IEKP-Bachelor_2016_11

Karlsruhe Institute of Technology

Verbesserung der Benutzerschnittstelle des Millepede II Alignment-Algorithmus für den CMS-Spurdetektor

Martin Descher

Bachelorarbeit

Martin Descher

An der Fakultät für Physik
Institut für Experimentelle Kernphysik (IEKP)

Erstgutachter:	Prof. Dr. Ulrich Husemann
Zweitgutachter:	Dr. Matthias Schröder

Karlsruhe, 20. Juni 2016

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, 20. Juni 2016

.....

(**Martin Descher**)

Inhaltsverzeichnis

1. Einleitung

Die Erforschung der fundamentalen Teilchen des Universums erfordert aufwändige Experimente. Eines dieser Projekte ist das CMS-Experiment am Teilchenbeschleuniger LHC in der Schweiz. Mit dem CMS-Teilchendetektor wird eine hohe Rate an Teilchenkollisionen aufgenommen und die Kollisionsprodukte werden mit sehr hoher Präzision gemessen.

Um die erforderte Messgenauigkeit zu gewährleisten, müssen die einzelnen Detektorteile sehr gut kalibriert werden. Ein zentraler Bestandteil des CMS-Detektors ist ein Spurdetektor, der aus über 25 000 Siliziumsensoren besteht und mit dessen Hilfe Spuren geladener Teilchen mit hoher Präzision vermessen werden können. Um die für die Physik-Analyse angestrebte Genauigkeit zu erreichen, ist es unter anderem notwendig die Positionen der einzelnen Sensoren auf wenige 10 µm und besser zu bestimmen, weil sich eine Unsicherheit in der Position auf die der Spurpunktmessungen auswirkt. Da es sich um eine räumliche Kalibrierung handelt, ist in diesem Zusammenhang der Begriff „Alignment“ geläufig. Erreicht wird das Kalibrierungsziel mit einer spurbasierten Alignment-Prozedur. Bei dieser wird anhand gemessener Teilchenspuren die Geometrie des Spurdetektors mit Algorithmen berechnet, indem die Messpunkt-Teilchenspur-Residuen minimiert werden. Für eine unabhängige Validierung des Spurdetektor-Alignments werden bei CMS zwei Alignment-Algorithmen verwendet. Einer dieser Algorithmen ist der *Millepede II Alignment-Algorithmus*.

Das Alignment ist keine einmalige Aufgabe, denn aufgrund der geforderten Präzision können schon kleinere Einflüsse wie Temperatur und Magnetfeld signifikante Abweichungen hervorrufen. Obwohl regelmäßig Alignments durchgeführt werden müssen, ist der praktische Umgang mit Millepede II aus einer Anwenderperspektive verbesserungswürdig. Das zur Konfiguration und Steuerung von Millepede II programmierte *Millepede Production System* (MPS) ist veraltet und teilweise kompliziert zu handhaben. In dieser Arbeit wird das MPS an die aktuellen Anforderungen angepasst.

In Kapitel 2 wird zunächst die experimentelle Umgebung, also der Teilchenbeschleuniger LHC und der CMS-Detektor, beschrieben. Der Silizium-Spurdetektor wird in Kapitel 3 im Detail präsentiert.

In Kapitel 4 wird die allgemeine Problemstellung des spurbasierten Alignments diskutiert und im darauf folgenden Kapitel 5 wird dessen Lösung mit dem Millepede-Algorithmus erläutert.

Kapitel 6 beschreibt das bereits erwähnte *Millepede Production System*. Einerseits werden in diesem Kapitel die Neuerungen innerhalb des MPS diskutiert. Andererseits soll

das Kapitel als Dokumentation für das MPS dienen, da es bisher nur wenige ausführliche Hilfen oder Leitfäden zu diesem Programm gibt.[1] Anschließend werden in Kapitel 7 Validierungsstudien zu den Änderungen am MPS präsentiert.

[1] Der Ablauf von Millepede-Alignments wird bisher hauptsächlich in persönlichen Gesprächen vermittelt.

2. Experimentelle Umgebung

2.1. Der große Hadronen-Speicherring LHC

Der große Hadronen-Speicherring LHC („**L**arge **H**adron **C**ollider“) ist der derzeit größte Teilchenbeschleuniger der Welt. Mit ihm werden entweder Bleiionen oder Protonen beschleunigt, wobei für Proton-Proton-Kollisionen momentan eine Schwerpunktsenergie von $\sqrt{s} = 13\,\text{TeV}$ erreicht wird. Als Konstruktionsgrundlage für den LHC diente der 27 km lange Tunnel des ehemaligen LEP-Beschleunigers („**L**arge **E**lectron-**P**ositron Collider“) am Standort der Europäischen Organisation für Kernforschung CERN in der Nähe von Genf. Die Informationen in diesem Kapitel basieren auf [1].

Der Beschleuniger besitzt zwei evakuierte Strahlröhren für die entgegengerichtete Beschleunigung zweier Hadronenstrahlen. Mittels supraleitender Dipolmagnete werden die Teilchen auf der Ringbahn gehalten. Zur Fokussierung der Strahlen werden Quadrupolmagnete verwendet. Dabei werden die Magnete mit flüssigem Helium auf einer Temperatur unter 2 Kelvin gehalten. Bevor die Hadronen in den LHC eingespeist werden, durchlaufen sie eine Reihe von Vorbeschleunigern, mit denen am CERN schon vor den Zeiten des LHC Experimente betrieben wurden. Der gesamte Beschleunigerkomplex des CERN ist in Abbildung 2.1 dargestellt.

An vier Punkten des LHC-Beschleunigerringes werden die zwei Strahlen zur Kollision gebracht. An diesen Punkten befinden sich verschiedene Detektoren, die auch als Experimente bezeichnet werden. Diese heißen ALICE („**A** **L**arge **I**on **C**ollider **E**xperiment“), LHCb („**L**arge **H**adron **C**ollider **b**eauty“), ATLAS („**A** **T**oroidal **L**HC **A**pparatu**S**“) und CMS („**C**ompact **M**uon **S**olenoid“).

Neben der Schwerpunktsenergie ist auch die Luminosität L eine wichtige Kenngröße von Teilchenbeschleunigern. Sie ist ein Maß für die Kollisionsrate des Beschleunigers. Multipliziert man die Luminosität mit dem Wirkungsquerschnitt σ einer Teilchenwechselwirkung, erhält man die Rate der zu erwartenden Ereignisse dieser Wechselwirkung.

$$\frac{dN}{dt} = L \cdot \sigma \ . \tag{2.1}$$

Der LHC erreicht Luminositäten im Bereich von $L \approx 10^{34}\,\text{cm}^{-2}\text{s}^{-1}$. Die über die Datennahmezeit integrierte Luminosität ist ein Maß für die aufgenommene Datenmenge.

$$L_{\text{int}} = \int L dt \ . \tag{2.2}$$

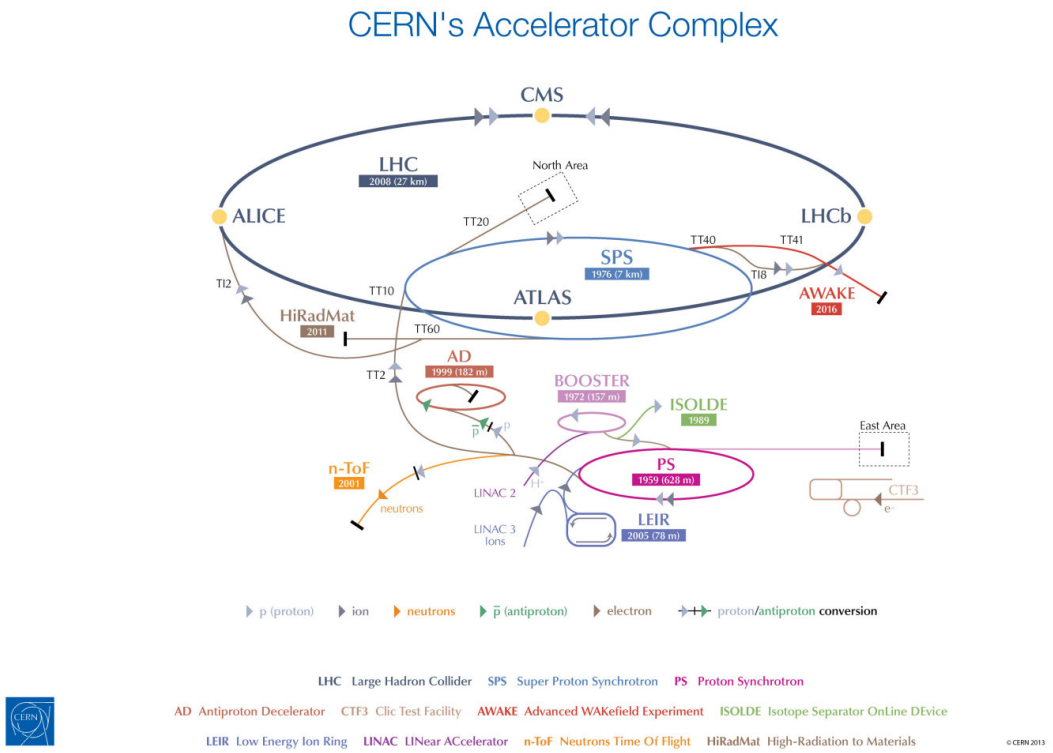


Abbildung 2.1.: Der Beschleunigerkomplex des CERN [2].

2.2. Der CMS-Detektor

Der CMS-Detektor („**C**ompact **M**uon **S**olenoid“) ist ein Mehrzweckdetektor am LHC, der für eine Vielzahl von Untersuchungen eingesetzt wird. Obwohl der Detektor das englische Wort für „kompakt“ im Namen trägt, ist er mit einer Länge von 21 m, einem Durchmesser von 15 m und einem Gewicht von 14 000 Tonnen [3] eine sehr große Maschine. In diesem Kapitel soll die Funktionsweise des CMS-Detektors umrissen werden, wobei die präsentierten Fakten [4, 5] folgen.

Der CMS-Detektor hat die Form eines Zylinders, dessen Achse sich mit der Strahlachse deckt, und im Zentrum des Zylinders befindet sich der Kollisionspunkt. Die einzelnen Detektorsysteme des CMS sind ähnlich den Schalen einer Zwiebel in Lagen um die Strahlachse angeordnet. Diese sind von innen nach außen der Silizium-Spurdetektor, das elektromagnetische Kalorimeter, das hadronische Kalorimeter und die Myonkammern.

Für CMS wird ein rechtshändiges Koordinatensystem definiert, dessen Ursprung am nominalen Kollisionspunkt liegt. Die x- und y-Achse spannen eine Ebene orthogonal zur Strahlachse auf. Dabei zeigt die x-Achse zum Mittelpunkt des LHC-Ringes und die y-Achse senkrecht nach oben. Die z-Achse zeigt dementsprechend entlang der Strahlachse und zwar entgegen dem Uhrzeigersinn, wenn man sich den LHC von oben vorstellt. Oft werden auch der Abstand $r = \sqrt{x^2 + y^2}$ von der Strahlachse, der Azimutalwinkel ϕ (von der positiven x-Achse aus gemessen) und der Polarwinkel θ (von der positiven z-Achse aus gemessen) verwendet. Anstatt des Polarwinkels wird für Teilchenbahnen jedoch meistens die Pseudorapidität η angegeben.

$$\eta = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right] \quad . \tag{2.3}$$

Diese ist Null für Teilchen, die senkrecht zur Strahlachse fliegen und geht gegen $\pm\infty$ für Teilchen, die sich entlang der Strahlachse bewegen. Die verschiedenen Detektorsysteme des CMS sind so angeordnet, dass auch Teilchen mit betragsmäßig hohen Pseudorapiditäten gemessen werden können. Maximal werden Teilchen mit $|\eta| \leq 5$ erfasst, womit nahezu der gesamte Raumwinkel um den Kollisionspunkt abgedeckt wird.

Das Herzstück des CMS ist eine 13 m lange, supraleitende Magnetspule mit einem Durchmesser von 6 m, die eine Feldstärke von 3,8 Tesla erreicht. Die Magnetspule umschließt den Spurdetektor, das elektromagnetische Kalorimeter und das hadronische Kalorimeter. Die Myonkammern liegen außerhalb. Das Magnetfeld zeigt entlang der Strahlachse und ermöglicht so die Messung des Transversalimpulses hochenergetischer, geladener Teilchen über die Krümmung der Teilchenspur aufgrund der Lorentzkraft. Der Aufbau des CMS ist in Abbildung 2.2 dargestellt. Im folgenden werden die einzelnen Detektorsysteme erläutert (siehe auch Abbildung 2.3).

