technische universität
dortmund

Arbeit zur Erlangung des akademischen Grades
Bachelor of Science

LaTeX-Vorlage für die Bachelorarbeit in TU-Farben

Katharina Frantzen
geboren in Castrop-Rauxel

2014

Lehrstuhl für Experimentelle Physik V
Fakultät Physik
Technische Universität Dortmund

Erstgutachter:	Prof. Dr. Erstasdasgutachter
Zweitgutachter:	Prof. Dr. Zweitgutachter
Abgabedatum:	11. Juli 2014

Inhaltsverzeichnis

1 Monte Carlo-Daten

Ein großes Ziel in der Astroteilchenphysik ist es, Aussagen über die Energiespektren von astrophysikalischen Quellen zu machen.

In der Cherenkovastronomie sind dafür Monte Carlo (MC)-Daten von grundlegender Bedeutung: Für die Rekonstruktion eines Energiespektrums einer Gamma-Quelle müssen zunächst einmal Signalevents, die aus der Quelle kommen und Background-Events voneinander getrennt werden. Dies geschieht heutzutage mit künstlichen Lernalgorithmen, die auf wohl bekannten Beispieldaten, den MC-Daten, trainiert werden müssen. Auch für die Methode der Entfaltung des Energiespektrums werden MC-Daten benötigt.

Die komplette Produktion der MC-Daten für das MAGIC Experiment wurde im Rahmen dieser Doktorarbeit durchgeführt und wird in diesem Kapitel detailliert beschrieben.

Es werden zunächst die Programme beschrieben, die in der MAGIC MC - Simulationskette und später in der Kalibrationskette genutzt werden. Angefangen mit der Simulation der Luftschauer mit *CORSIKA* über die Reflektor- und die Kamera-Simulation bis zur Berechnung der Bildparameter des bereinigten Events wird hier alles beschrieben. Dabei bauen die Simulationsschritte aufeinander auf, sodass die simulierten Corsika-Events bis zum letzten Schritt weiter prozessiert werden.

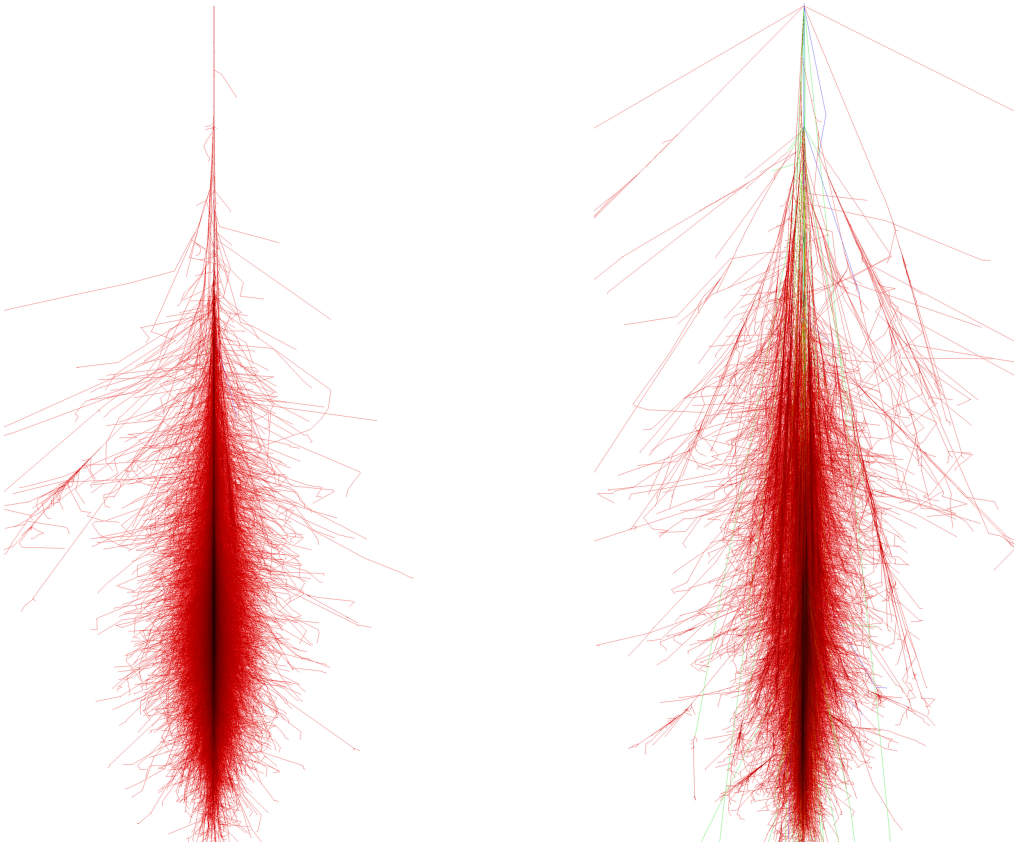
Für die eigentlichen Simulationsprogramme (*CORSIKA*, *Reflector* und *Camera*) wird jeweils eine Übersicht über einige wichtige Inputparameter, die den Programmen als text-Dateien, sogenannten Inputcards, übergeben werden, gegeben. Nach Abschluss der Simulationskette liegen dann die MC Daten in der gleichen Form vor wie die aufgenommenen Daten der Teleskope. Die anschließenden Schritte der Kalibrationskette (*Sorcerer*, *Star* und *Superstar*) werden ebenfalls beschrieben. Diese Kalibration wird im MAGIC Datenzentrum genauso mit den real aufgenommenen Daten durchgeführt, sodass am Ende alle Daten (reale und simulierte) im gleichen Format vorliegen.

Nachdem alle Programme zur Simulation und Kalibration erklärt wurden, wird auf die automatische Produktionsstruktur auf dem Rechencluster LiDO an der TU Dortmund eingegangen.

1.1 Schauersimulation mit CORSIKA

CORSIKA wurde am Forschungszentrum Karlsruhe ursprünglich für das KASCADE-Experiment entwickelt und wird heute in vielen Astroteilchenexperimenten eingesetzt. Mit *CORSIKA* werden ausgedehnte Luftschauer, ausgelöst von kosmischer Strahlung simuliert.

Es können verschiedene Primärteilchen wie Protonen, Eisenkerne oder Photonen als Primärteilchen eingestellt werden. Diese Primärteilchen werden durch die Atmosphäre propagiert, wo sie Wechselwirkungen mit den Kernen der Luft eingehen und Schauer produzieren oder aber zerfallen. Die entstandenen Schauer werden dann bis zum Teleskop simuliert.



(a) Simulation eines Schauers mit 1TeV-Photon als Primärteilchen

(b) Simulation eines Schauers mit 1TeV-Proton als Primärteilchen

Abbildung 1.1: Schauersimulation mit CORSIKA mit 1TeV Primärteilchen

In Abb.1.1 sieht man die verschiedenen Eigenschaften der Schauer: Hadronische Schauer haben einen größeren Querschnitt und sind weniger kompakt verglichen mit elektromagnetischen.

Es sind verschiedene Modelle für die hadronische Wechselwirkung bei hohen Energien und bei niedrigen Energien implementiert. Für die MAGIC MC-Simulation werden QGSJET II für die hadronische Wechselwirkung bei hohen Energien und FLUKA für die Wechselwirkung bei niedrigen Energien benutzt. Elektromagnetische Prozesse werden mit dem EGS4-Modell beschrieben. Eine Simulation der Cherenkovphotonen, die von den geladenen Teilchen produziert werden, die durch die Luft propgaieren, erfolgt ebenfalls.

1.1.1 Mmcs

Eine speziell für das MAGIC-Teleskop adaptierte Version von *CORSIKA* wird in der Standardsimulation genutzt. In dieser Version wurde der Einfluss des Magnetfeldes vor der ersten Wechselwirkung auf das Primärteilchen vernachlässigt um zu verhindern, dass es zu weit vom Teleskop abgelenkt wird. Außerdem wurde die Simulation der Cherenkovwellenlänge eingebaut und es wurde eingebaut, dass alle Informationen über das Primärteilchen gespeichert werden.

1.1.2 Inputcards

Die Inputcard enthält Informationen über den Teleskopstandort, die simulierten Schauer und über das Magnetfeld.

Zunächst können allgemeine Angaben zur Anzahl der simulierten Events gemacht werden. So erhält jeder *CORSIKA*-Run eine eigene Runnumer und die Anzahl der Schauer pro Run wird durch den Parameter `NSHOW` angegeben.

