicated, as the lookups needed for the TMVA training have to be produced before the training can be performed. When the training is complete, the remaining lookups can then be produced. See section B of the appendix for more details. Help on the `fillLookups.pl` script and a list of available options can be found by typing

```
perldoc fillLookups.pl
```

For reference, table 3 contains the `Maker`s and programs involved in filling the lookups discussed above. The `fillLookups.pl` script creates `hap` config files for filling lookups

lookup	makers and programs

shape, energy	`Reco::EnergyAndShapeLookupMaker`
	`hddst/src/ProcessEnergyAndShapeLookups.C`
	`hddst/src/MergeLookups.C`

eff. area, PSF, energy res.	`Flux::EffectiveAreaMaker`
	`hddst/src/ProcessEffectiveAreas.C`
	`hddst/src/MergeLookups.C`

rad. acc.	`Background::AcceptanceLookupBgMaker`
	`hddst/src/ProcessRadialAcceptance.C`
	`hddst/src/MergeLookups.C`

Table 3: Makers and programs used in lookup filling with `hap`.

based on the templates located in

```
$HESSROOT/hddst/scripts/lookups/config
```

The workflow for filling lookups can be summarized as follows: The `Maker`s listed in table 3 are used to fill so-called proto-lookups, corresponding to one set of parameters. The `Process...` tools then perform tasks to enhance the usability of the lookups, e.g. smoothing and extending. The final lookup files containing the histograms for all parameter sets are produced by the `MergeLookups` task. Lookup classes for accessing the response functions stored in the lookups are set up and are the main interaction point for the `hap` user.

6 Open issues

In this section, the most obvious points for future improvement and development of the lookup scheme as well as some points for discussion are given.

- Adapt French analysis to use new lookup scheme if desired.
- What about H.E.S.S.-II? Adding the fifth telescope will increase the number of telescope configurations even more.
- Energy and shape lookups:
  - There is a difference for ln(size) $\gtrsim 12$ compared to old lookups, likely related to the ExtendNicely/Smoothing functions carried over from the old scheme. Maybe introduce a warning when a query is made for such a high number. How reliable is the ExtendNicely for regions with low MC statistics anyway?
- Effective area lookups:
  - The polynomial fit used for smoothing does not always work well, and it does not include the points at the lowest energies. At the moment, effective areas for non-TMVA configurations are smoothed in two iterations, to increase stability: After the first round of polynomial fits is complete, all effective areas are refitted using the best-fit parameters found for a set of similar configurations as start values and then using the χ^2 to find the best parametrization among the resulting curves, which is then used for the smoothing.
- PSF lookups:
  - Use a finer binning and smaller θ^2 range when filling lookups.
  - Try to find a simpler parametrization for fitting the PSF histogram. Stycz [8] uses a modified Student's t-function with apparently good success.
- Radial acceptance lookups:
  - Investigate using all extragalactic runs as OFF runs by cutting out wedges containing the known sources.

  - Zenith is given as range here because the lookups are generated from OFF runs. How do we interpolate correctly?
  - Is the acceptance really radially symmetric, especially after mirror refurbishments?
  - Make sure source candidates (also weak ones) are excluded in lookup generation.
  - Use finer zenith binning?
  - Dependence on time interval seen, possibly due to changing optical efficiencies. Can we take this into account?
  - Investigate smoothing with high-dimensional polynomial instead.

- It would be great if we could put all information on optical efficiencies into the DSTs and get rid of the dependence on the database. We could also just merge the three muon parameter classes and the two reader classes to make things simpler.

- As an alternative to the currently implemented lookup scheme, one could think about lookups that are custom-made for each run, i.e. filled from Monte Carlos generated with the appropriate telescope pattern, optical efficiency of the telescope system, zenith angle of the observation, and so on. The only remaining paramter for an interpolation would be the offset angle. Such an approach would reduce systematic effects and is likely to be the only viable option for large telescope arrays like CTA, where the number of possible telescope patterns becomes enormous.

A Dealing with raw Monte Carlo simulations

Here, we briefly sketch the necessary steps to prepare new Monte Carlo simulations for use in filling lookups. This might become necessary, e.g. after a mirror refurbishment campaign.

- Produce MonteCarlo DSTs. You can use the `produce_DSTs` task in the script `hddst/scripts/produce_MC_DSTs.pl` for this, after changing the directory paths at the top of the script.

- Check integrity of the DSTs. For example, use the `check_DSTs` task of the `produce_MC_DSTs.pl` script.

- Extract muon efficiencies. Run the `extractMuonEff` task in the `produce_MC_DSTs.pl` script. Then, run the `hddst/McMuonEff` task on the resulting `*muonEff*.root` files. Read off the mean muon efficiencies corresponding to the given set of simulations.

- Update the database table `Monitor_Run_Muon` with the new muon efficiencies.

```
ssh hess@lfs4
dbeditor -C calib
```

Select the `Monitor_Run_Muon` table. Run numbers for Monte Carlo configurations are numbered from 100 upwards. Enter the condition `Run=103` and press `Ctrl+L` to see an example. Add the entries and use `insert new` to save.

- Determine a suitable time frame to use for the new configuration and update the functions in `Muon::MuonTimes` accordingly. The `hddst/PlotMuonEfficiencies` tool may prove useful for this.
- Copy the DSTs to the hess account.
- Adapt the `fillLookups.pl` script to include the new configuration if necessary, as explained in its header.

B Producing lookups for Heidelberg TMVA configurations

Use the `--zeta` and `--zetaFile` options of `fillLookups.pl` for correct handling of zeta configurations!

- Make sure the ScaleInfoOff lookup is present. A raw version of this can pe produced using the TMVA tools. (Ask the TMVA expert.) Then, use the `ShapeOff` task of `fillLookups.pl` and the `--rawScaleOffFile` option to produce the final ScaleInfoOff lookup.
- In addition, produce the energy and shape lookups.
- Run the TMVA training.
- Then, continue with lookup production. The remaining lookups will depend on the event selection done by the TMVA analysis, which is why the training has to be performed before the high-level lookups can be filled.

Acknowledgements

Dalibor, Bernhard and André already put a lot of effort in the improvement of the lookup scheme. I would also like to thank them, as well as Karl, Konrad and Christopher, for extremely useful discussions.

References

[1] D. Berge, PhD thesis (2006), Heidelberg

[2] D. Nedbal, PhD thesis (2008), Heidelberg

[3] F. Aharonian et al., A&A 457 (2006) 899-915

[4] O. Bolz, PhD thesis (2004), Heidelberg

[5] S. Hoppe, PhD thesis (2008), Heidelberg

[6] S. Carrigan et al., H.E.S.S. internal note in preparation

[7] K. Bernlöhr, H.E.S.S. internal note 05/02

[8] K. Stycz, "An unbinned Fit of H.E.S.S. Sources using the Log-Likelihood Method", diploma thesis (2010), Friedrich-Alexander Universität Erlangen-Nürnberg