- **Silizium-Spurdetektor**
  Der Silizium-Spurdetektor ist der innerste Teil des CMS und deckt Pseudorapiditäten von $|\eta| \leq 2{,}5$ ab. Er ist aus Siliziumsensoren aufgebaut, die einen Strom messen, wenn sich ein geladenes Teilchen durch sie hindurch bewegt. Mit sehr vielen dieser Sensoren werden von den Teilchen genug Spurpunkte hinterlassen, dass sich die Teilchenspuren rekonstruieren lassen. Anhand dieser Spuren werden beispielsweise die Impulse und die Vertizes der Kollisionen bestimmt. Die Wahl eines Siliziumdetektors führt zu einem relativ hohen Materialbudget (0,4 Strahlungslängen bei $\eta = 0$) [4], was bei der Rekonstruktion von Elektronen und Photonen berücksichtigt werden muss. In Kapitel 3 wird der Aufbau des Spurdetektors tiefer erklärt.

- **Elektromagnetisches Kalorimeter**
  Zur Messung der Energie von elektromagnetisch wechselwirkenden Teilchen wie Elektronen, Positronen und Photonen werden diese im elektromagnetischen Kalorimeter absorbiert. Unter Bildung elektromagnetischer Schauer werden Szintillatorkristalle aus Bleiwolframat angeregt. Beim Abregen emittieren die Kristalle Licht, welches von Silizium-Lawinenphotodioden gemessen wird. Aus der Anzahl der emittierten Photonen lässt sich die Energie des eingegangenen Teilchens bestimmen.

- **Hadronisches Kalorimeter**
  Das hadronische Kalorimeter ist ein „Sampling"-Kalorimeter und besteht als solches aus abwechselnden Schichten Absorbermaterial (Messing) und Plastikszintillatoren. Im Absorber werden durch eingehende Hadronen Teilchenschauer ausgelöst. Die entstehenden Teilchen regen den Szintillator an, dessen emittiertes Licht anschließend von Hybrid-Photodioden gemessen wird. Daraus lässt sich wiederum die Energie des ursprünglichen Hadrons bestimmen.

- **Myonkammern**
  Da Myonen aufgrund ihrer hohen Masse wenig Energie in den vorangehenden Detektoren hinterlassen, sind zur ihrer Identifikation die äußeren Myonkammern nötig. In diesen werden drei Arten von Detektoren verwendet, die sich in ihrer Funktionsweise ähneln. Das Messprinzip beruht auf der Anregung von Gasen, wodurch es zur Bildung von Ionen und Elektronen kommt. Diese werden von Kathoden und Anoden angezogen und es fließt ein elektrischer Strom. Aus den Spurpunkten kann die Teilchenspur und so der Impuls rekonstruiert werden. Im inneren Spurdetektor hinterlassen die Myonen zwar auch Spuren, doch erst die Myonkammern erlauben eine Identifikation als Myonen, da alle anderen detektierbaren Teilchen bereits in den vorigen Detektoren absorbiert werden.

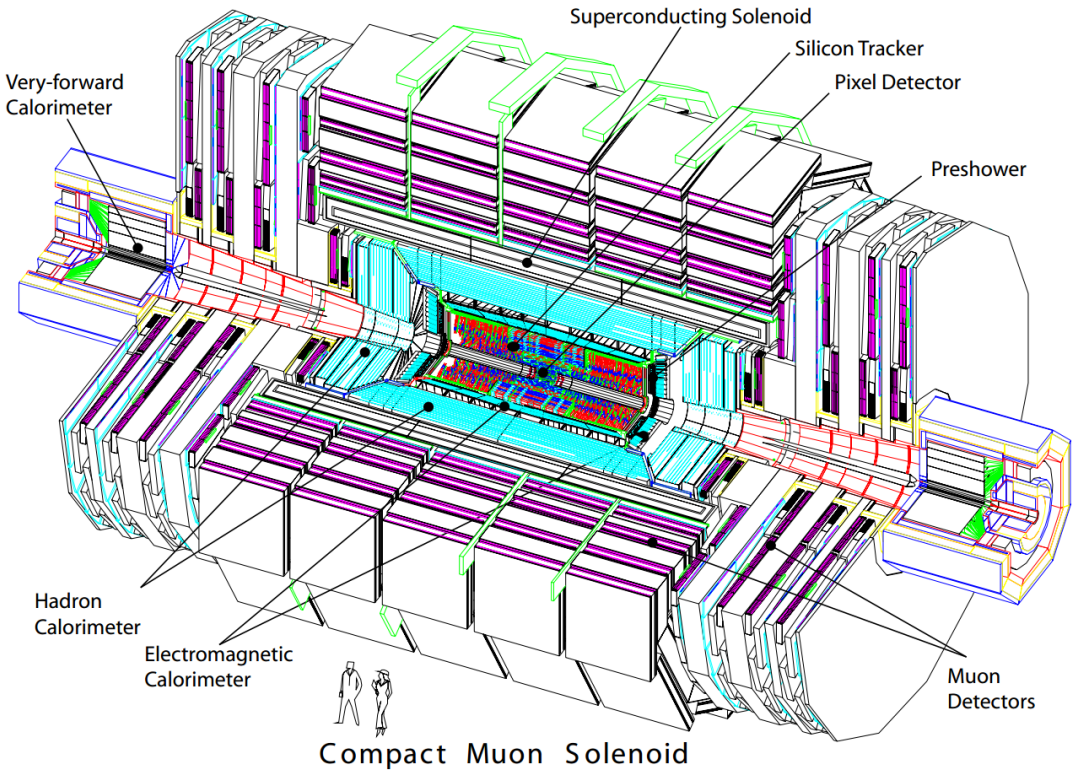


Abbildung 2.2.: Längsschnitt durch den CMS-Detektor [4].

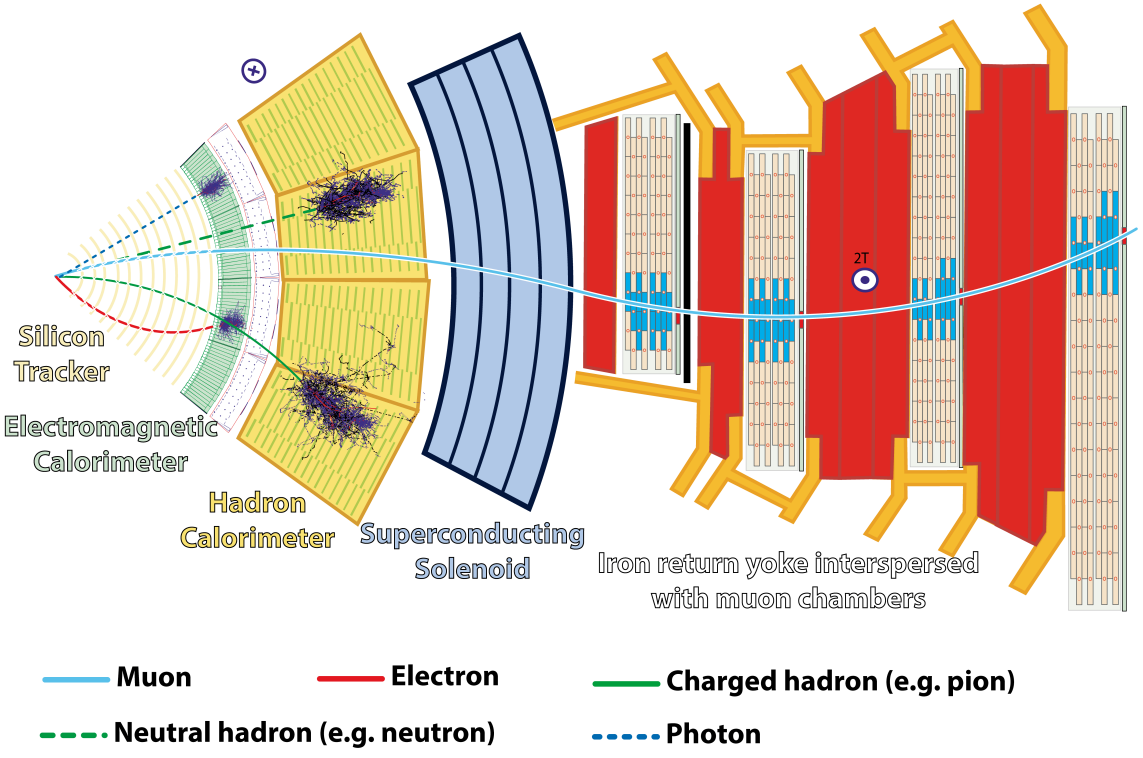


Abbildung 2.3.: Schematischer Querschnitt durch den CMS zur Veranschaulichung der Detektorsysteme [6].

3. Der Spurdetektor des CMS-Experiments

Der CMS-Spurdetektor [7] besteht aus insgesamt 25 644 Silizium-Sensoren und ist mit einer Länge von 5,4 m und einem Durchmesser von 2,5 m der größte Silizium-Spurdetektor, der bislang gebaut wurde. Er ist unterteilt in einen Silizium-Pixeldetektor und einen Silizium-Mikrostreifendetektor, die ihrerseits weitere Substrukturen aufweisen. Der Aufbau des Spurdetektors ist in Abbildung 3.1 anhand eines Querschnitts dargestellt.

Der innere Silizium-Pixeldetektor besteht aus dem „barrel pixel detector“ (PXB) und dem „forward pixel detector“ (PXF). Die Sensormodule des PXB sind in drei zylindrischen Lagen mit den Radien 4,4 cm, 7,3 cm und 10,2 cm um die Strahlachse angeordnet. Die Module des PXF bilden Endkappen aus jeweils zwei ringförmigen Lagen auf jeder Seite. Auf den 1440 Modulen des Pixeldetektors befinden sich insgesamt etwa 66 Millionen Siliziumpixel der Maße 100 µm × 150 µm. Neben der Messung des Transversalimpulses geladener Teilchen ist der Pixeldetektor wichtig für die Vertexbestimmung. Diese ermöglicht Teilchenidentifikationen mittels der Bestimmung von Sekundärvertizes und verringert unter anderem die Auswirkungen des „Pileup“-Effekts, also der gleichzeitigen Kollision von mehr als zwei Protonen. Aus diesem Grund sind die Module sehr nah am Kollisionspunkt angeordnet. Dies hat zur Folge, dass die Module wegen des hohen Teilchenflusses sehr strahlenhart sein müssen.

Der Silizium-Mikrostreifendetektor umschließt den Pixeldetektor und ist ebenfalls in zylindrische Substrukturen und Endkappen aufgeteilt. Der innere Teil besteht aus dem „tracker inner barrel“ (TIB) und den „tracker inner disks“ (TID). Umschlossen werden diese wiederum vom „tracker outer barrel“ (TOB) und den „tracker endcaps“ (TEC). Die 15 148 Streifenmodule sind mit mit insgesamt 24 204 Sensoren bestückt. Diese tragen insgesamt 9,6 Millionen längliche Silizium-Mikrostreifen mit Breiten zwischen 80 und 205 µm [8].

Alle Substrukturen (PXB, PXF, TIB, TID, TOB, TEC) sind für das Alignment hierarchisch in weitere Strukturen unterteilt. Hauptsächlich sind diese an Haltestrukturen orientiert, aber teilweise sind sie auch von der Softwareseite motiviert und weichen vom physischen Detektor ab. Beispielsweise gibt es für die einzelnen Lagen des TOB keine eigenen Stützstrukturen[9]. Eine Übersicht der Substruktur-Hierarchie ist in Abbildung 3.2 gegeben.

Um das Potential des Detektors voll auszuschöpfen, müssen die Positionen und Ausrichtungen der Module sehr genau bekannt sein. Im Laufe des Betriebes ändert sich die Anordnung

der Module häufig aufgrund verschiedenster Einflüsse wie Temperaturänderungen, dem angelegten Magnetfeld oder Wartungsarbeiten. Der Detektor wird bei Temperaturen unter $-10\,^{\circ}\mathrm{C}$ betrieben und muss deshalb nach Wartungsarbeiten heruntergekühlt werden und auch das Magnetfeld muss manchmal ab- und angeschaltet werden. Jeder solcher Eingriff kann Auswirkungen auf die Geometrie des Spurdetektors haben, woraufhin der Detektor neu kalibriert werden muss.

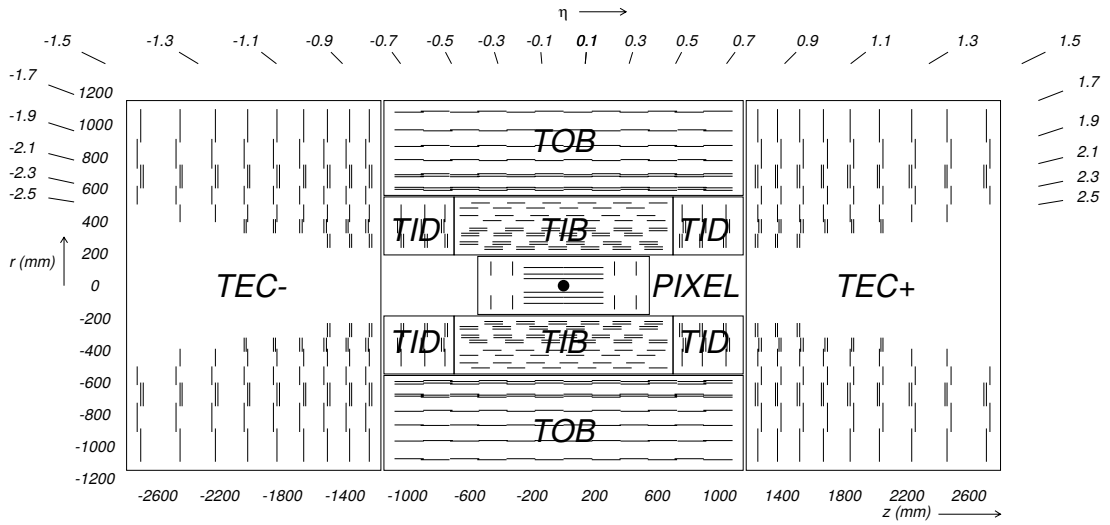


Abbildung 3.1.: Schematischer Querschnitt durch den CMS-Spurdetektor mit Bezeichnungen der größeren Substrukturen [4].