Außerdem erfolgen noch Einstellungen über die Eigenschaften der Schauer. Der Parameter `PRMPAR` gibt die Art des Primärteilchens an und `ERANGE` beschreibt den Energiebereich, in dem die Primärteilchen simuliert werden. Die Steigung des Energiespektrums wird mit dem Parameter `NSLOPE` eingestellt. Zenit- und Azimutbereich, in dem die Schauer simuliert werden, wer den durch `THETAP` und `PHIP` angegeben.

Zudem werden in *CORSIKA* noch Standortangaben zur Geographie des Teleskopstandortes gemacht. So wird die Höhe über NN im Parameter `OBSLEV` angegeben und eine Angabe über die horizontale, bzw. vertikale Komponente des Magnetfeldes auf La Palam befindet sich im Parameter `MAGNET`. Der Parameter `ATMOSPHERE` gibt an, welche Parametrisierung der Atmosphäre genutzt wird.

Abgesehen von diesen allgemeinen Angaben gibt es in der Inputcard noch einige Parameter, die dediziert für die Simulation der Cherenkovphotonen sind. Der Wellenlängenbereich, in welchem Cherenkovphotonen simuliert werden, wird mit `CWAVLG`

angegeben. Der Impact-Parameter, der angibt, in welcher Entfernung der Schauer vom Teleskop entfernt auftreffen kann, wird mit `CSCAT` angegeben. Der Parameter `CERTEL` beinhaltet Standort und Größe der Teleskope, die man simuliert und mit der Option `CERFIL` wird angegeben, ob der Output über die Cherenkovphotonen in eine extra Datei geschrieben wird.

1.2 Simulation des Reflektors mit Reflector

Wie der Name schon sagt, wird im Programm *Reflector* vor allem der Reflektor simuliert, allerdings kommen noch einige Zusatzfunktionen ins Spiel.

Mit *Reflector* wird zuerst die atmosphärische Absorption der Cherenkovstrahlung in der Luft simuliert. Dabei werden Rayleigh-Streuung, Mie-Streuung und die Absorption an Ozon simuliert. Als nächstes wird mit Hilfe von Ray-Tracing simuliert, ob die Cherenkovphotonen den Teleskopspiegel treffen. Dort angekommen muss die Absorption der Aluminium-, bzw. Glasspiegel berücksichtigt werden und die Photonen reflektiert werden. Nach der Reflexion am Spiegel wird nacheinander überprüft, ob die Photonen die Kamera treffen und letztendlich die Ankunftszeiten in der Kamera bestimmt.

1.2.1 Inputcards

Für diese Simulationsschritte werden bestimmte Inputparameter, bzw. Inputdateien benötigt, die die Eigenschaften des Reflektors beschreiben. Die wichtigsten werden im Folgenden kurz erklärt.

Es muss der Pfad zur *CORSIKA*-Inputdatei sowie die Output-Pfade für die beiden Reflektordateien (je eine pro Teleskop) angegeben werden.

Teleskopstandort und Wobble Position müssen angegeben werden.

Pro Teleskop wird ein `mirror_definition_file` für die verschiedenen Spiegel (Glas und Aluminium) benötigt. In dieser Datei sind grundlegende Eigenschaften des Teleskops wie zum Beispiel der Abstand zwischen Spiegel und Kamera festgelegt. Des Weiteren wird der Kameraradius angegeben und die Anzahl der Spiegel mit ihrer jeweilige Größe. Für jeden Spiegel sind einzeln ihre jeweilige ID, der Fokalabstand, die Koordinaten des Spiegelmittelpunkts und der Normalenvektor des Spiegels aufgelistet.

Im sogenannten `reflectivity_file` sind die gemessenen Reflektivitäten der einzelnen Spiegel aufgelistet.

Das `axisdev_file` beschreibt die Abweichung der einzelnen Spiegel von der idealen Pointing Position.

Im `measuredpsf_file` ist die gemessene PSF eingetragen.

Letztendlich muss man in Reflector noch das Atmosphärenmodell, welches für die Simulation benutzt werden soll, angegeben werden. Aktuell wird das MagicWinter-

Modell benutzt. Eine Parametrisierung des Modells enthält abhängig von der Höhe die Dichte, die Dicke und der Brechungsindex (siehe Arbeit von Mareijke Haffner).

1.3 Simulation von Kamera und Elektronik mit Camera

Das Programm *Camera* simuliert das komplette Verhalten der Kamera inklusive der Beiträge des Nachthimmeluntergrundes (NSB). Es werden neben den Schauern also auch der diffuse NSB und Sterne im Field of View simuliert. Des Weiteren werden der Trigger und das elektronische Rauschen simuliert, sodass bei einer Änderung der Hardware der Kamera, bzw der Elektronik nicht die komplette MC Kette noch einmal durchlaufen werden muss. In Abb.1.2 befindet sich ein generiertes Cherenkovevent in der Kamera mit einer Energie von 3,0 TeV.

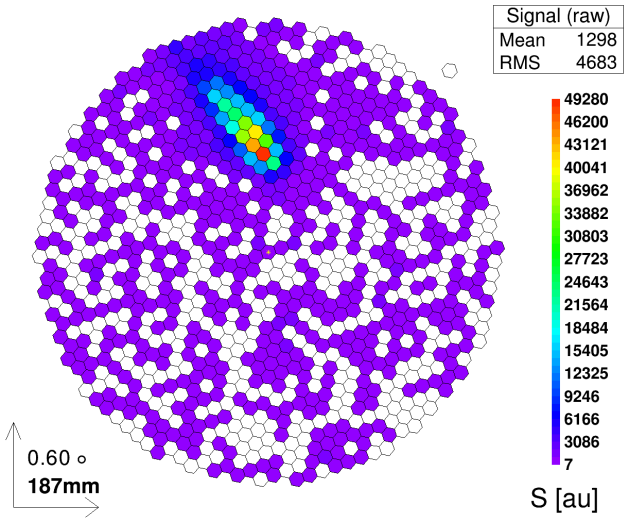


Abbildung 1.2: Darstellung eines simulierten Schauers mit einer Primärteilchenenergie von 3,0 TeV in der Kamera. Dargestellt ist der Pixelinhalt in [a.u.].

Im Folgenden wird die Simulation beschrieben, die mit zwei möglichen vorbereitenden Schritten beginnt: der Simulation des NSB und der Sterne im FoV.

1.3.1 StarFieldAdder und StarResponse

Da MAGIC sensitiv für Sterne bis zur Magnitude 10 ist, tragen die Sterne zum Rauschen in der Kamera bei. Das Programm *StarFieldAdder* berechnet anhand von einem Katalog, welche Sterne im FoV sind und berechnet dann, wieviele Photonen von diesen Sternen mit welcher Wellenlänge den Spiegel treffen. Der Output wird

im *CORSIKA*-Format geschrieben und muss noch durch *Reflector* laufen, bevor er von *Camera* benutzt werden kann. Eine Simulation der Sterne im FoV wird in der Standard-MC-Simulation nicht durchgeführt, da die MC-Daten für alle Quellen in einem bestimmten Zenitbereich benutzt werden sollen und nicht für jede Quelle eigene MC-Daten produziert werden.

Um die Simulation zu beschleunigen und damit nicht für jedes Event der diffuse NSB neu berechnen werden muss, wird mit *StarResponse* eine NSB-Datenbank generiert und daraus dann das NSB-Rauschen berechnet, welches dann zusammen mit dem Cherenkovphotonensignal zum Trigger geht.

1.3.2 Simulation von Kamera und Elektronik

Nachdem alle Parameter aus der Inputcard eingelesen worden sind und die individuelle PixelResponse mit NSB gefüllt wurde, erfolgt die eigentliche Verarbeitung der zuvor simulierten *Reflector*-Daten. Die Photonen aus den Schauern werden eingelesen und für jedes Photon werden folgende Werte einzeln bestimmt:

- Pixelization: In welchem Pixel kommt das Photon an
- PhE-Produktion: Unter Berücksichtigung der wellenlängenabhängigen Quanteneffizienz jedes PMTs und der Winston Cones wird entschieden, ob ein Photoelektron erzeugt wird.
- Channel Response: Für jedes Photoelektron, was die Photokathode verlässt, wird das Analogsignals des PMTs generiert.

Danach werden die Signale und Ankunftszeiten aller Photonen eines Pixel superponiert und die Response des Triggers und FADC Systems berechnet und elektronisches Rauschen hinzugefügt. Das wird für alle Pixel gemacht und man erhält das analoge Signal. Dann wird durch eine Baseline-Subtraction die AC Kopplung simuliert, die zwischen dem PMT Output und dem Signal ist, welches in den Discriminator Trigger geht. Danach steht noch die Simulation des Triggers aus. Es wird gecheckt, ob das analoge Signal eine bestimmte Schwelle, die Discriminator Schwelle, überschreitet. Falls ja, wird ein digitales Output Signal simuliert. Dann wird der first Level Trigger simuliert: Ob das Event triggert beruht auf seiner Topologie und der Multiplizität. Dafür wird eine Nächste-Nachbarn-Bedingung überprüft, d.h. die minimale Anzahl an Pixeln, die einen bestimmten Photoneninhalt haben und ihre Verteilung in der Kamera. Falls diese Bedingungen erfüllt sind, wird ein First Level Trigger Signal generiert und der Output des FADC Systems, welches die digitalisierte Form des analogen Signals ist, wird gespeichert. Dann wird der nächste Schauer prozessiert.

1.3.3 Inputcard

Im Folgenden werden einige Parameter erklärt, die in der Inputcard von *Camera* enthalten sind. Wie in jeder Inputcard müssen die Pfade zu den zu prozessierenden *Reflector*-Dateien angegeben sein sowie ein Output-Pfad. Im Folgenden werden einige Dateien angegeben, die das Programm Camera benötigt:

- `qe_file`, in welchem die Quanteneffizienz der PMTs als Funktion ihrer Wellenlänge angegeben ist
- `lightcollision.dat`, was die Lichtkollektionseffizienz der Pixel als Funktion des Winkels zwischen Photontrajektorie nach der Reflexion am Spiegel und der Kameraebene angibt. In diesem Wert muss die Transmission des Plexiglasfensters der Kamera, die Reflektivität der Winston Cones (Lichtleiter) und die Kollektionseffizienz der Photoelektronen der ersten Dynode der PMTs beinhalten.
- `star_field_file`, welches vorher generiert wurde und die Sterne im FoV enthält.

Zudem werden einige Parameter, die den NSB betreffen hier eingeführt. Erst einmal muss definiert werden, ob der NSB simuliert werden soll oder nicht. Dies geschieht mit dem Befehl `nsb_on`, bzw. `nsb_off`. Des Weiteren muss der Pfad zur vorher generierten NSB Datenbank gegeben werden, was mit dem Parameter `nsb_directory` geschieht. Falls die äußeren (früher größeren) Pixel einen anderen Gain haben, ist dort auch der Einfluss des NSB anders. Dies wird durch `nsb_dir_outer` angegeben. Ein weiterer Parameter, der den NSB betrifft, ist `nsb_mean`. Die erste Zahl gibt die Amplitude des NSB in Anzahl an Photoelektronen pro einem inneren Pixel der Kamera pro ns an. Wenn man also eine andere Geometrie (größere Spiegel oder größere Pixel) oder eine andere Kamera (andere Quanteneffizienz) simulieren möchte, wird die Photoelektronrate automatisch skaliert. Die zweite Zahl gibt an, wie groß die Anzahl der Photoelektronen eines Schauers minimal sein muss, damit NSB für diesen Schauer generiert wird. Die meisten Schauer produzieren wenige oder gar keine Photonen in der Kamera und werden ignoriert. Für alle Schauer mit weniger als 10 Cherenkovphotonen wird kein NSB produziert, da diese Schauer mit hoher Wahrscheinlichkeit auch nicht getriggert werden.

Der Parameter `mirror_fraction` gibt die Zahl der Spiegel an, die zur Reflexion des Lichts beiträgt. Mit Hilfe dieses Parameters können fehlende Spiegel in der Simulation berücksichtigt werden.

Der Parameter `ct_geom` gibt Aufschluss über die Kamera-Geometrie für jedes Teleskop und beinhaltet Anzahl, Größe und Position der Pixel.

Abgesehen von den oben beschriebenen Parametern gibt es noch zahlreiche weitere, die die Triggereinstellungen und die FADC-Einstellungen betreffen.

Nachdem *Camera* durchgelaufen ist, ist die eigentliche Simulationskette beendet und die Daten liegen im gleichen Format vor wie die real aufgenommenen. Es erfolgt die gleiche Kalibration wie auch bei den echten Daten und die Einstellungen in den folgenden Programmen unterscheiden sich kaum noch.

1.4 Calibration

Ziel der Kalibration ist es, für jeden Pixel die Ladung in Photoelektronen und die Ankunftszeit der Photonen zu bestimmen. Dafür muss der Lichtpuls extrahiert und die Baseline abgezogen werden. Die Baseline wird mit Hilfe von pedestal Events bestimmt, also Events mit zufälligem Trigger, welche keine Schauerpulse enthalten sollten und dann in den richtigen Events abgezogen [vgl. Abb.1.3]. Ziel der Calibration Runs - Events mit bekanntem Lichtpuls - ist es, die Konversionsfaktoren der einzelnen Pixel zu berechnen.

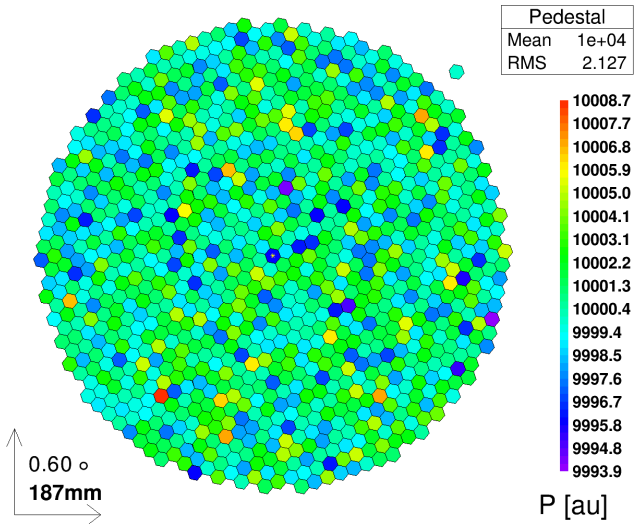


Abbildung 1.3: Darstellung eines simulierten Pedestal Events in der Kamera. Dargestellt ist der Pixelinhalt in [a.u.].

In der Realität sind Pedestal Events, Events mit zufälligem Trigger und ohne Puls und Calibration Runs sind Runs, die genommen werden, wobei die Calibration Box benutzt wird. Diese Calibration Box befindet sich im Zentrum der Spiegel und verursacht kurze Lichtpulse mit konstanter Intensität in Richtung Kamera.

1.4.1 Signalextraktion

Ziel der Signalextraktion ist die Integration der Counts in der Pulsregion ohne Baseline. Dafür gibt es verschiedene Methoden:

- Fixed window: Mit dieser Methode wird an einer a priori bekannten Position, an der man den Cherenkovpuls erwartet, über eine bestimmte Länge integriert.
- Sliding window: Bei dieser Methode wird das Integrationsfenster so lange verschoben, bis man den Bereich gefunden hat, in dem das Signal am höchsten ist und integriert dort.
- Spline: Diese Methode beruht auf der Sliding Window-Methode, allerdings erfolgt die Integration mit Hilfe eines Polynoms.

Aktuell wird in MAGIC die Sliding Window Methode benutzt. Es wird in einem 60 time slice großen Bereich nach dem Pulse gesucht und dann über 6 time slices integriert (3 ns).

Nachdem man nun das Signal extrahiert hat, wird es noch von readout counts in Photoelektronen umgerechnet.

1.4.2 Ankunftszeitbestimmung

Die Ankunftszeit des Pulses wird als mittlere Time slice des integrierten Windows gewichtet mit dem Signal darin berechnet:

$$t_{arrival} = \frac{\sum i s_i}{\sum s_i}, \tag{1.1}$$

mit i: time slice Nummer, s_i: Signal in slice i und der Summierung über 6 slices als Integrationsfenster.

Eine typische Verteilung der Ankunftszeiten für einen Cherenkovschauer in Kamera ist in Abb.1.4 zu sehen.