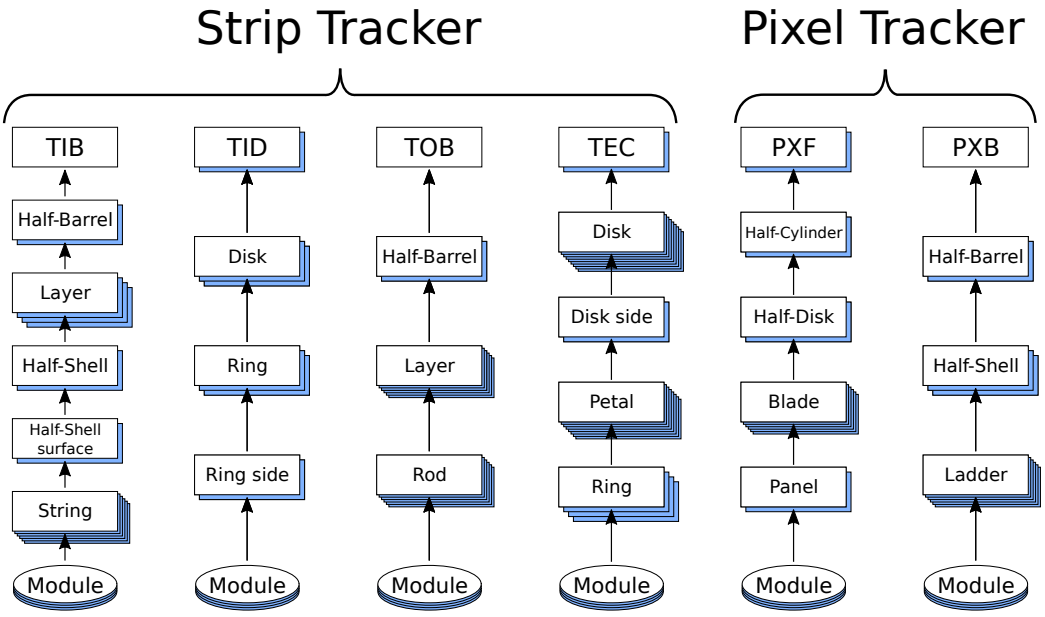


Abbildung 3.2.: Hierarchie der Substrukturen des Mikrostreifen-Detektors und des Pixel-Detektors basierend auf [8, 10].

4. Spurbasiertes Alignment

Unter dem Begriff „Alignment“ versteht man in der Experimentalphysik die räumliche Kalibrierung eines Messgerätes. Im Falle des CMS-Spurdetektors können z.B. Temperaturschwankungen oder das angelegte Magnetfeld die Geometrie des Detektors verändern. Darüber hinaus kommt es wegen begrenzter Präzision beim Zusammenbau des Detektors zu Abweichungen von der Sollgeometrie. In diesem Kapitel soll begründet werden, weshalb das Alignment des CMS-Spurdetektors wichtig ist und wie es mithilfe gemessener Teilchenspuren realisiert wird. In Kapitel 4.1 wird das allgemeine Prinzip des spurbasierten Alignments erläutert und im darauf folgenden Abschnitt 4.2 wird der Umgang mit bestimmten systematischen Unsicherheiten des spurbasierten Alignments, den sogenannten schwachen Moden, beschrieben.

Die Geometrie des Spurdetektors wird durch Parameter für Position, Rotation und Verformung jedes Sensors (lower level alignables) und weitere Parameter für die größeren Strukturen (higher level alignables) beschrieben. Wie in Kapitel 3 angesprochen, bezeichnen diese größeren Strukturen beispielsweise die Halbschalen des PXB oder TIB. Die Gesamtheit der Parameter, die die Geometrie des Spurdetektors angeben, nennt man Alignmentparameter. Nur mit sehr genauen Informationen über die Geometrie ist es möglich, die Teilchenspuren präzise genug zu rekonstruieren, um die Anforderungen des Experiments zu erfüllen. Die Rekonstruktion der Teilchenspuren ist besonders wichtig bei der Berechnung des Transversalimpulses geladener Teilchen und bei der Vertexbestimmung [11, 12]. Eine ungenaue Bestimmung der Geometrie kann somit z.B. Auswirkungen auf die Identifikation von b-Jets haben [13]. Die Auflösung der Spurpunktbestimmung beim CMS-Spurdetektor beträgt bis zu 10 µm in der $r\phi$-Koordinate [11]. Eine angemessene Alignment-Strategie soll die Alignmentparameter auf <10 µm genau bestimmen können, was weitgehend erreicht oder oft auch signifikant übertroffen wird [5].

Eine Schwierigkeit ist die große Anzahl der zu bestimmenden Parameter. Für jeden der etwa 25 000 Sensoren des Spurdetektors werden bis zu 9 Parameter gesucht. Diese entsprechen drei Freiheitsgraden der Translation, drei der Rotation und zusätzlich drei Freiheitsgraden für Verformungen. Das führt zu einer Parameteranzahl in der Größenordnung 10^5. Äußere Messmethoden, zum Beispiel mit dem optischen Alignment-System des CMS [14], sind aufgrund der Genauigkeitsanforderungen und wegen mangelndem Zugang zum Spurdetektor nicht ausreichend. Daher werden für die Bestimmung der Alignmentparameter die gemessenen Teilchenspuren selbst verwendet. Außer den Spuren benötigt man für das spurbasierte Alignment hinreichend genaue Startwerte (Startgeometrie) für die gesuchten Parameter, viel Rechenkapazität und einen geschickten Algorithmus.

4.1. Prinzip der Parameterbestimmung

Die Vorgehensweise beim spurbasierten Alignment lässt sich wie folgt zusammenfassen: Der Detektor misst Spurpunkte (auch Hitpositionen) i von geladenen Teilchen aus Kollisionen oder kosmischer Strahlung. Es wird dann eine Startgeometrie angenommen, mit der Teilchenspuren („Tracks“) an die Spurpunkte angepasst werden („Fit“). Die von der angepassten Teilchenspur j vorhergesagten Spurpunkte y_i weichen jedoch von den gemessenen Spurpunkten m_i ab. Die jeweiligen Abstände dieser Punkte nennt man Residuen. Die Residuen werden auf die Messunsicherheit σ_{ij} der Spurpunkte normiert. Zuletzt werden die Modulparameter $\mathbf{p}$ und die Parameter der Teilchenspur $\boldsymbol{\tau}$ als frei angesehen und so verändert, dass die normierten Residuen r_{ij} minimiert werden. Dies entspricht einem χ^2-Minimierungsproblem.

$$\chi^2(\boldsymbol{\tau}, \mathbf{p}) = \sum_{j}^{\text{Spuren}} \sum_{i}^{\text{Spurpunkte}} r_{ij}^2 = \sum_j \sum_i \left(\frac{m_i - y_i(\boldsymbol{\tau_j}, \mathbf{p})}{\sigma_{ij}} \right)^2 . \tag{4.1}$$

Nimmt man an, dass die Messung der Spurpunkte normalverteilt ist, entspricht diese Methode der Wahl der Alignmentparameter mit der größten Likelihood für einen Satz gegebener Messpunkte. Das Prinzip ist in Abbildung 4.1 veranschaulicht.

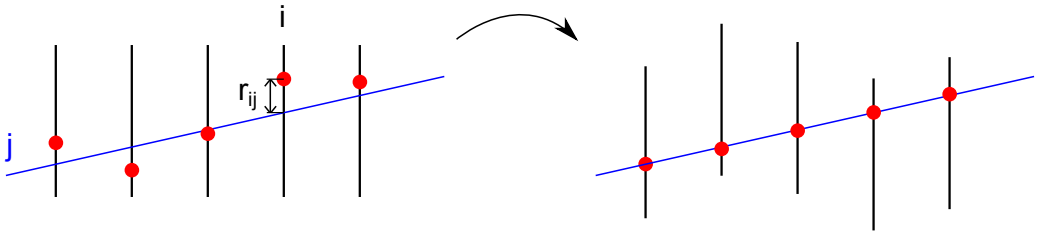


Abbildung 4.1.: Veranschaulichung des Alignment-Prinzips für einen Freiheitsgrad, nämlich einer Translation in einer Dimension. Die Residuen r_{ij} werden minimiert, was einer Minimierung des χ^2-Wertes aus Gleichung 4.1 entspricht.

Die Lösung des χ^2-Problems ist aufgrund der großen Anzahl an Parametern eine Herausforderung für Alignment-Algorithmen. Bei CMS werden zwei Strategien verfolgt.

Im HIP-Algorithmus (**H**it and **I**mpact **P**oint Algorithm) [15] wird ein lokal iteratives Verfahren verwendet. Es wird der χ^2-Wert für jeden Sensor separat minimiert, wobei die Teilchenspurparameter bei jeder Iteration konstant gehalten werden. Korrelationen der Spurparameter zu anderen Modulen werden somit zunächst vernachlässigt. Es ist trotzdem mittels mehrfacher Iteration möglich, Konvergenz zu erreichen, indem die Spuren in jedem Zyklus neu angepasst werden.

Das Alignment mit dem Millepede-Algorithmus [16] hingegen basiert auf einer globalen und simultanen Minimierung des χ^2-Wertes 4.1 bezüglich aller Teilchenspur- und Alignmentparameter. Hierbei werden alle Korrelationen berücksichtigt. Um einen globalen Fit möglich zu machen, werden einige Methoden angewendet, die das Problem vereinfachen. Ein Überblick über diese Vorgehensweise wird in Kapitel 5.1 gegeben. In konzeptioneller Hinsicht ist der Millepede-Algorithmus dem HIP-Algorithmus überlegen. Andererseits ist der HIP-Algorithmus schneller und robuster gegenüber unpassenden Teilchenspurmodellen, wie z.B. dem Modell einer Helixbahn. Bei CMS werden beide Algorithmen angewendet, um eine unabhängige Validierung des Alignments zu gewährleisten.

4.2. Schwache Moden und Auswahl der Teilchenspuren

Um die passendste Geometrie für den Detektor zu ermitteln, minimieren spurbasierte Alignment-Algorithmen den χ^2-Wert aus Gleichung 4.1. Weicht die Geometrie vom Optimum ab, so ermittelt der Algorithmus in der Regel einen schlechteren χ^2-Wert und korrigiert in Richtung der idealen Geometrie. Transformationen der Alignmentparameter, die eine solche Abweichung hervorrufen, nennt man starke Moden (strong modes). Schwache Moden (weak modes) hingegen sind Transformationen der Parameter, die den χ^2-Wert nicht beeinflussen. Es sind also verschiedene Sätze von Alignmentparametern möglich, die denselben minimalen χ^2-Wert aufweisen. Der Algorithmus ist gegenüber diesen schwachen Moden blind. Ein Beispiel einer solchen Transformation ist in Abbildung 4.2 demonstriert.

Das χ^2-Minimierungsproblem lässt sich auch als lineares Gleichungssystem formulieren und schwache Moden treten auf, wenn dieses Gleichungssystem unterbestimmt ist. Das Problem der schwachen Moden lässt sich lösen, indem weitere Bedingungen, zum Beispiel in Form von verschiedenartigen Teilchenspuren, gestellt werden. So werden in Abbildung 4.2 Spuren von kosmischer Strahlung eingeführt, da ihr χ^2 sensitiv gegenüber einer Teleskop-Transformation (Verschiebung der Detektorteile in z-Richtung um einen Betrag $\Delta z \propto r$) ist. Bei CMS werden aus diesem Grund vier Sorten von Teilchenspuren aus Kollisionsereignissen sowie kosmischer Strahlung für das Alignment verwendet [5].

- **Minimum-Bias-Ereignisse**: Mit Minimum-Bias werden Ereignisse bezeichnet, bei denen mit sehr lockeren Triggerbedingungen inelastische Proton-Proton-Kollisionen selektiert werden. Dabei ist im allgemeinen nicht exakt festgelegt, welche Triggerbedingungen das sind. Minimum-Bias-Daten sollen einen möglichst unverzerrten Querschnitt aller vorkommenden inelastischen Proton-Proton-Kollisionen wiedergeben. Es sind überwiegend weiche, aber auch harte Prozesse enthalten. Durch die vielfältigen Transversalimpulse der Teilchen sind die Spuren unterschiedlich gekrümmt und korrelieren verschiedenste Module miteinander.
- **Kosmische Strahlung**: Die Spuren der kosmischen Strahlung laufen, im Gegensatz zu Spuren aus Kollisionsereignissen, nicht durch den Kollisionspunkt. Sie verknüpfen daher ganz andere Detektorteile miteinander, besonders die obere mit der unteren Detektorhälfte (vgl. Abbildung 4.2). Es werden auch Spuren kosmischer Strahlung verwendet, die bei abgeschaltetem Magnetfeld aufgenommen werden und somit ungekrümmt sind.
- **Ereignisse mit isolierten Myonen**: Die Teilchenspuren isolierter Myonen werden verwendet, da Myonen im Detektormaterial wenig gestreut werden. Außerdem sind sie bei hoher Luminosität ausreichend vorhanden.
- **Ereignisse mit Myonen aus dem Zerfall eines Z-Bosons**: Diese Ereignisse erlauben eine zusätzliche Massen-Randbedingung des rekonstruierten Z-Boson-Kandidaten.

Darüber hinaus können noch weitere Zwangsbedingungen gestellt werden, zum Beispiel Informationen aus äußeren Messungen oder Primärvertex-Einschränkungen. Meistens ist es nicht leicht, eine schwache Mode zu identifizieren und aufzulösen. Eine Strategie ist dabei, auf künstliche Weise die Alignmentparameter zu transformieren und zu testen, ob der Algorithmus diese Verzerrung behebt [9].

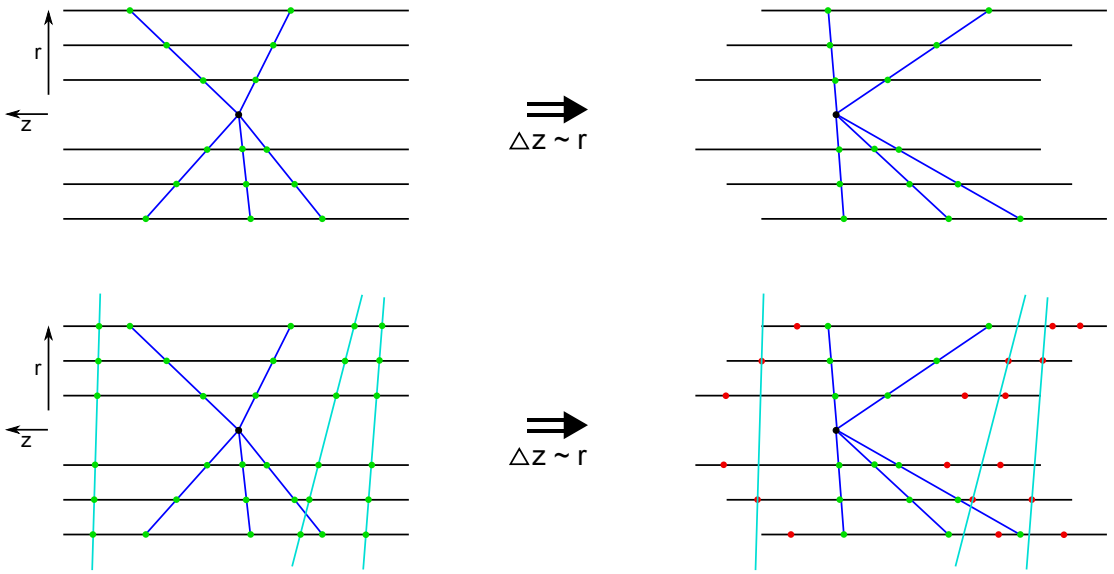


Abbildung 4.2.: Beispiel einer schwachen Mode: Auswirkung einer künstlich hervorgerufenen Teleskop-Transformation (Verschiebung der Detektorteile in z-Richtung um einen Betrag $\Delta z \propto r$) bei unterschiedlichen Teilchenspur-Sorten. Oben nur Teilchenspuren aus Kollisionen. Unten mit Spuren von kosmischer Strahlung als zusätzliche Bedingung. Je besser die Punkte durch die Spuren beschrieben werden, desto geringer ist der resultierende χ^2-Wert. Im unteren Fall wird daher die linke Geometrie bevorzugt, während im oberen Fall nach wie vor derselbe χ^2-Wert ermittelt wird. Die künstliche Transformation wird also nur im unteren Fall korrigiert.

5. Millepede Alignment-Algorithmus

Millepede ist ein χ^2-Minimierungsalgorithmus, der auf der Unterscheidung der gesuchten Parameter in lokale und globale Parameter beruht. Im Fall des spurbasierten Alignments entsprechen die globalen Parameter den Alignmentparametern, also jenen, die die Anordnung der Sensoren angeben. Die Bezeichnung „global“ rührt daher, dass diese Parameter für alle Teilchenspuren und Ereignisse dieselben sind. Die lokalen Parameter entsprechen den Parametern für das Modell der Teilchenspuren und hängen mit nur einem einzigen Ereignis zusammen. Mittels einiger mathematischer Handgriffe kann das Minimierungsproblem stark reduziert werden, sodass es in brauchbaren Zeitspannen lösbar ist.

In der zweiten Version Millepede II [16] wird das Vorbereiten der Spuren von der eigentlichen Minimierung separiert. Die beiden Schritte werden „Mille“ und „Pede“ genannt. Der Mille-Schritt produziert aus den gemessenen Teilchenspuren binäre Dateien, die die vorbereiteten Matrixelemente des Minimierungsproblems enthalten. Im Pede-Schritt wird die Lösung berechnet. Durch diese Trennung ist es möglich den Mille-Schritt zu parallelisieren, was erheblich Zeit spart.

In diesem Kapitel wird zunächst der mathematische Aspekt des Millepede-Algorithmus skizziert und anschließend die softwaretechnische Herausforderung diskutiert.

5.1. Mathematische Vorgehensweise

Ausgehend von Gleichung 4.1 ist es sinnvoll, eine Linearisierung des Teilchenspurmodells in der Nähe der Sensormodule durchzuführen. Dies entspricht einer Taylorentwicklung um die Startwerte $\boldsymbol{p_0}$ und $\boldsymbol{\tau_{j0}}$ der globalen und lokalen Parameter. Gleichung 4.1 wird so zu

$$\chi^2 = \sum_j \sum_i r_{ij}^2(\boldsymbol{\tau_j}, \boldsymbol{p}) \tag{5.1}$$

$$\approx \sum_j \sum_i \frac{1}{\sigma_{ij}^2} \left(m_i - \left[y_i(\boldsymbol{\tau_{j0}}, \mathbf{p_0}) + \frac{\partial y_i}{\partial \boldsymbol{p}} \boldsymbol{\Delta p} + \frac{\partial y_i}{\partial \boldsymbol{\tau_j}} \boldsymbol{\Delta \tau_j} \right] \right)^2 . \tag{5.2}$$

Dieser Term wird mit der Bedingung minimiert, dass die Ableitungen bezüglich der gesuchten Parameter verschwinden, was zu den sogenannten Normalgleichungen führt [17]. Mit den Vektoren

$$\boldsymbol{c_i}^T = \left(\frac{\partial y_i}{\partial p_1}, ..., \frac{\partial y_i}{\partial p_n}, \frac{\partial y_i}{\partial \tau_1}, ... \frac{\partial y_i}{\partial \tau_m} \right) \tag{5.3}$$

$$\boldsymbol{a}^T = (\Delta p_1, ... \Delta p_n, \Delta \tau_1, ..., \Delta \tau_m) \tag{5.4}$$

lassen sich die Normalgleichungen als Matrixgleichung formulieren.

$$\underbrace{\sum_{ij} \frac{1}{\sigma_{ij}^2} \left(\boldsymbol{c_i} \boldsymbol{c_i}^T\right)}_{\boldsymbol{C}} \cdot \boldsymbol{a} = \underbrace{\sum_{ij} \frac{1}{\sigma_{ij}^2} \left(m_i - y_i(\boldsymbol{\tau_{j0}}, \boldsymbol{p})\right) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}}_{\boldsymbol{b}} \tag{5.5}$$

$$\mathbf{C}\boldsymbol{a} = \boldsymbol{b} \tag{5.6}$$

$\mathbf{C}$ ist eine $(n+m) \times (n+m)$-Matrix, wobei n die Anzahl der globalen und m die der lokalen Parameter bezeichnet. Theoretisch könnte man die Lösung mittels Matrixinversion finden, doch praktisch ist dies mit einer Rechenzeit proportional zu $(n+m)^3$ verbunden.

Der entscheidende Punkt des Millepede-Algorithmus ist eine Reduktion der Größe der Matrix [16, 18, 10]. Die Matrix weist eine Struktur auf, die der einer Blockdiagonalmatrix ähnelt. Dadurch ist es möglich die Matrix für die Lösung nach den globalen Parametern auf die Größe $n \times n$ zu bringen. Trotz dieser Reduktion wäre die Rechenzeit von n^3 mittels Matrixinversion zu hoch, da n bei großen Detektoren wie dem CMS-Spurdetektor im Bereich von 10^5 liegt. Deshalb wird bei Millepede II üblicherweise auf eine numerische Lösung mit dem MINRES-Algorithmus [19] zurückgegriffen. Die erste Version von Millepede beruht noch auf Matrixinversion und funktioniert bis zu einer Größenordnung von 1000 globalen Parametern.

5.2. Aufspaltung in Mille- und Pede-Schritt

Aufgrund des Ansatzes der globalen Minimierung bezüglich aller Parameter ist es nicht möglich, den Gesamtprozess zu parallelisieren, um Zeit zu sparen. Seit Millepede II werden jedoch die zwei separaten Schritte „Mille“ und „Pede“ unterschieden, wobei beim Mille-Schritt eine starke Parallelisierung möglich ist.

Mille konvertiert die Daten, die mit dem konkreten Detektor (in diesem Fall der CMS-Spurdetektor) zusammenhängen, in binäre Dateien (Mille-Binaries), welche vom Experiment abstrahiert sind. Sie enthalten die Messungen und ihre Unsicherheiten, die Ableitungen des Teilchenspurmodells bezüglich der globalen und lokalen Parameter, sowie verallgemeinerte Kennzeichnungszahlen (Labels), die bei CMS den jeweiligen Teil des Detektors angeben. Der Mille-Schritt lässt sich in mehrere Rechenjobs aufspalten und auf verschiedene Rechner des CERN-Batchsystems [20] verteilen. Diese Rechenjobs werden im folgenden auch als Millejobs bezeichnet.

Pede ist ein eigenständiges Programm, das die binären Dateien liest und die Minimierung durchführt. Die Ergebnisse werden zunächst in einfachen Textdateien ausgegeben. Da für den globalen Fit die Matrix aus Abschnitt 5.1 komplett verfügbar sein muss, sind nur bestimmte Maschinen des Batchsystems, welche über genügend Arbeitsspeicher verfügen, in der Lage, den Pede-Schritt durchzuführen. Der Pede-Schritt lässt sich zwar nicht auf mehrere Rechenjobs aufteilen, aber durch Multithreading lässt sich die Rechendauer verringern. Rechenjobs, die Pede ausführen, werden im folgenden als Pedejobs bezeichnet.

Abgestimmt auf CMS ist Millepede II in das CMS-Software-Framework (CMSSW) [7, 21, 22] integriert. In CMSSW werden Rechenjobs für das Batchsystem (im folgenden manchmal auch nur als Jobs bezeichnet) mittels Konfigurationsdateien gesteuert, welche in der Programmiersprache Python [23] geschrieben sind. Der Aufbau der Mille- und Pedejobs innerhalb des Frameworks wird in Kapitel 5.3 näher beschrieben.

In Abbildung 5.1 ist der Ablauf einer Alignmentkampagne mit Millepede II in CMSSW dargestellt. Pro Millejob werden eine oder mehrere AlCaRECO-Dateien („Alignment and Calibration Reconstructed files“) verarbeitet. AlCaRECO-Dateien sind eine Sorte von RECO-Dateien [7], die speziell für Kalibrierung und Alignment konzipiert sind. In RECO-Dateien

sind für jedes Ereignis rekonstruierte Objekte wie Teilchenspuren, Vertex-Kandidaten und Teilchen-Kandidaten enthalten. Als Teil der Millejobs werden mit verschiedenen Auswahlkriterien unerwünschte Teilchenspuren herausgefiltert. Von der Alignment-Producer-Klasse werden die Spuren zum General-Broken-Lines-Modell [24] umparametrisiert, da dieses Spurmodell besonders gut für Millepede II basiertes Alignment geeignet ist. Anschließend wird Mille aufgerufen, welches die binären Dateien für Pede vorbereitet. Pede liest diese Dateien und führt den globalen Fit durch.

Um mit einer großen Anzahl von Millejobs umzugehen, wurde das „Millepede Production System“ (MPS) [25] als Teil von CMSSW entwickelt. Es bildet eine Datenbank mit Informationen zu allen Rechenjobs, bietet einfache Befehle zum Abschicken der Jobs und prüft die Ausgabe auf Fehlermeldungen. Die erste Version des MPS wurde 2007 fertiggestellt und in der Skriptsprache Perl [26] geschrieben. Über die Jahre haben sich die Anforderungen an das MPS jedoch gewandelt und eine Überarbeitung ist nötig. Mit dieser Arbeit wird dazu ein Beitrag geleistet. Auf die Probleme und Neuerungen im MPS wird in Kapitel 6 eingegangen.