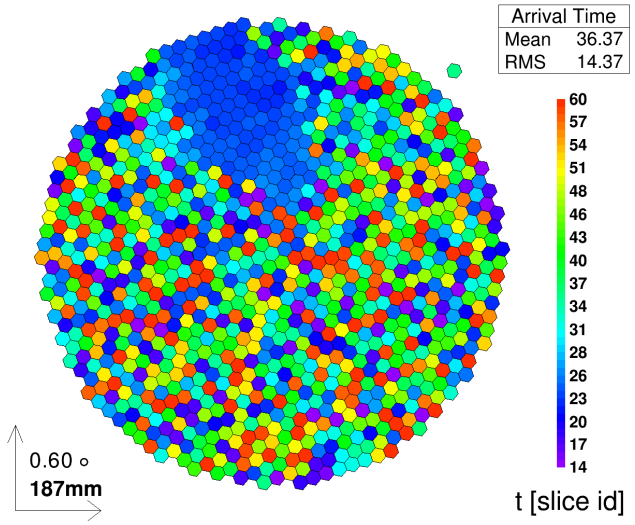


Abbildung 1.4: Darstellung der Ankunftszeiten eines Schauers in der Kamera. Dargestellt sind die Ankunftszeiten in time slices.

1.5 Star - Image Cleaning und Bildparametrisierung

1.5.1 Image Cleaning

Im Programm *Star* geschieht das Image Cleaning und die Parametrisierung des Schauerbildes.

Wird ein Event getriggert, werden die Signale der einzelnen Pixel gespeichert. Sowohl Night Sky Background (NSB) als auch elektrisches Rauschen sind dann noch in dem Eventbild enthalten (siehe Abb.1.2). Ziel des Image Cleanings ist es, das Bild von allem Background zu bereinigen, sodass nur noch das Signal vom eigentlichen Schauer überbleibt und eine robuste und stabile Parametrisierung dieses Schauerbildes durchgeführt werden kann. Einerseits können Pixel, die nicht zum eigentlichen Schauer gehören und das Image Cleaning überleben, zu einer falschen Rekonstruktion führen, andererseits ist es zu vermeiden, dass zu viele Pixel im Cleaning wegfallen, da es so zu einem Signalverlust kommt, was ebenfalls zu einer schlechteren Rekonstruktion führen kann. Also werden schlaue Algorithmen benötigt!

Der einfachste und älteste Algorithmus ist das Absolute Image Cleaning. Dabei wird nur die Photoncharge in den einzelnen Pixeln benutzt. Es werden zwei Schwellwerte definiert für die Core (Q_{Core}) und die Boundary Pixel ($Q_{boundary}$). Nun werden alle Pixel mit einer Photoncharge die größer als Q_{Core} ist, ausgewählt. Ein Pixel ist dann ein Corepixel, wenn er noch einen Nachbarn hat, welcher ebenfalls eine Photoncharge hat, die dieses Limit überschreitet. Im zweiten Schritt werden alle

Pixel mit direkten Nachbarn, die den vorherigen Schritt überlebt haben und eine Ladung größer als $Q_{boundary}$ haben, als boundary Pixel markiert. Alle anderen Pixel fliegen raus. Dieser Image Cleaning Algorithmus benutzt keine Zeitinformation und es wird damit keine niedrige Energieschwelle erreicht. Die verschiedenen Rauschlevel zwischen den Pixeln werden auch nicht berücksichtigt.

Eine Weiterentwicklung dieses Cleaningalgorithmus, welcher auch die Zeitinformationen benutzt, ist das Time Constrained Absolute Image Cleaning. Es funktioniert genauso wie das absolute Image Cleaning, allerdings wird die Ankunftszeit der Photonen zusätzlich berücksichtigt.

Eine weitere Entwicklung ist das Sum Image Cleaning, welches folgendermaßen funktioniert: Es wird eine Zweier-, Dreier- oder Vierer-Kombination von Nachbarpixeln gesucht und deren Signale aufsummiert. Ist das aufsummierte Signal über einer bestimmten Schwelle, werden die Ankunftszeiten in der Gruppe untereinander verglichen. Liegen diese nahe genug zusammen, werden die Pixel behalten, ansonsten verworfen. Anschließend werden Nachbarpixel gesucht und deren Pixelinhalte und Ankunftszeiten betrachtet.

Beim dynamischen Sum Cleaning wird zusätzlich noch die Size, der Gesamtphotoneninhalt, eines Events mit berücksichtigt.

1.5.2 Bildparametrisierung

Die Bildparameter basieren auf den Hillas Parametern von 1985 [] und berücksichtigen die Verteilung der Photonen in den Pixeln, die zum gecleanten Event gehören. Abb.1.5 zeigt ein gecleantes Event in der Kamera.

Im Folgenden werden einige wichtige Bildparameter aufgelistet und beschrieben (vgl. Abb.1.6):

- `size`: Die Gesamtzahl der Photoelektronen in einem ShowerImage wird als `size` bezeichnet. Für feste Zenitwinkel und Impactparameter ist diese Größe proportional zur gesuchten Primärteilchenenergie.

- `CoG` (Center of Gravity): Das Center of Gravity des Schauers bezeichnet die Position des gewichteten Mean Signals entlang der X- und Y-Achse in der Kamera. X und Y sind die ersten Momente der Ladungsverteilung und werden `MeanX`, bzw. `MeanY` genannt.

- `Width`: Die halbe Breite der kleinen Halbachse der an den Schauer gefitteten Ellipse wird als `Width` bezeichnet. Mit Hilfe dieses Paramters lassen sich Aussagen über die transversale Ausbreitung des Schauers und damit auch über den Ursprung des Schauers (hadronisch oder elektromagnetisch) treffen. Dieser Parameter ist somit ein guter Trennparameter.

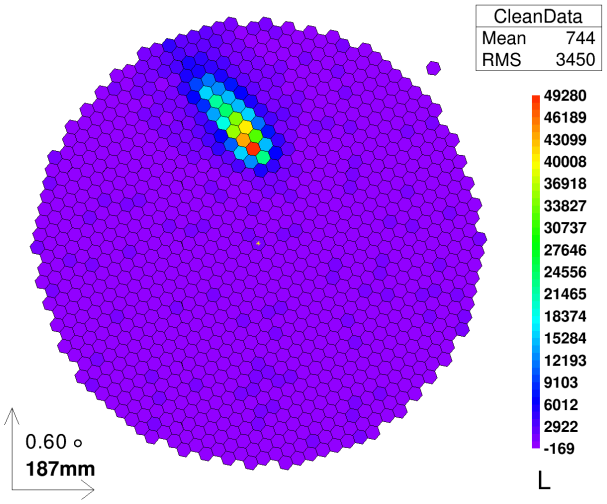


Abbildung 1.5: Darstellung des gecleanten Events in der Kamera.

- `Length`: Die halbe Länge der großen Halbachse wird mit dem Parameter `Length` bezeichnet. Dieser Parameter trifft eine Aussage über die longitudinale Entwicklung des Schauers und ist im Allgemeinen größer für Hadroninduzierte Schauer als für Gammainduzierte Schauer.
- `Conc-n`: Der Anteil der Photoelektronen, welche in den n hellsten Pixeln enthalten sind, wird als `Conc-n` bezeichnet. Damit ist es möglich, die Kompaktheit des Schauermaximums zu beschreiben. Bei Gamma-Schauern ist die Region sehr kompakt.
- `Leakage`: Dieser Parameter beschreibt den Anteil des Signals im äußeren Kameraring im Vergleich zur totalen Image Size. Es ist möglich mit diesem Parameter den Signalverlust zu beschreiben und zu entscheiden, ob der Schauer überhaupt noch vernünftig rekonstruiert werden kann.
- `M3Long`: Dieser Parameter ist das dritte Moment entlang der großen Halbachse und beschreibt die Asymmetrie des Schauers. Es lässt sich damit also auf die Herkunftsrichtung des Schauers schließen. `M3Long` ist positiv wenn der Schauerschwerpunkt in Richtung des Kamerazentrums liegt, ansonsten negativ.
- `Number_of_Islands`: Wie der Name schon sagt wird mit diesem Parameter die Anzahl der Inseln, die nach dem Cleaning übergeblieben sind, bezeichnet. Je größer dieser Wert ist, umso mehr Inseln sind noch vorhanden und umso wahrscheinlicher ist der Schauer hadronischen Ursprungs.