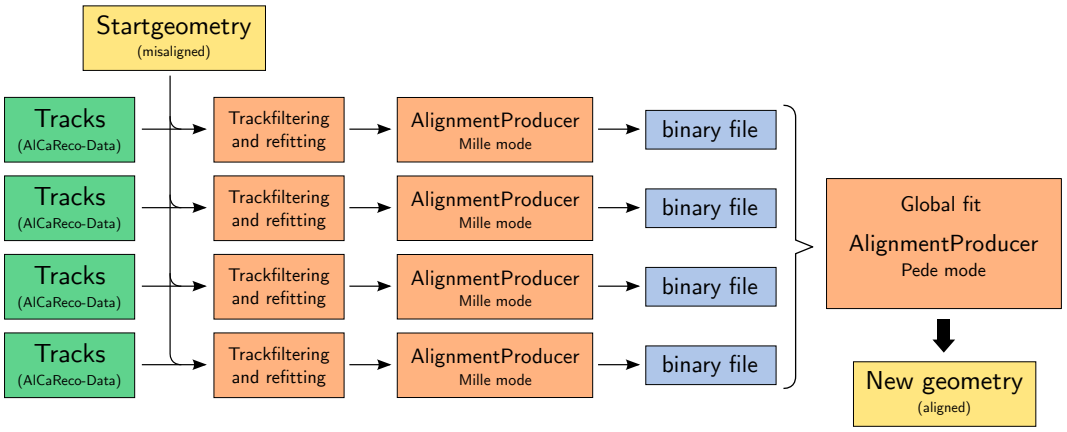


Abbildung 5.1.: Ablauf einer Alignmentkampagne mit Millepede II in CMSSW (basierend auf [27]).

5.3. Aufbau der Jobs in CMSSW

In CMSSW werden Rechenjobs in Python-Konfigurationsdateien zusammengestellt und mit der cmsRun-Anwendung ausgeführt [28]. Eine solche Datei ist mittels einer Objekthierarchie strukturiert. Die Objekte bezeichnen Komponenten von cmsRun und können modular zusammengestellt werden. Zentral ist dabei das „process“-Objekt, als dessen Attribute die verwendeten Module spezifiziert werden. Im „source“-Attribut wird zum Beispiel der Pfad zur Eingabedatei angegeben. Weitere mögliche Module sind beispielsweise Teilchenspurfitter oder auch die Alignment-Producer-Klasse, welche die Schnittstelle zu Millepede darstellt. Das „path“-Attribut gibt an, in welcher Reihenfolge die Module ausgeführt werden. In Abbildung 5.2 ist der Ablauf eines Millejobs anhand eines solchen Pfades veranschaulicht.

Eine Konfigurationsdatei beginnt damit, dass das „process“-Objekt initialisiert wird. Des Weiteren muss eine Startgeometrie des Detektors vorgegeben werden, indem ein sogenannter „Global Tag“ angegeben wird, der auf einen bestimmten Zustand des gesamten Detektors verweist. Die Geometrieparameter werden bei CMSSW in sqlite-Datenbanken

gespeichert, und für genauere Anpassungen können Teile des Global Tags mit Informationen aus anderen sqlite-Dateien überschrieben werden.

Für die Konfiguration des Alignment-Producers muss angegeben werden, welche Teile des Detektors beim globalen Fit berücksichtigt werden sollen. Oft ist es unsinnig ein Alignment für den kompletten Detektor durchzuführen, wenn zu wenige Teilchenspuren dafür vorhanden sind. Zum Beispiel werden manchmal nur die Alignmentparameter der größeren Strukturen berücksichtigt. Es ist auch möglich, einzelne Freiheitsgrade wie die Deformationsparameter wegzulassen. Speziell für den Pede-Schritt müssen im Alignment-Producer auch Einstellungen vorgenommen werden. In diesen wird beispielsweise die Anzahl der Threads für den ausführenden Prozessor und die Methode der Minimierung (Matrixinversion, Diagonalisierung, MINRES oder andere numerische Verfahren) festgelegt.

Danach werden die verschiedenen CMSSW-Module spezifiziert und deren Parameter auf die gewünschten Werte gesetzt. In einem gewöhnlichen Millejob sind vor der Übergabe an den Alignment-Producer einige Module zum Vorbereiten der Teilchenspuren vorgeschaltet. Eine Übersicht dieser Prozedur ist in Abbildung 5.2 gegeben. Die Module filtern unerwünschte Teilchenspuren heraus, die die Qualitätsansprüche für das Alignment nicht erfüllen. Aus technischen Gründen sind dafür erneute Anpassungen der Teilchenspuren nötig. Das Vorbereiten der Spuren nimmt in den Job-Konfigurationsdateien den meisten Platz ein, obwohl der Code überwiegend statisch ist und ausgelagert werden kann. Anschießend übernimmt der Alignment-Producer, welcher die Teilchenspuren zum General-Broken-Lines-Modell [24] umparametrisiert und Mille beziehungsweise Pede aufruft.

Obwohl der Aufbau jedes Millejobs sehr ähnlich ist, unterscheidet sich die Selektion der Teilchenspuren und somit die Einstellung der CMSSW-Module je nach Ereignistyp. Es werden die Ereignistypen aus Kapitel 4.2 unterschieden (Minimum-Bias-Ereignisse, kosmische Strahlung, $Z^0 \to \mu\mu$ Ereignisse oder Ereignisse mit isolierten Myonen). In der bisherigen Version des MPS werden daher verschiedene Vorlagen für die Konfigurationsdateien benötigt. Davon abgesehen sind diese Vorlagen überwiegend statisch und unterscheiden sich in der Regel nicht zwischen aktuellen Alignmentkampagnen. Die zu alignierenden Parameter, Pede-Einstellungen und die Startgeometrie unterscheiden sich jedoch von Kampagne zu Kampagne, sind aber für alle Ereignistypen innerhalb einer Kampagne gleich.

In der CMSSW-Konfiguration für den Pede-Schritt werden die Module zur Behandlung der Teilchenspuren nicht benötigt. Ansonsten muss im Vergleich zu einer Mille-Konfiguration der Modus des Alignment-Producers von „mille“ zu „pede“ gewechselt und eine Liste der Dateinamen der Mille-Ausgabe angegeben werden. Dabei ist es möglich den binären Dateien einzelner Millejobs statistische Gewichte zuzuordnen, die als Faktoren für die Likelihood der betroffenen Terme im globalen Fit des Pede-Schrittes verwendet werden. Dies ist nötig, um die Dominanz eines Datensatzes zu skalieren, falls beispielsweise besonders viele oder wenige Teilchenspuren in einem Datensatz vorhanden sind.

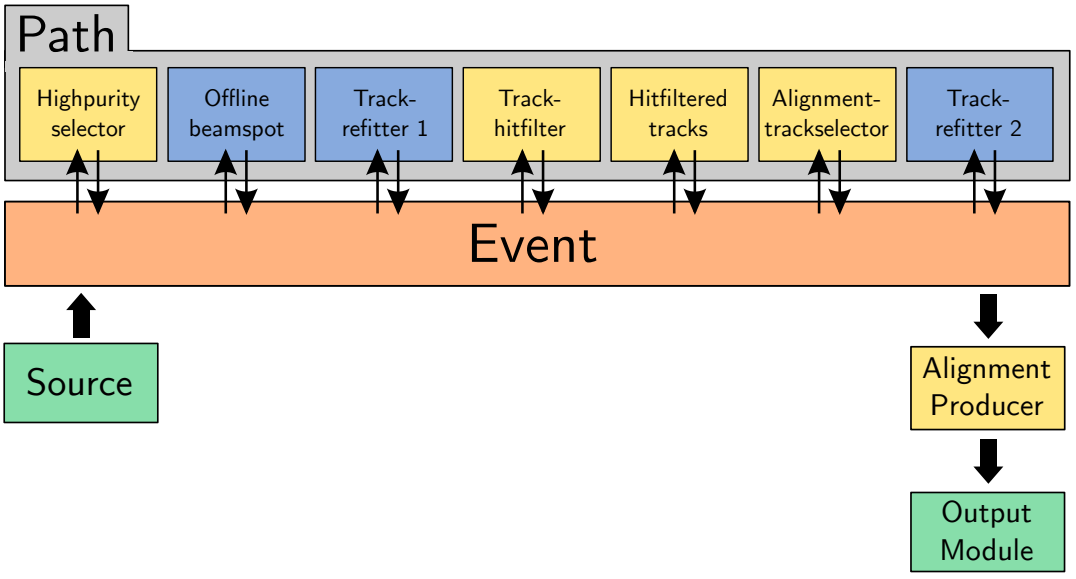


Abbildung 5.2.: Struktur einer CMSSW-Konfigurationsdatei für einen Millejob. Jedes Ereignis wird von einer Reihe von Modulen behandelt, bevor der Alignment-Producer übernimmt. Letzterer wird nicht im Pfad angegeben. Die Module sind einerseits verschiedene Filtermodule (orange) und andererseits Rekonstruktionsmodule (blau).

6. Das Millepede Production System (MPS)

Das Millepede Production System (MPS) [25] ist ein für die Bedingungen beim CMS-Experiment programmiertes Konstrukt zur effizienten Behandlung von Millepede-Alignmentkampagnen mit vielen Rechenjobs. Es hilft die CMSSW-Konfigurationsdateien für die Mille- und Pedejobs zu erstellen und bietet eine zeichenorientierte Benutzerschnittstelle um Jobs an das Batchsystem des CERN [20] zu submittieren und ihren Status zu überwachen. Außerdem prüft es eine Vielzahl von Ausgabeprotokollen auf herkömmliche Fehlermeldungen.

In diesem Kapitel werden die Funktionsweise des MPS und die Neuerungen besprochen, die den Kern dieser Arbeit bilden.

6.1. Aufbau und Funktionsweise des MPS

Das MPS ist Teil des CMSSW und besteht aus mehreren Skripten, welche eine Benutzerschnittstelle zur Behandlung der Jobs bilden. Diese Skripte wurden ursprünglich in der Programmiersprache Perl [26] geschrieben. Heutzutage ist diese Sprache aber nicht mehr weit genug verbreitet, und in der CMS-Kollaboration wird vorrangig auf Python [23] als Skriptsprache gesetzt. Python ist auch allgemein sehr beliebt wegen seiner guten Lesbarkeit und eingängigen Syntax. Im Rahmen dieser Arbeit wurden daher einige wichtige Skripte des MPS in Python übersetzt. In Zukunft ist es so einfacher, Änderungen am MPS vorzunehmen.

Eine Übersicht der MPS-Skripte ist in Abbildung 6.1 gegeben, und Erläuterungen zu den wichtigsten Skripten sind in den Tabellen 6.1 und 6.2 zu finden. Dabei sind vor allem die Skripte aufgeführt, die sehr häufig vom Nutzer angewendet werden oder solche, die relevant für das Aufsetzen von Alignmentkampagnen sind.

Da das MPS speziell für die rechnerstrukturellen Rahmenbedingungen des CMS-Experiments ausgelegt ist, werden zum Verständnis in Kapitel 6.1.1 die wichtigsten Verzeichnisse und Dateien erläutert.