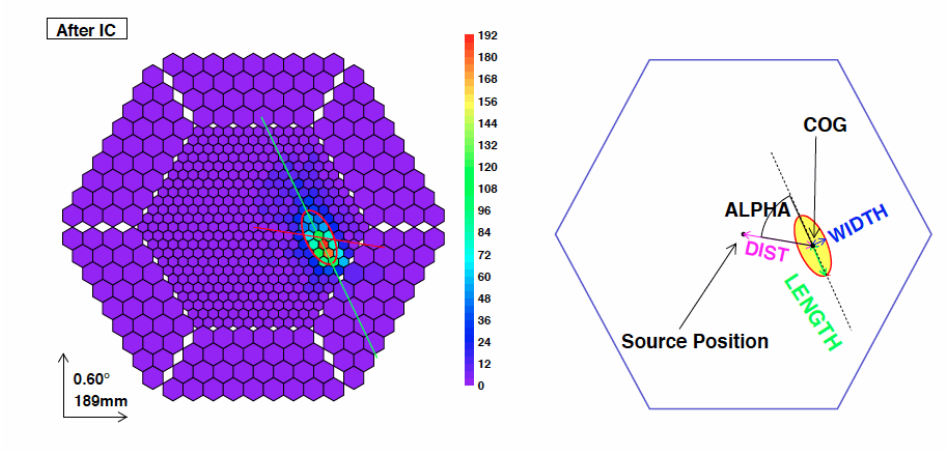


Abbildung 1.6: Beispielhafte Darstellung eines bereinigten Bildes und einiger Bildparameter in der alten MAGIC-Kamera.

Abgesehen von diesen Bildparametern gibt es auch noch Parameter, die zu einem bestimmten Referenzpunkt, z.B. der Quellposition, in der Kamera berechnet werden.

- `Alpha`: `Alpha` bezeichnet den Winkel zwischen der großen Halbachse der Ellipse und der Linie vom CoG zum Referenzpunkt. Dieser Parameter beinhaltet eine große Gamma-Hadron-Trennkraft, da gamma-induzierte Schauer zur Quellposition in der Kamera zeigen und somit alpha klein ist. Hadroninduzierte Schauer sind isotrop in der Kamera verteilt.
- `Dist`: `Dist` ist der Abstand vom CoG zum Referenzpunkt und bietet Informationen über den Abstand von Schauermaximum zur Teleskopachse.

Außerdem gibt es noch einige Parameter, die die Ankunftszeit der Cherenkovphotonen zusätzlich berücksichtigen, wie z.B.:

- `TimeGradient`: Der `TimeGradient` bietet ein Zeitprofil eines Events. Die Pixel werden auf die Hauptachse projiziert. Dann wird ein Graph der Ankunftszeiten der einzelnen Pixel erstellt und mit einer linearen Funktion gefittet. Die Steigung dieser gefitteten Geraden wird dann als Time Gradient bezeichnet.
- `TimeRMS`: So wird der Arrivaltime Spread der Cherenkovphotonen in den Bildpixeln bezeichnet:

$$Time - RMS = \sqrt{\sum_{i=1}^{k}(t_i - t_{mean})^2} \tag{1.2}$$

mit k:Anzahl der Pixel, t_i: Ankunftszeit im i-ten Pixel und t_mean:mittlere Ankunftszeit

Mit Hilfe dieser Bildparameter und der stereoskopischen Bildparameter kann dann im Folgenden die Gamma-Hadron-Separation durchgeführt werden.

1.6 Superstar - Stereoskopische Rekonstruktion der Schauer

Mit Hilfe des Programms Superstar geschieht die stereoskopische Rekonstruktion der Schauerparameter.

Der Kreuzungspunkt der beiden Hauptachsen der projizierten Bilder des Schauers in den beiden MAGIC-Kameras erlaubt einen Rückschluss auf die Ursprungsrichtung des Schauers. Anhand geometrischer Überlegungen können die Schauerachse und der Core Impact Punkt auf der Erde, sowie die beiden individuellen Impactparameter der Teleskope bestimmt werden (siehe Abb.1.7).

Auch die Höhe des Schauermaximums wird in Superstar bestimmt.

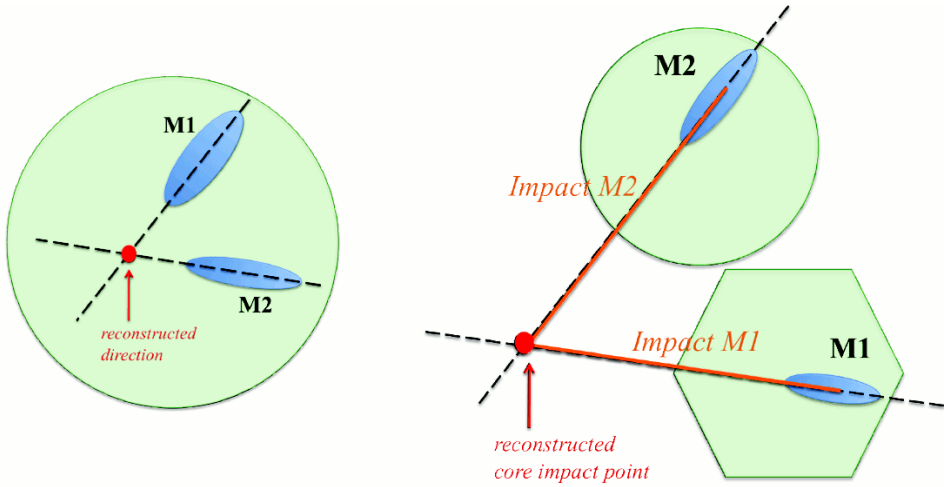


Abbildung 1.7: Rekonstruierte Richtung und rekonstruierter Core Impact Punkt

Des Weiteren werden noch der Cherenkov-Radius und die Cherenkovlichtdichte bestimmt.

Der Cherenkovradius ist der Radius am Boden, der von einem 87MeV Elektron,welches in Schauerrichtung fliegt, in der Höhe des Schauermaximums produziert wird.

Die Cherenkovlichtdichte ist die Lichtdichte am Boden, die von einem 1m langen Track eines 86MeV Elektrons im Schauermaximum produziert wird, welches ebenfalls in Schauerrichtung fliegt.

1.7 Automatische MC-Produktion hier in Dortmund geht so

Die Monte Carlo Produktion an der TU Dortmund geschieht automatisiert mit Hilfe von bash-Skripten und einer mysql-Datenbank siehe 1.8.

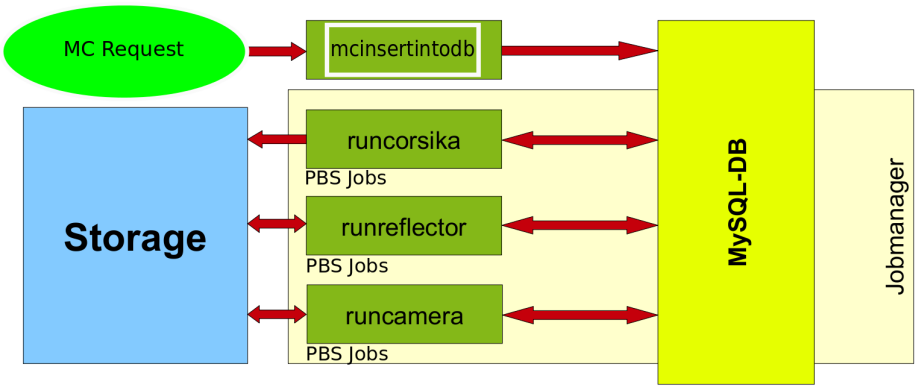


Abbildung 1.8: Schematische Darstellung der automatischen MC Produktionskette.

Bei einer neuen Anfrage für MC Daten werden mit Hilfe des mcinsertintodb-Skripts die für die gewünschte Produktion benötigten Inputcard-Parameter in die Datenbank geschrieben. Im Hintergrund läuft die ganze Zeit das jobmanager-Skript, welches überprüft, ob in der Datenbank ein neuer Auftrag (Job) eingegangen sind. Falls dies der Fall ist, startet das jobmanager-Skript automatisch das runcorsika-Skript, welches wiederum Corsika für diesen neuen Job mit den gewählten Einstellungen startet. Sobald Corsika beendet ist, wird dies in die Datenbank geschrieben, sodass das nächste Programm in der Monte Carlo-Kette gestartet werden kann. Für das nächste Programm in der Kette sind alle benötigten Parameter schon bei der Inauftraggabe des Jobs in der Datenbank gespeichert worden. Also wird Reflector mit Hilfe des runreflector-Skripts gestartet und nach Beendigung und Eintragen in die Datenbank auch Camera.

Nach dem erfolgreichen Durchlauf eines kompletten Jobs bestehend aus 1000-2000 Runs mit je 1000 Events werden die generierten MC Daten nach jedem Programmdurchlauf auf dem Cluster gespeichert. Sobald die Simulationskette für einen Job beendet wurde, wird dies wiederum in die Datenbank eingetragen.

Die Kalibrationskette, die aus den Programmen Sorcerer, Star und Superstar besteht wird analog durchgeführt: Mit Hilfe der run-Skripte werden die Programme gestartet und nach erfolgreichem Durchlauf die Daten gespeichert.

1.7.1 Datenbank

Wie oben beschrieben, werden die wichtigsten Inputparameter und die Pfade zu den MCs in einer mysql-Datenbank gespeichert. Diese Datenbank beinhaltet die Tabellen in Tab.1.1:

Tabelle 1.1: Auflistung der Tabellen, die in der Datenbank existieren....Hmpf!

Tabellen
AtmosphericModel
AzimuthBinning
FADCType
M1CalibrationProcessStatus
M1CameraCopytoGridStatus
M1StarCopytoGridStatus
M2CalibrationProcessStatus
M2CameraCopytoGridStatus
M2StarCopytoGridStatus
MCCalibrationRuns
MCCameraRunData
MCCorsikaRunData
MCJobs
MCPedestalRuns
MCReflectorRunData
MCRunData
MCRunProcessStatus
MCStatistics
MCSuperstarProcessStatus
MCUserID
MarsVersion
ObservationMode
ParticleType
Source
ZenithBinning

Die Tabelle MCJobs bietet eine Übersicht über alle Jobs. Enthalten sind in dieser Tabelle u.a. die JobID, die jedem Job individuell zugewiesen wird, die erste und letzte Runnummer eines Jobs, wann der Job gestartet wurde, wann Camera und Star beendet wurden und den Pfad zu den Daten.
Die Tabellen M1CalibrationProcessStatus und M2CalibrationProcessStatus enthal-

ten ebenfalls die JobID, und die Zeitpunkte wann Camera, Calibration und Star beendet worden. Des Weiteren kann man den Startzeitpunkt und den Zeitpunkt eines Abbruchs des Jobs, sowie den zugehörigen Fehlercode sehen, der Rückschlüsse über die Ursache des Fehlers bietet.

In den Tabellen MCCorsikaRunData, MCReflectorRunData und MCCameraRunData kann man die wichtigsten Inputparameter für die jeweiligen Programme sehen.

Die Tabelle MCRunData bietet eine Übersicht über die Runnummern, die innerhalb der einzelnen Programme verteilt wurden. So gehört zu jeder Runnumber eine CorsikaRunNumber, eine ReflectorRunNumber und eine CameraRunNumber. Also bietet ein Job mit 2000 x 1000 Events Platz für 2000 RunNummern pro Programm.

Die Tabelle MCRunProcessStatus ist eine Übersichtstabelle über die Zeitpunkte zu denen die jeweiligen Inputcards geschrieben und die Programme Corsika, Reflector und Camera beendet wurden.

Die gleiche Tabelle gibt es auch noch für Superstar.

1.7.2 LiDO

Die Monte Carlo-Produktion wird an der TU Dortmund vorwiegend auf dem LiDo-Cluster (Linux Cluster Dortmund) durchgeführt. Für die gesamte MAGIC Kollaboration werden hier die Gamma-MCs produziert und gespeichert.

Dafür stehen insgesamt 3328 CPUs und 215TB Speicher auf dem LiDO zur Verfügung. Die Struktur des Clusters lässt sich Abb.1.9 entnehmen.

Über die beiden Gateway-Server erhält man Zugriff auf die Rechenknoten und den Speicher. Wie in Abb.1.9 zu sehen ist, gibt es verschiedene Knotentypen, die sich in der Zahl der maximalen Jobs pro Queue und Nutzer unterscheiden und über ein Queuingsystem erreichbar sind:

- Die ib-Knoten sind für parallele Jobs reserviert, die die Infiniband interconnect nutzen
- eth-Knoten: Auf diesen Knoten werden serielle und parallele Jobs gerechnet, die die GigabitEthernet Connection nutzen
- quad-Knoten: Für Anforderungen an viel Speicher oder parallele OpenMP/Shared Memory Jobs werden die quad Knoten genutzt
- GPU: Rechnungen auf Graphikkarten finden hier statt.
- nehalem: neue Testknoten mit einer maximaler Walltime von 96h und mehr Arbeitsspecher als die eth-Knoten

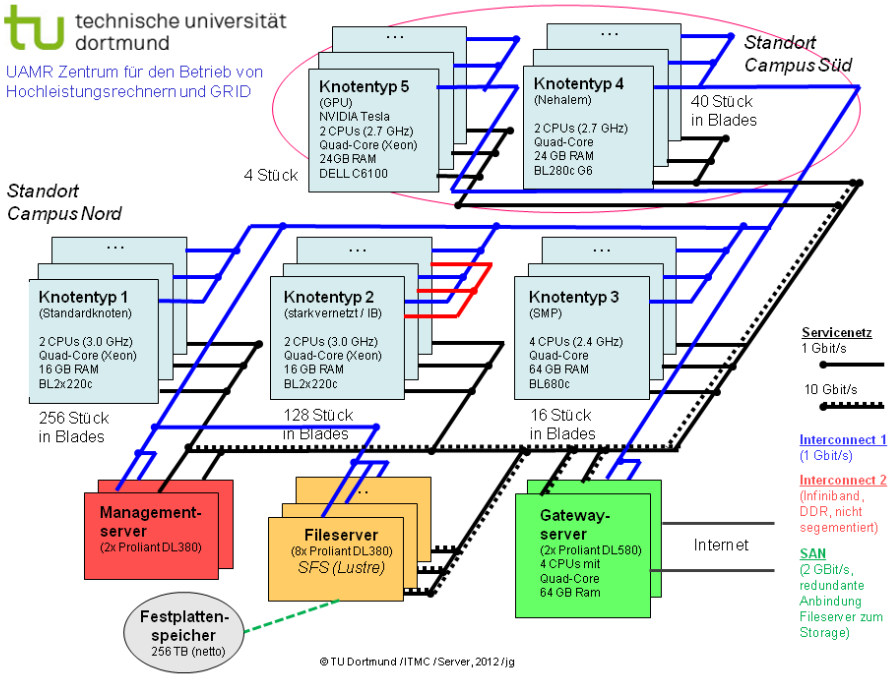


Abbildung 1.9: Schematische Darstellung des Rechenclusters LiDO.

Die meisten Knoten haben drei verschiedene Queues, die sich in ihrer maximal zur Verfügung stehenden Walltime unterscheiden. Es gibt als die short Queues (eth, ib, quad) mit einer maximalen Walltime von 1h, die medium Queues (eth, eth_nhm, ib, quad) mit einer Walltime von 8h und die long Queues mit einer maximalen Walltime von 48h. Falls es doch mal länger dauern sollte, gibt es noch die ultralong Queue (eth) mit einer maximalen Walltime von 2688h. Um also einen Job zu starten, gibt man den benötigten Speicher und eine maximale Walltime an und submittet den Job in das Queuingsystem.

Für die MAGIC MC Produktion werden die eth-Knoten benutzt und pro Run 1000 Corsika Events produziert und weiter prozessiert. Ein Standardjob von 2Mio Corsika Events wird also in 2000 Runs mit je 1000 Events darin aufgeteilt und somit 2000 Runs nacheinander an das Queuingsystem submitted.

Im Moment wird auf dem LiDO folgendes Speichervolumen für die Daten nach den verschiedenen Programmen genutzt:

Alternativ kann auch noch der PhiDo Cluster für Testproduktionen, bzw. früher für dezidierte Protonsimulationen benutzt werden. Dieser Cluster stellt 1200 CPUs und 200TB(ZAHLEN NACHGUCKEN) Speicher zur Verfügung, hat allerdings auch eine größere Auslastung, was dazu führt, dass eine komplette Produktion wesentlich

Tabelle 1.2: Soviel Speichervolumen wird auf dem LiDo benutzt

Programm	Corsika	Reflector	Camera	Sorcerer	Star	Superstar
Speichervolumen [TB]	49	36	9	1.3	0.5	0.5

länger dauert.

1.7.3 Wenn die MC fertig sind...Keine Ahnung, ob das interessant ist...

Sobald eine Produktion an MCs fertig gerechnet ist, werden die Daten ins Grid kopiert. Das heißt im Moment werden sie auf Computer im Rechenzentrum in Spanien PIC kopiert, welche zum Grid gehören und dort gespeichert, sodass sie immer über eine Internetseite erreichbar sind. Die Struktur, in der die MCs gespeichert sind, ist so:

MonteCarlo / Chipsatz der beiden Teloskope / PSF und Mirror fraction / Teilchentyp / Zenitbereich / Observationsart / Standart der Software / Level der Prozessierung / Versionsnummer

`http://data.magic.pic.es/Data/MonteCarlo_Stereo/M1_DRS4_1039_M2_DRS4_1039/M1_PSF10.1_MF0.60_M2_PSF8.6_MF0.66/gammas/za05to35/ringwobble/std20140317/superstar/mc_v07/`

2 MAGIC Standardanalyse

Sowohl die Programme zur Kalibration, als auch die Standard-Analyseprogramme sind im MARS (MAGIC Analysis and Reconstruction Software) -Paket enthalten. Dieses Software Paket ist eine Sammlung an ROOT-Skripten und Macros. Im Folgenden werden die wichtigsten Programme zur Gamma-Hadron-Separation, Lichtkurven-Berechnung und Spektrumsrekonstruktion kurz erklärt.