6.1.1. Relevante Dateien und Verzeichnisse

- **Gemeinsame MP-Umgebung**: Dies ist ein Verzeichnis auf dem AFS-Speicher [29] des CERN, das von allen Mitwirkenden des Millepede-Alignments bei CMS genutzt wird. Für jede Alignmentkampagne wird ein Unterverzeichnis (namens mp****, wobei * jeweils eine Ziffer ist) angelegt, das als Arbeitsverzeichnis dient.
- **EOS-Speicher**[30]: Aus Platzgründen können nicht alle für ein Alignment nötigen Dateien auf dem AFS-Speicher verweilen. Daher werden besonders große Dateien, wie die binären Milledateien, auf den EOS-Speicher des CERN ausgelagert.
- **mps.db-Datei**: Diese Datei befindet sich im Arbeitsverzeichnis der betreffenden Alignmentkampagne und bildet den Knotenpunkt für die MPS-Steuerungsskripte. Sie ist eine Datenbank, in der die Jobs mit allen nötigen Informationen aufgelistet sind. Zu jedem Job findet man hier den derzeitigen Status, die Referenznummer des Batchsystems, Bezeichnungen, zu welchem Datensatz der Job gehört, den Pfad, unter dem die CMSSW-Konfigurationsdatei und das Batch-Skript für diesen Job abgelegt sind, und einige weitere Angaben.
- **jobData-Verzeichnis**: Dieses Verzeichnis befindet sich im Arbeitsverzeichnis einer Kampagne. Es beinhaltet für jeden Rechenjob ein Unterverzeichnis, in dem die CMSSW-Konfigurationsdatei, das Batch-Skript und die theSplit-Datei (s.u.) des jeweiligen Jobs liegen.
- **CMSSW-Konfigurationsdatei**: (Dateiname: „the.py“ für Millejobs und „alignment_merge.py für Pedejobs) Der Aufbau dieser Datei ist in Kapitel 5.3 beschrieben. Sie bestimmt den Ablauf eines Rechenjobs und kann auch lokal mit „cmsRun“ ausgeführt werden.
- **Batch-Skript**: (Dateiname: „theScript.sh“) Ein Zsh-Skript [31], das die Anweisungen für das Batchsystem enthält. Darin werden beispielsweise Eingabe- und Ausgabedateien an die richtigen Stellen kopiert und cmsRun mit der jeweiligen CMSSW-Konfigurationsdatei aufgerufen.
- **theSplit**: Eine Textdatei, die bei Millejobs die Pfade zu den verwendeten Eingabedateien angibt.
- **CMSSW-Konfigurationsvorlage**: Für jeden der Mille- sowie Pedejobs wird vom MPS eine CMSSW-Konfigurationsdatei erstellt, deren Anweisungen von der cmsRun-Anwendung ausgeführt werden. Der Aufbau einer solchen Datei ist in Kapitel 5.3 beschrieben. Für das Erstellen der fertigen CMSSW-Konfigurationsdatei benötigt das MPS eine Vorlage, die an mehreren Stellen auf den jeweiligen Job angepasst werden muss.
- **Quelltextfragmente**: Für das Aufsetzen von Alignmentkampagnen mit dem `setup_align.pl`-Skript (siehe Kapitel 6.2.1) werden verschiedene Quelltextfragmente benötigt, die in die CMSSW-Konfigurationsvorlage kopiert werden. Eines davon gibt die Startgeometrie für das Alignment an (Datei: `startgeometry.txt`), indem Geometrieinformationen des Global Tags (siehe Kapitel 5.3) überschrieben werden. Ein weiteres Quelltextfragment enthält Einstellungen zum Pede-Schritt (Datei: `pedesettings.txt`) und ein drittes enthält Informationen darüber, welche der Alignmentparameter berücksichtigt werden sollen (Datei: `alignables.txt`). Es existiert noch ein letztes Quelltextfragment, in welchem defekte Module angegeben werden (Datei: `deadmodules.txt`). Dieses ist im neuen Arbeitsablauf (siehe Kapitel 6.2.2) abgeschafft.

6.1.2. MPS-Steuerungsskripte

Das MPS besteht momentan aus 23 Skripten (siehe Abbildung 6.1), welche vom Nutzer in der Konsole aufgerufen werden können. Davon kann man sieben als Subskripte bezeichnen, da sie üblicherweise nicht direkt ausgeführt werden, sondern innerhalb anderer Skripte aufgerufen werden. Die Funktionen der wichtigsten Skripte sind in den Tabellen 6.1 und 6.2 zusammengefasst (siehe auch Tabelle 6.3 für eine Erklärung der verschiedenen Rechenjobzustände). Alle Skripte, die mit der `mps.db`-Datenbank interagieren, greifen auf die zu diesem Zweck entworfene Datenbankklasse `Mpslibclass.py` zu. Zuletzt findet man im MPS zwei Batch-Skript-Vorlagen für Mille- beziehungsweise Pedejobs, aus denen die Skripte `mps_script.pl` und `mps_scriptm.pl` die finalen Batch-Skripte erstellen.

Im ersten Teil dieser Arbeit lag der Fokus auf dem Erstellen von Python-Versionen einiger Skripte, einerseits um ein tiefes Verständnis für das MPS zu erlangen, andererseits um die Skripte lesbarer und einfacher anpassbar zu machen. So sind nun die Skripte `mps_fire.py`, `mps_stat.py`, `mps_fetch.py`, `mps_splice.py`, `mps_merge.py`, `mps_update.py`, `mps_check.py` und die Datenbankklasse als Python-Version vorhanden. Das Skript `mps_list_evts.py` wurde bereits ursprünglich in Python verfasst.

Die meisten Skripte sollen zunächst die exakt gleiche Funktion wie ihre Perl-Gegenstücke erfüllen. Zusätzlich wurden bei der Übersetzung einige kleinere Fehler behoben. Die beiden Skripte `mps_splice.py` und `mps_merge.py` sind jedoch grundlegend umstrukturiert, da die entsprechenden Perl-Skripte mit aktuellen CMSSW-Konfigurationsvorlagen nicht wie vorgesehen arbeiten. Der Grund ist, dass die CMSSW-Konfigurationsvorlagen sich im Laufe der Jahre den Ansprüchen entsprechend wandeln und es zu Inkompatibilitäten kommt. Die neuen Python-Versionen sind auf die Struktur einer neuen, universellen CMSSW-Konfigurationsvorlage (siehe Abschnitt 6.2.2) ausgelegt und nicht mit den bisherigen CMSSW-Konfigurationsvorlagen kompatibel. Besonderes Augenmerk liegt auf der Robustheit der neuen Skripte gegenüber minimaler Änderungen an der CMSSW-Konfigurationsvorlage, da das alte Skript `mps_splice.pl` schon wegen kleinsten Abweichungen, wie einem Leerzeichen an einer fragilen Stelle, versagen kann.

Das neu hinzugefügte Skript `mps_alisetup.py` erfüllt eine Funktion, die zuvor nicht im MPS vorgesehen war. Es ermöglicht ein vereinfachtes Aufsetzen von Alignmentkampagnen mit mehreren Datensätzen und stellt eine große Verbesserung des Arbeitsablaufes des MPS dar. Ausführlich wird dies in Kapitel 6.2 besprochen.

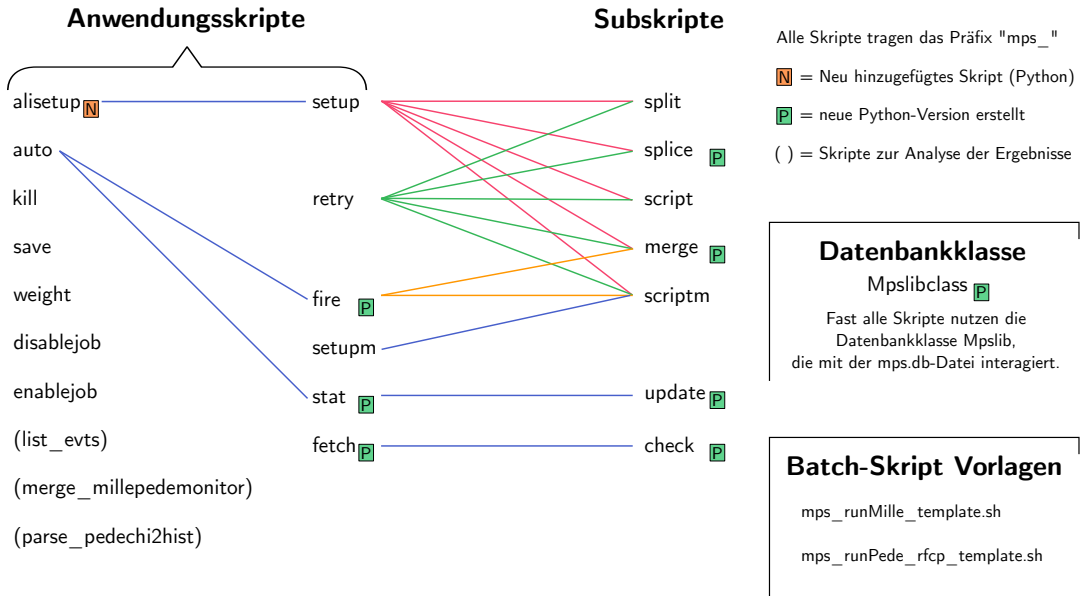


Abbildung 6.1.: Die MPS-Skripte und ihre Relationen. Links die Anwendungsskripte, die der Nutzer im Terminal ausführt. Rechts die Subskripte, die innerhalb der Anwendungsskripte aufgerufen werden. Die Klasse Mpslibclass.py wird von allen Skripten verwendet, die mit der `mps.db`-Datei interagieren.

Tabelle 6.1.: Die MPS-Subskripte (s. Tabelle 6.3 für Jobzustände).

Skript	Funktion
`mps_check.py`	Prüft die verschiedenen Ausgabeprotokolle aller Jobs, die sich im „FETCH“-Zustand befinden, auf übliche Fehlermeldungen. Setzt den Jobzustand anschließend auf „OK“, „FAIL“, „WARN“, „ABEND“ oder „TIMEL“. Außerdem extrahiert es die Rechendauer aus einem der Protokolle.
`mps_update.py`	Überprüft den Status der Jobs, die an das Batchsystem übergeben wurden, und aktualisiert dementsprechend die `mps.db`-Datenbank.
`mps_split.pl`	Wird primär von `mps_setup.pl` aufgerufen. Es verteilt eine Liste von Eingabe-Dateipfaden auf die Anzahl der gewünschten Jobs und schreibt die jeweiligen Pfade in die Datei „theSplit“, die im betreffenden Jobverzeichnis abgelegt wird.
`mps_splice.py`	Wird primär von `mps_setup.pl` aufgerufen. Liest die Eingabedatei-Pfade aus „theSplit“ und schreibt sie in die CMSSW-Konfiguration des Jobs als Quelldateien.
`mps_script.pl`	Wird primär von `mps_setup.pl` aufgerufen. Finalisiert das Batch-Skript der Millejobs auf Basis einer Vorlage, die als Teil des MPS im CMSSW-Release enthalten ist.
`mps_scriptm.pl`	Wird primär von `mps_setup.pl` aufgerufen. Finalisiert das Batch-Skript der Pedejobs auf Basis einer Vorlage, die als Teil des MPS im CMSSW-Release enthalten ist.
`mps_merge.py`	Wird primär von `mps_setup.pl` aufgerufen. Es nimmt die CMSSW-Konfiguration des letzten Millejobs und baut daraus die CMSSW-Konfiguration des Pedejobs.

Tabelle 6.2.: Die wichtigsten MPS-Anwendungsskripte (s. Tabelle 6.3 für Jobzustände).

Skript	Funktion	Kommentar
`mps_setup.pl`	Wird beim Aufsetzen neuer Alignmentkampagnen verwendet. Es baut aus schon vorbereiteten Vorlagen die finalen CMSSW-Konfigurationsdateien und Batch-Skripte und speichert sie im jobData-Verzeichnis ab, wobei für jeden Job ein Unterverzeichnis angelegt wird. Außerdem erstellt es die `mps.db`-Datenbank.	Da `mps_setup` sehr viele Argumente nimmt und immer nur einen Datensatz, z.B. eine Liste von Minimum-Bias-AlCaReco-Dateien, zu den Jobs hinzufügt, wird in der Praxis ein Helferskript namens `setup_align.pl` verwendet. In der neuen Version des MPS ist dieses Helferskript durch das neue MPS-Skript `mps_alisetup.py` abgelöst, welches Teil des CMSSW-Release ist (s. Kapitel 6.2).
`mps_alisetup.py`	Ruft `mps_setup.pl` für mehrere Datensätze auf. In der Regel werden mindestens vier Datensätze verwendet - einen für jede Ereignissorte aus Kapitel 4.2.	Dieses Skript ist neu und löst das Helferskript `setup_align.pl` ab.
`mps_fire.py`	Submittiert Mille- sowie Pedejobs an das Batchsystem. Dabei werden nur Jobs im „SETUP“-Zustand berücksichtigt. Als Argument nimmt das Skript die Anzahl der zu startenden Jobs.	Mit der Option „-m“ wird der nächste Pedejob losgeschickt und mit der Zusatzoption „-f“ wird eine neue CMSSW-Konfiguration für den Pedejob erstellt, bei dem nur Millejobs mit dem Zustand „OK“ berücksichtigt werden.
`mps_stat.py`	Liest die `mps.db`-Datei und gibt dessen wichtigste Informationen im Terminal aus. Außerdem ruft es das Subskript `mps_update.py` auf, welches den Zustand der zum Batchsystem übergebenen Jobs aktualisiert.	Die Ausgabe ist hauptsächlich eine Liste der Jobs mit ihrem jeweiligen Zustand, ihrer Jobnummer, die Batchsystem-Referenznummer, dem Namen des Datensatzes, Informationen zur Rechenzeit, die Anzahl der Events und das statistische Gewicht für den Pedejob.
`mps_fetch.py`	Prüft welche Jobs erledigt sind (Zustand: „DONE“) und verschiebt deren Ausgabeprotokolle in das passende Unterverzeichnis im jobData-Verzeichnis. Anschließend ruft es für diese Jobs das Subskript `mps_check.py` auf.	Setzt den Zustand der erledigten Jobs auf „FETCH“.
`mps_setupm.pl`	Mit diesem Skript wird ein zusätzlicher Pedejob auf Basis eines schon vorhandenen Pedejobs erstellt.	Oft ist es nützlich mehrere Pedejobs mit den gleichen binären Milledateien zu starten um verschiedene Einstellungen für den globalen Fit oder andere statistische Gewichte für die Millejobs zu verwenden.
`mps_weight.pl`	Setzt oder ändert statistische Gewichte für Millejobs und notiert sie in der `mps.db`-Datenbank. Diese werden von `mps_merge.py` und implizit bei `mps_setupm.pl` für den Bau neuer Pede-CMSSW-Konfigurationen verwendet.	Statistische Gewichte können auch manuell in der CMSSW-Konfiguration eingetragen werden. Mit dem neuen `mps_alisetup.py`-Skript lassen sich direkt beim Aufsetzen der Alignmentkampagne oder auch nachträglich mit der Option „-w“ Gewichte festlegen, die durch `mps_weight.pl` zu den CMSSW-Konfigurationen propagiert werden.