2.1 Signal-Untergrund-Trennung und Energieschätzung

2.1.1 GH Separation

Da das Signal-Untergrund-Verhältnis zwischen Gamma-Schauern aus der Quelle und hadronischen Schauern kleiner als 1% ist (sogar für helle Quellen), werden gute Verfahren benötigt, um das Signal vom Untergrund zu trennen. In der Standard-MAGIC-Analysekette übernimmt diese Aufgabe das Programm `Coach` und es wird zu diesem Zweck ein Random Forest genutzt. [siehe Breimann] Dieser RF basiert auf einer Sammlung an Entscheidungsbäumen mit zufälligen Parametern in ihren Knoten. Um einen solchen RF zu trainieren, benötigt man ein Trainingsset aus MCs und Untergrunddaten, von denen die Klassenzugehörigkeit (Signal oder Untergrund) genau bekannt ist.

Jedes Event wird durch die in Kapitel 1.5 beschriebenen Imageparameter charakterisiert. Im Ausgangsknoten ist das komplettes Sample mit allen Bildarametern. Dieser Knoten wird dann in 2 Nachfolgeknoten geteilt, indem in einem Bildparameter geschnitten wird. Beim Splitting Prozess werden die Parameter für das Splitten zufällig aus einer vorher begrenzten Menge gezogen und der Parameter mit dem minimalen Gini-Index zum Splitten benutzt. Mit Hilfe des Gini-Index kann die Ungleichheit der beiden Verteilungen als Funktion des Schnittes angegeben werden, der gerade angewendet wurde. Ist der Gini-Index von einem Knoten null, so ist in diesem Knoten nur noch eine Klasse vorhanden.

Dieses Splitten geschieht so lange bis die Anzahl der Events in einem Knoten zu gering wird, oder in einem Knoten nur noch eine Klasse vertreten ist. In diesen Endknoten (terminal nodes) werden dann die Events mit einem Label (Gamma oder Hadron) versehen. Befindet sich in einem Endknoten noch eine Mischung beider Klassen, wird ein Mittelwert vergeben. So folgt jedes Event einem Pfad durch die i

verschiedenen Bäume und wird von allen klassifiziert. Danach wird ihm ein finales Label, die hadroness, zugewiesen:

$$h(Event) = \frac{\sum_{i=1}^{n_{trees}} l_i(Event)}{n_{trees}} \tag{2.1}$$

2.1.2 Energierekonstruktion mit Hilfe von Look up Tables

Die Energie der Primärteilchen ist proportional zur Anzahl der Cherenkovphotonen im Schauer und so zum Parameter `size`. Allerdings ist `size` abhängig vom Zenitwinkel, der Lage des Schauers in der Kamera, dem Impact-Parameter und der Höhe des Schauermaximums. Beruhend auf dieser Tatsache wird nun eine Tabelle erstellt. Das MC Trainingsset wird nun in Bins für jeden Parameter, der für die Energierekonstruktion benutzt werden soll, aufgeteilt. So wird eine mehrdimensionale Tabelle mit der Energie der MC Events, die zu jedem Bin gehört, erstellt. Den realen Daten wird dann anhand ihrer Parameter das passende Energiebin in der Tabelle zugeteilt und so eine estimated Energy zugeordnet.

2.1.3 Position Reconstruction

Ziel ist es, die Herkunft des Primärteilchens zu rekonstruieren. Der Abstand zwischen dem Schauerschwerpunkt und der Quellposition auf der Hauptachse in der Kamera wird mit dem Parameter `Disp` bezeichnet. Es gibt zwei Möglichkeiten, diesen Parameter zu rekonstruieren: Zum Einen sind dies Ghostbusting-Methoden, die die Asymmetrie der Schauer benutzen oder aber RF.

Wird mit zwei Teleskopen observiert, ist die Disp-Rekonstruktion einfacher. Wie in Abb.2.2 zu sehen ist, ist perfektes Ghostbusting möglich.

Nur eine der beiden möglichken Quellpositionen ist kompatibel mit dem anderen Teleskop und so ist die bevorzugte Position der Quelle die, die näher am Schnittpunkt der beiden Hauptachsen liegt und damit ist dann der Parameter `Disp` eindeutig bestimmt.

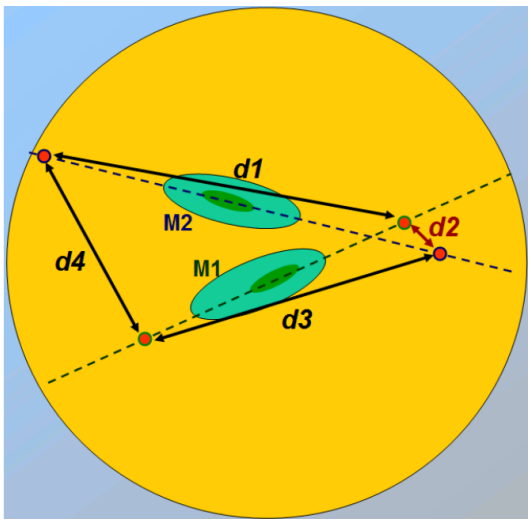


Abbildung 2.1: Rekonstruktion des Parameters `Disp`

2.2 Berechnung der Lichtkurve

Der Gamma-Fluss, d.h. die Rate der Gammateilchen pro Einheitsfläche ist Ausgangsgröße für die Lichtkurve:

$$\Phi = \frac{d^2N}{dSdt}. \tag{2.2}$$

Dafür wird die Anzahl der detektierten Gammas, die effektive Observationszeit und die effektive Fläche des Detektors benötigt. Nach der Energie differenziert ist diese Größe der differentielle Fluss pro Energie:

$$\frac{d\Phi}{dE} = \frac{d^3N}{dSdtdE}, \tag{2.3}$$

bzw. der integrale Fluss:

$$\Phi_{E>500GeV} = \int_{500GeV}^{\infty} \frac{d\Phi}{dE} dE. \tag{2.4}$$

Die zeitliche Entwicklung des integralen Flusses wird nun Lichtkurve genannt. Das Binning kann nach Lust und Laune gewählt werden.

2.2.1 Anzahl der exzess gammas

Um die Anzahl der Gammateilchen aus der Quelle zu bestimmen, wird ein θ^2-Histogramm benutzt. Dies ist ein Histogramm der quadrierten Entfernungen zwischen der rekonstruierten Quellposition und der nominalen Quellposition. Gammas aus der Quelle

haben ein kleineres theta während der Background eine annähernd flache Verteilung liefert.

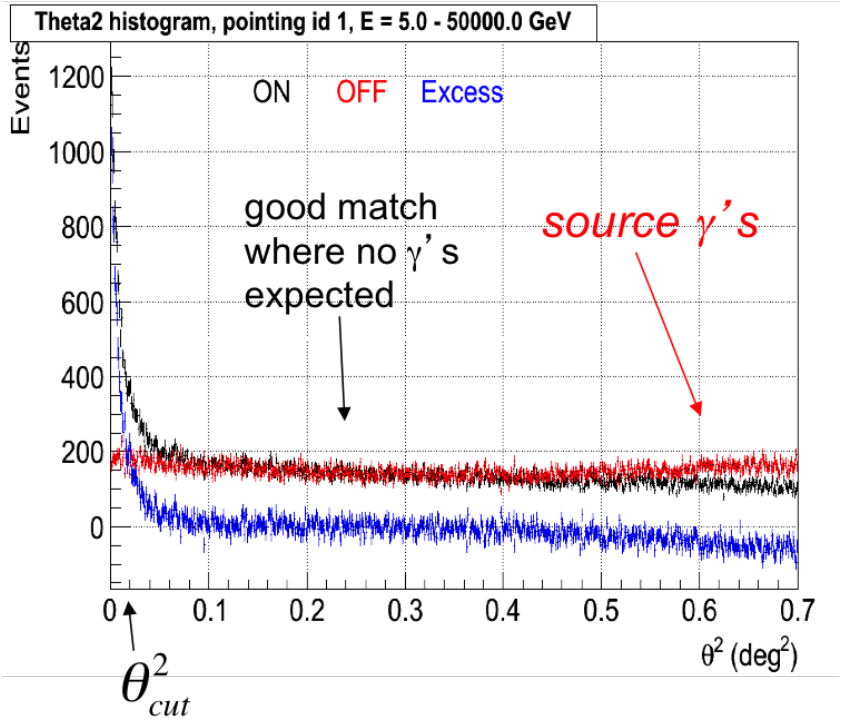


Abbildung 2.2: θ^2-Verteilung... blabla

Um die Anzahl der wirklich Exzess-Events zu kriegen, müssen von den Events aus der Quellrichtung noch die Background Events abgezogen werden und es wird meist noch ein Schnitt in θ^2 gemacht.