6.1.3. Ablauf einer Alignmentkampagne im MPS

In diesem Abschnitt soll beschrieben werden, wie eine Alignmentkampagne abläuft. Der Begriff „Alignmentkampagne“ soll dabei nur den Teil eines Alignments bezeichnen, der im MPS erfolgt. Alle anderen Arbeitsschritte, wie z.B. das Zusammenstellen der Datensätze, werden hier vernachlässigt.

Zum Aufsetzen einer Alignmentkampagne wird das neue Skript `mps_alisetup.py` verwendet. Bis Januar 2016 wurde dafür das Helferskript `setup_align.pl` eingesetzt. Beide Methoden sowie die Vorzüge des neuen Skriptes werden in Kapitel 6.2 diskutiert.

Nach erfolgreichem Aufsetzten werden zunächst die Millejobs mit dem `mps_fire.py`-Skript an das Batchsystem übermittelt und der Nutzer erlangt erst wieder die Kontrolle über die Jobs, wenn sie vom Batchsystem beendet wurden. Die großen Dateien der Mille-Ausgabe (binäre Dateien und Kontrollhistogramm-Dateien) werden automatisch auf dem EOS-Speicher (siehe 6.1.1) platziert. Protokolle zum Ablauf der Jobs werden hingegen im Arbeitsverzeichnis abgelegt. Der Nutzer kann dann das `mps_fetch.py`-Skript verwenden um die Protokolle zu überprüfen. Sollte ein Job die Tests nicht bestehen, so ist es über das `mps_retry.pl`-Skript möglich, den Job erneut auszuführen. Die Millejobs, die die Tests bestehen, bekommen den Zustand „OK“ zugewiesen und sind bereit für den Pede-Schritt. Dabei ist die Abwicklung des Pedejobs im MPS analog zu der der Millejobs. In Abbildung 6.2 ist der Ablauf anhand der Rechenjobzustände (siehe auch Tabelle 6.3) veranschaulicht.

Die Ergebnisse des Alignments sind dann in verschiedenen Formaten im `jobData`-Verzeichnis des Pedejobs zu finden, einerseits als einfache Liste aller ermittelten Alignmentparameteränderungen in der Datei `millepede.res`, andererseits als neue Geometrie für CMSSW in einer oder mehreren sqlite-Dateien. Des Weiteren sind in der Ausgabe auch einige Protokolle und Kontrolldiagramme über die Güte des globalen Fits enthalten.

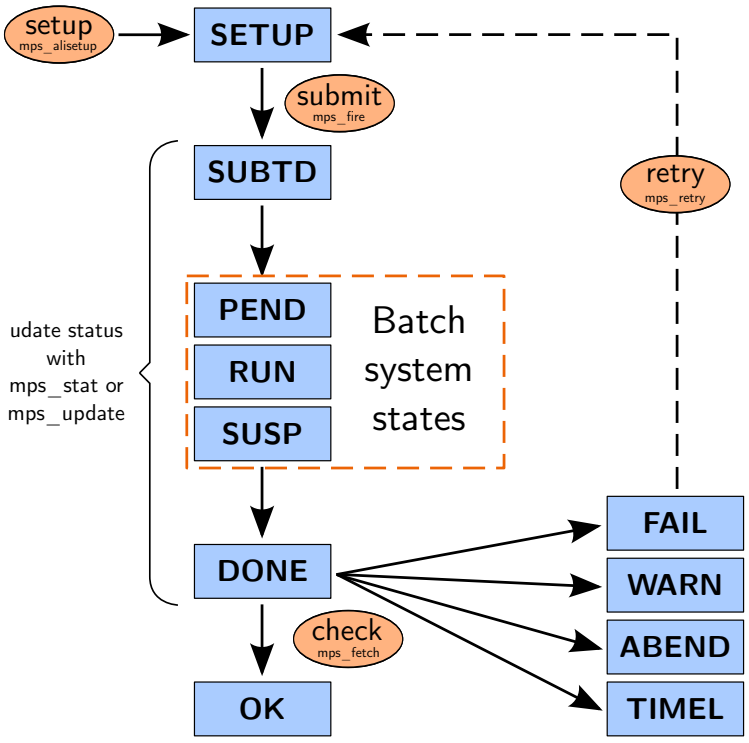


Abbildung 6.2.: Die Behandlung der Rechenjobs innerhalb des MPS – veranschaulicht anhand der Rechenjobzustände (basierend auf [32]). Eine Erläuterung der Jobzustände ist in Tabelle 6.3 zu finden.

Tabelle 6.3.: Liste aller Rechenjobzustände des MPS [25].

Status	Beschreibung
`SETUP`	Job wurde erstellt und ist bereit submittiert zu werden.
`SUBTD`	Job wurde submittiert, aber noch nicht vom Batchsystem registriert.
`PEND`	Job ist vom Batchsystem registriert und wartet auf Ausführung.
`RUN`	Job wird vom Batchsystem ausgeführt.
`SUSP`	Job wurde vom Batchsystem unterbrochen.
`DONE`	Job ist beendet und nicht mehr unter der Kontrolle des Batchsystems.
`OK`	Job hat alle Tests des MPS erfolgreich bestanden.
`FAIL`	Job hat mindestens einen kritischen Test nicht bestanden.
`WARN`	Job hat mindestens einen unkritischen Test nicht bestanden.
`ABEND`	Job kam zu keinem normalen Ende.
`TIMEL`	Zeitlimit wurde beim Ausführen des Jobs überschritten.

6.2. Aufsetzen von Alignmentkampagnen

Kern dieser Arbeit ist die Umstrukturierung des Arbeitsablaufes bezüglich des Aufsetzens von Alignmentkampagnen mit dem MPS. Das MPS selbst bietet ursprünglich keine Funktion ein Alignment mit mehreren Datensätzen auf einfache Weise aufzusetzen. Einen Ausweg stellt das Helferskript `setup_align.pl` dar, welches für mehrere Datensätze die CMSSW-Konfigurationsvorlagen so weit wie nötig vorbereitet und an `mps_setup.pl` übergibt. In der gegenwärtigen Form ist es aber nicht möglich das Skript als Teil des MPS in das CMSSW aufzunehmen, da man das Skript zur Spezifikation der Datensätze direkt editieren muss. Außerdem ist der Arbeitsablauf mit diesem Skript nicht sehr geschickt und sorgt aufgrund vieler involvierter Dateien für Verwirrung. Ziel ist es daher ein besseres Skript (`mps_alisetup.py`) in Python zu konstruieren um `setup_align.pl` abzulösen. Ein Kerngedanke ist dabei, die Spezifikation der Datensätze aus dem Skript auszulagern, was die Integration in das CMSSW ermöglicht. Außerdem soll das neue Skript für mehr Übersicht und Klarheit sorgen, indem weniger Dateien involviert werden. Zuletzt soll das Skript möglichst einfach zu handhaben und flexibel für weitere Verbesserungen sein.

Zur Darstellung der Unterschiede werden in diesem Kapitel die Arbeitsabläufe mit dem alten `setup_align.pl`-Skript sowie mit dem neuen `mps_alisetup.py`-Skript beschrieben.

6.2.1. Bisheriger Ablauf: Aufsetzen von Alignmentkampagnen mit setup_align.pl

Die Alignmentkampagne beginnt üblicherweise mit dem Kopieren aller relevanten Dateien von einem vorherigen Alignment. Dies beinhaltet das `setup_align.pl`-Skript, die Quelltextfragmente für Startgeometrie, Pede-Einstellungen, Alignmentparameter und defekte Module und die CMSSW-Konfigurationsvorlagen für die verschiedenen Datensatz-Sorten (kosmische Strahlung, Minimum-Bias-Daten, isolierte Myonen und $Z \to \mu\mu$ Daten). Insgesamt also neun Dateien.

Das Quelltextfragment zu den defekten Modulen ist veraltet und mittlerweile nicht mehr relevant. Die restlichen Quelltextfragmente müssen vom Nutzer auf die Alignmentkampagne angepasst werden. Der Zweck dieser Dateien ist in Kapitel 6.1.1 beschrieben.

Zur Spezifikation der Datensätze wird `setup_align.pl` direkt editiert. Für jeden Datensatz werden mehrere Angaben beziehungsweise Variablen benötigt. Diese sind in Tabelle 6.4 zusammengefasst.

Anschließend kann `setup_align.pl` ausgeführt werden. Für jeden Datensatz kopiert es die Quelltextfragmente in die CMSSW-Konfigurationsvorlage und fügt den Global Tag ein. Je nach Bedarf werden noch weitere Zeilen zum Quelltext hinzugefügt. Zuletzt wird `mps_setup.pl` mit allen verlangten Argumenten aufgerufen. Der Arbeitsablauf wird in der Abbildung 6.3 veranschaulicht.

Tabelle 6.4.: Variablen zur Spezifikation der Datensätze innerhalb des `setup_align.pl`-Helferskriptes.

Name	Bedeutung	optional
py	Der Pfad zu der dem Datensatz entsprechenden CMSSW-Konfigurationsvorlage.	nein
data	Der Pfad zur Liste der Eingabedateien.	nein
njobs	Die Anzahl der Millejobs, die für diesen Datensatz erstellt werden soll.	nein
globaltag	Ein Global Tag, also eine CMSSW-Kennzeichnung für den entsprechenden Detektorzustand.	nein
json	Pfad zu einer json-Datei, in der gute und schlechte Datennahmeperiode aufgelistet sind und somit unerwünschte Ereignisse ausgelassen werden.	ja
apvmode	Unterscheidung zwischen „peak“- und „deconvolution“-Modus [33, 34] für die Auslesetechnik des Streifendetektors bei kosmischer Strahlung.	ja

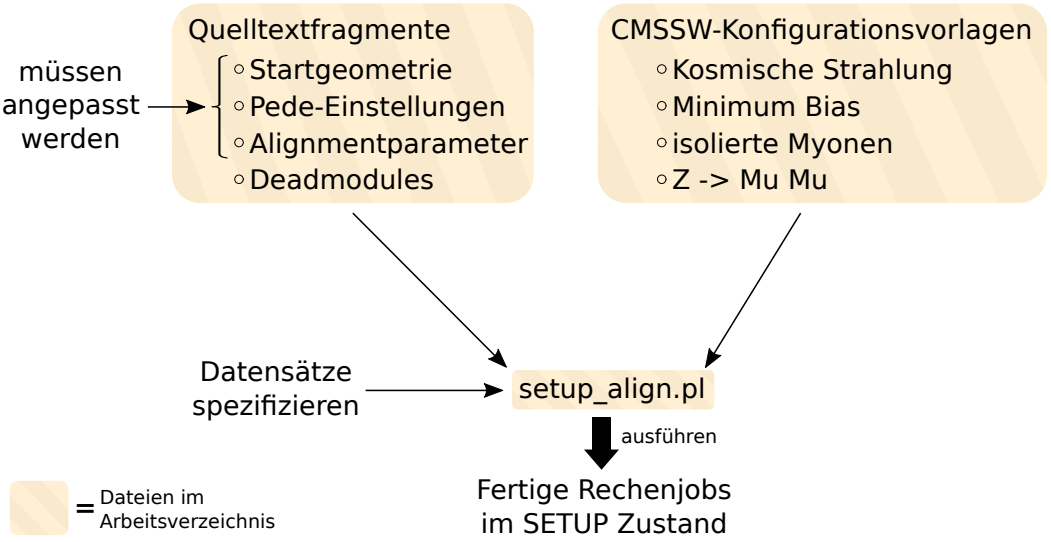


Abbildung 6.3.: Schematischer Ablauf des Aufsetzens einer Alignmentkampagne mit dem `setup_align.pl`-Skript.

6.2.2. Neuer Ablauf: Aufsetzen von Alignmentkampagnen mit mps_alisetup.py

Der neue Arbeitsablauf basiert auf der Abspaltung der Datensatz-Spezifizierung in eine Alignment-Konfigurationsdatei vom „ini“-Dateityp und der Vereinigung aller CMSSW-Konfigurationsvorlagen zu einer universellen Konfigurationsvorlage für alle Datensatz-Sorten. Der Arbeitsablauf besteht nun daraus, die Quelltextfragmente direkt in die universelle CMSSW-Konfigurationsvorlage einzutragen und die Datensätze in der ini-Konfigurationsdatei zu spezifizieren. Wenn beides erledigt ist, wird das neue MPS-Skript `mps_alisetup.py` verwendet, an das als Argument der Name der ini-Datei übergeben werden muss. Somit sind im Arbeitsverzeichnis nur noch zwei anstatt neun Dateien nötig, was für wesentlich mehr Überblick sorgt. Der grobe Arbeitsablauf ist in Abbildung 6.4 illustriert.