Die Beobachtungsmethode von MAGIC ist die Wobble-Beobachtung, d.h. das Teleskop ist nicht direkt auf die Quelle ausgerichtet, sondern einen kleinen Winkel daneben. Dank der alt-azimutalen Montierung rotiert die Quelle um das Zentrum in der Kamera und es ist möglich einen Punkt gegenüber der Quelle als Off-Position zu benutzen. Vorteil dieser Methode ist, dass es keine separate Off-Datennahme geben muss. Allerdings tauchen so die Quellgammas auch in der Off-θ^2-Verteilung auf, haben aber ein großes θ^2. Eine Off-Position, die zu nahe an der Quelle ist, ist nicht zu empfehlen.

2.2.2 Effektive On Zeit

Die effektive On-Time berücksichtigt die Totzeit in der Datennahme. Abhängig vom Chip ist die Totzeit bei den aktuellen Chips bei 26μs.

2.2.3 Effektive Fläche

Als effektive Fläche wird die Fläche am Boden, die orthogonal zur Herkunftsrichtung der Schauerteilchen ist, bezeichnet. Die Größe dieser effektiven Fläche ist abhängig von der Energie und dem Zenitwinkel des Schauers. In MARS wird diese Größe mit Hilfe von MCs folgendermaßen berechnet:

$$A_{eff}(E) = \frac{N_{\gamma,final}}{N_{\gamma,simulated}} A_{MC,total} \tag{2.5}$$

Dafür wird eine bestimmte Anzahl an Gammas ($N_{\gamma,simulated}$) auf einer uniformen Fläche $A_{MC,total}$ simuliert. Die Größe $N_{\gamma,final}$ ist die Anzahl der Gammas, die das Cleaning und alle Cuts überlebt hat.

2.3 Entfaltung des Energiespektrumpfs

Bei der Messung mit IACTs handelt es sich um eine indirekte Messung. Die Energie des Schauer-auslösenden Teilchens ist nicht direkt messbar. Die Bildparameter und damit auch die geschätzte Energie E_{est} haben eine begrenzte Auflösung und erfordern die Methode der Entfaltung.

Die Probleme, die bei der Messung auftreten sind:

- Begrenzte Akzeptanz: Nicht alle Schauer, die ein Teilchen auslöst können vom Teleskop detektiert werden.
- Indirekte Messung: Da eine direkte Messung nicht möglich ist, wird anhand von gemessenen Parametern, wie der Größe des Schauers in der Kamera etwa, mit Hilfe eines RF die Energie geschätzt. Die Vorraussetzung dafür ist, dass diese in echt gemessenen Parameter stark mit der Energie korrelliert sind.
- Begrenzte Auflösung: Es ist nur möglich mit begrenzter Genauigkeit aus den Bildparametern die Energie zu rekonstruieren, d.h. es existiert eine Migration von Events. Wird die geschätzte Energie gegen die reale Energie aufgetragen, erhält man eine verschmierte Diagonale (siehe Abb.2.3)

Durch die Methode der Entfaltung können diese Probleme berücksichtigt werden und das Problem lässt sich mit einer Fredholmschen Integralgleichung darstellen:

$$g(y) = \int_c^d M(x,y) f(x) dx + b(y) \tag{2.6}$$

mit g(y): gemessene Verteilung, f(x): gesuchte Verteilung, M(x,y): Migrationsmatrix bestimmt auf MCs, b(y): Background-Verteilung

Diese Gleichung lässt sich auch bin-Weise darstellen:

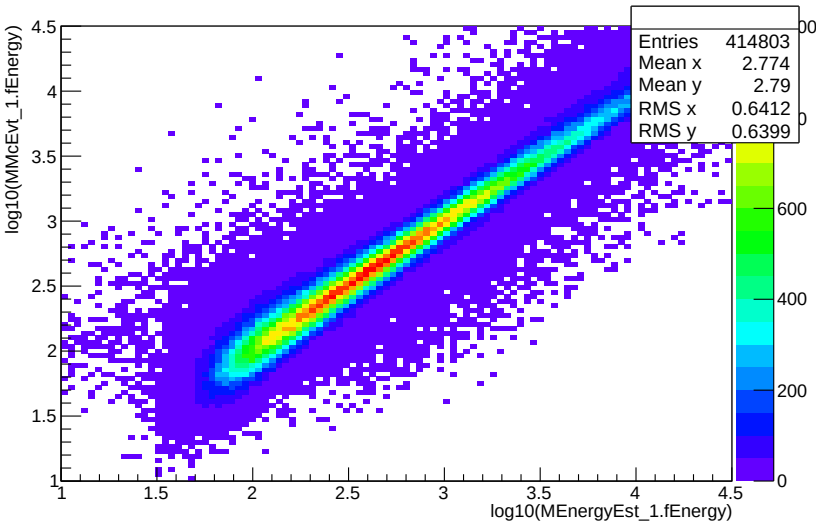


Abbildung 2.3: Estimated Energy gegen True Energy.

$$g_i = \sum_j M_{ij} f_j + b_i, \tag{2.7}$$

wobei M_{ij} nun die Wahrscheinlichkeit ist, dass ein Event in bin j in bin i gezählt wird.

Das Ziel der Entfaltung ist nun die wahre Verteilung f zu finden. Die Kovarianzmatrix der gesuchten Verteilung ergibt sich dann mit der Kovarianzmatrix der gemessenen Verteilung zu:

$$\boldsymbol{V}[\boldsymbol{f}] = \boldsymbol{M}^{-1}\boldsymbol{V}[\boldsymbol{g}](\boldsymbol{M}^{-1})^T \tag{2.8}$$

Da die Invertierung der Migrationsmatrix oft zu oszillierenden Lösungen führt, versucht man die Methode der kleinsten Quadrate.

$$\chi_0^2 = (\vec{g} - \boldsymbol{M}\vec{f})^T \boldsymbol{V}^{-1}[\vec{g}](\vec{g} - \boldsymbol{M}vecf). \tag{2.9}$$

Dies gilt nur für Gaussverteilte Daten, also nicht für Bins mit kleinen Eventszahlen. Für diese muss nun die Poisson-Statistik benutzt werden und der Log-Likelihood-Ausdruck minimiert werden:

$$L_0(a) = \sum_i (g_i(a) - g_{i,m} \cdot \ln g_i(a)). \tag{2.10}$$

Außerdem ist es nötig, eine Regularisierung einzuführen, um die kleinen Ausdrücke in der Migrationsmatrix, die während der Entfaltung verstärkt werden, zu unterdrücken. Durch Einführung eines Regularisierungsterms werden Anforderungen an die Lösung gestellt, bei zu doller Regularisierung aber auch ein Bias eingeführt.

Im Allgemeinen wird Regularisierung durch Addition eines Regularisierungsterms gemacht, sodass:

$$\chi^2 = \chi_0^2 + \frac{\tau}{2} Reg(f). \tag{2.11}$$

Verschiedene Arten der Regularisierung können in der Analyse gewählt werden. Es ist ebenfalls möglich, eine Vorwärtsfaltung durchzuführen, wobei ein bestimmtes Modell als Annahme gewählt wird und freie Parameter dieses Modells bestimmt werden. Zum Testen ist dies eine gute Alternative, allerdings keine richtige Entfaltung, da das Ergebnis Modellabhängig bleibt.

Eidesstattliche Versicherung

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit mit dem Titel „LaTeX-Vorlage für die Bachelorarbeit in TU-Farben“ selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum	Unterschrift

Belehrung

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50 000 e geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden (§63 Abs. 5 Hochschulgesetz -HG-).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen.

Ort, Datum	Unterschrift