In den folgenden Abschnitten werden der Aufbau der ini-Konfigurationsdatei und die Besonderheiten der universellen CMSSW-Konfigurationsvorlage beschrieben.

6.2.2.1. Alignment-Konfigurationsdatei (.ini)

ini-Dateien bieten eine einfache Syntax um Variablenwerte zu initialisieren und in Python gibt es eine Klasse, die diese Syntax verarbeiten kann. Eine ini-Datei ist in Abschnitte gegliedert. Unter jedem Abschnitt können mehreren Variablen Werte zugewiesen werden. Für jeden Datensatz, den der Nutzer spezifizieren möchte, wird ein Abschnitt mit dem Schlüsselwort „dataset“ im Titel erstellt. Zusätzlich gibt es einen weiteren Abschnitt mit dem Namen „general“. Hier werden einige allgemeine Variablen gesetzt, z.B. wie viel Arbeitsspeicher vom Batchsystem für den Pedejob verlangt wird und an welche Warteschlange des Batchsystems die Rechenjobs überreicht werden sollen. Außerdem kann man hier für einige Variablen, die auch in den Datensatz-Abschnitten auftauchen, Standardwerte setzen. Auf diese wird zurückgegriffen, wenn die entsprechende Variable bei einem Datensatz nicht angegeben wird. Alle Variablen zur Spezifikation der Datensätze sind in Tabelle 6.5 beschrieben. Dabei wird zwischen essentiellen und optionalen Variablen unterschieden. In Anhang B ist ein Beispiel einer fertigen Alignment-Konfigurationsdatei zu finden.

Durch die neue Möglichkeit Standardwerte im „general“-Abschnitt zu setzen, ist es nun möglich Datensätze sehr kompakt anzugeben. In besonders einfachen Fällen reichen sogar die Variablen „collection“ und „inputFileList“ vollkommen aus. Die neue Variable „weight“ stellt eine weitere große Verbesserung dar. Zuvor mussten statistische Gewichte der Millejobs von Hand in die CMSSW-Konfiguration des Pedejobs eingetragen werden. Bei etwa 100 Millejobs ist das sehr mühselig und fehleranfällig, weshalb diese Vereinfachung eine dringend angeforderte neue Funktion ist.

Die überarbeitete Methodik der Angabe von Datensätzen erleichtert den Arbeitsablauf beträchtlich. Zudem bietet das `mps_alisetup.py`-Skript ausgeklügelte Fehlerbehandlungen, falls Variablen fehlen sollten, und eine aussagekräftigere Ausgabe in der Kommandozeile. Auf Anfrage der Alignmentgruppe wurde zum Skript `mps_alisetup.py` eine Option hinzugefügt, die das Aufsetzen weiterer Pedejobs mit anderen statistischen Gewichten ermöglicht.

6.2.2.2. Universelle CMSSW-Konfigurationsvorlage

Zur Reduktion der Anzahl involvierter Dateien gibt es nun eine universelle CMSSW-Konfigurationsvorlage, welche die vier Vorlagen für die unterschiedlichen Datensatz-Sorten ablöst. Zuvor waren Abschnitte, die sich in allen Vorlagen decken sollen, aber für jede Alignmentkampagne editiert werden müssen, als Quelltextfragmente für Startgeometrie, Alignmentparameter, Pede-Einstellungen und defekte Module abgespalten. Diese Fragmente

wurden dann von `setup_align.pl` in die CMSSW-Konfigurationsvorlagen kopiert. Mit der universellen Konfigurationsvorlage ist es möglich diese Quelltextfragmente direkt in die Vorlage einzutragen, was für mehr Überblick sorgt. Außerdem ist nach Absprache mit der Alignmentgruppe das Quelltextfragment über defekte Module abgeschafft. So ersetzt die neue Konfigurationsvorlage insgesamt acht Dateien.

Die universelle CMSSW-Konfigurationsvorlage basiert auf dem Trackselection-Refitting-Tool im Common-Alignment-Paket von CMSSW [35]. Dieses Hilfsprogramm liefert die Prozedur der Teilchenspurvorbereitung, wie sie in Kapitel 5.3 in der Grafik' 5.1 veranschaulicht ist. Das Programm nimmt als Argument die „collection"-Variable aus der ini-Konfigurationsdatei (siehe Abschnitt 6.2.2.1) und erstellt anhand dessen die passenden Anweisungen für den Rechenjob. So übernimmt dieses Hilfsprogramm einen großen Teil der Konfigurationsvorlage. Nun muss lediglich in einem weiteren Programmteil, dem Alignment-Producer, zwischen den verschiedenen Datensatz-Sorten unterschieden werden, was mit einfachen if-Abfragen realisiert wird.

Um noch mehr Überblick zu schaffen, sind aus der universellen Konfigurationsvorlage einige statische Abschnitte komplett in das CMSSW ausgelagert. Dies betrifft beispielsweise die Einstellung des Alignment-Producers, aber auch andere kleinere Programmabschnitte. Für eine erleichterte Angabe der Startgeometrie gibt es nun eine Funktion, die die erforderte Syntax drastisch minimiert.

Abschließend lässt sich sagen, dass der neue Arbeitsablauf einfacher, übersichtlicher und schneller ist als der vorherige. Außerdem entfällt mit der Option, statistische Gewichte für ganze Datensätze anzugeben, ein besonders umständlicher Teil des alten Arbeitsablaufes. Zur Einübung des neuen Arbeitsablaufes und zur Erklärung aller Neuerungen wurde am 4. Februar 2016 innerhalb der Millepede-Alignmentgruppe ein Tutorium veranstaltet. Dabei wurde Raum für Rückmeldung geboten, und es wurden weitere Ideen vorgebracht, die dann noch implementiert wurden.

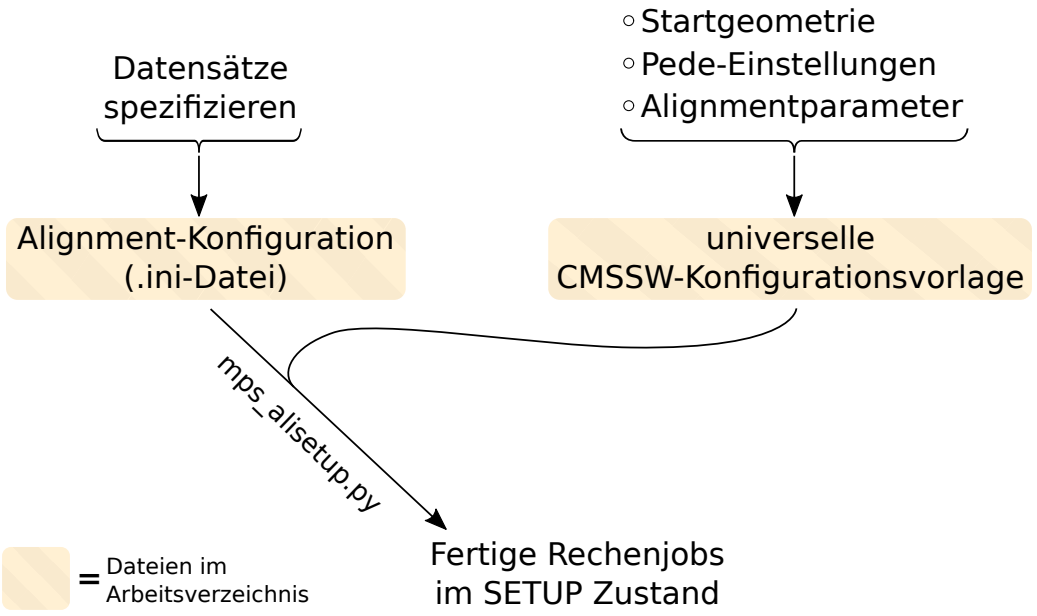


Abbildung 6.4.: Schematischer Ablauf des Aufsetzens einer Alignmentkampagne mit dem `mps_alisetup.py`-Skript.

Tabelle 6.5.: Variablen zur Spezifikation der Datensätze innerhalb der ini-Konfigurationsdatei.

Name	Bedeutung	optional
collection	Eine Zeichenkette, die die Datensatz-Sorte (siehe Kapitel 4.2) angibt. Diese Variable ist nötig, da es nun nur noch eine CMSSW-Konfigurationsvorlage gibt, in der anhand dieser Variable unterschieden wird, was geschieht.	nein
inputFileList	Der Pfad zur Liste der Eingabedateien. Entspricht der Variable „data“ aus Tabelle 6.4 und trägt nun einen deskriptiveren Namen.	nein
globaltag	Wie im alten Arbeitsablauf muss auch hier ein Global Tag angegeben werden. Doch es ist möglich einen Standardwert im „general“-Abschnitt für alle Datensatz-Abschnitte zu setzen, was meistens sinnvoll ist.	nein
configTemplate	Entspricht der Variable „py“ aus Tabelle 6.4. Da es in der Regel nur noch die eine universelle CMSSW-Konfigurationsvorlage gibt, wird der Name der Vorlage vorrangig im „general“-Abschnitt als Standard gesetzt. Für mehr Flexibilität ist es dennoch möglich diesen Standard in einzelnen Datensatz-Abschnitten zu überschreiben.	nein
cosmicsZeroTesla	Sollte der betreffende Datensatz kosmische Strahlung enthalten, die bei angeschaltetem Magnetfeld gemessen wurde, muss diese Variable auf „False“ gesetzt werden.	ja
cosmicsDecoMode	Entspricht der Variable „apvmode“ aus Tabelle 6.4. Es wird unterschieden zwischen „True“ für „deconvolution“-Modus und „False“ für „peak“-Modus [33, 34].	ja
njobs	Legt die Anzahl der Millejobs fest, die für den Datensatz erstellt werden sollen. Wenn nichts angegeben wird, verwendet `mps_alisetup.py` die Anzahl der Dateien in der Eingabe-Dateiliste und weist somit jedem Rechenjob eine Datei zu. Manchmal ist es sinnvoll weniger Rechenjobs zu produzieren, z.B. wenn manche Dateien sehr wenige Teilchenspuren enthalten.	ja
json	Wie beim alten Arbeitsablauf kann der Pfad zu einer json-Datei angegeben werden, um unerwünschte Datennahmeperioden auszuschließen.	ja
weight	Für die Millejobs des Datensatzes kann eine Gleitkommazahl als statistisches Gewicht für den Pedejob übergeben werden. Dieses Gewicht wird direkt mit der Likelihood der entsprechenden Terme im globalen Fit multipliziert.	ja

7. Ausgabe und Validierung

In Kapitel 6 werden beträchtliche Änderungen am MPS beschrieben. Wichtig ist nun, dass mit allen Veränderungen die gleichen Alignment-Ergebnisse wie zuvor erreicht werden. Dabei dürfte die Übersetzung der alten MPS-Skripte keine Auswirkungen haben, da diese größtenteils vom Inhalt der Rechenjobs abgekoppelt sind. An wenigen Stellen wird zwar von Skripten wie `mps_merge.py` oder `mps_splice.py` in den Quelltext der Rechenjobs eingegriffen, aber dies resultiert entweder in äquivalentem oder exakt gleichem Quelltext. Der neue Aufbau der universellen CMSSW-Konfigurationsvorlage muss hingegen getestet werden. Die korrekte Funktionsweise des Aufsetzens einer Alignmentkampagne mit `mps_alisetup.py` lässt sich schon direkt am Quelltext der erstellten Rechenjobs überprüfen. In der Konfigurationsvorlage bietet aber besonders die Verwendung des Trackselection-Refitting-Tools Raum für Abweichungen zu den vorherigen Konfigurationsvorlagen, da innerhalb dieses Hilfsprogramms anhand einiger Kriterien Teilchenspuren aussortiert werden. Bei nicht übereinstimmenden Kriterien kann es sein, dass mit dem neuen Arbeitsablauf geringfügig abweichende Teilchenspuren verwendet werden, was zu signifikanten Abweichungen im Endergebnis führen kann. Im Laufe der Untersuchung hat sich dies auch herausgestellt, und es mussten viele Variablenwerte im Trackselection-Refitting-Tool an Werte aus den alten CMSSW-Konfigurationsvorlagen angepasst werden.

Die Strategie der Validierung ist die Reproduktion eines Alignments, welches mit dem alten MPS und dem `setup_align.pl`-Skript durchgeführt wurde. Dafür wird das Alignment mit der Bezeichnung mp1885 herangezogen. Dieses bietet sich an, da es als Teil des Jahresend-Alignments von 2015 mit sehr vielen Daten betrieben wurde. Es wäre zwar sinnvoll mehrere verschiedene Alignmentkampagnen zu reproduzieren, aber wegen des hohen Zeitaufwandes pro Kampagne soll ein einzelnes Projekt zu Validierungszwecken ausreichen.

Ideal wäre es, bei der Reproduktion exakt die gleichen Ergebnisse wie bei mp1885 zu erhalten. Aufgrund numerischer Effekte ist dies jedoch nicht möglich. Zum einen kann das Ergebnis davon abhängen, auf welcher Maschine des Batchsystems der Pedejob ausgeführt wird, da sich die Rechner in ihren Spezifikationen unterscheiden. Es ist jedoch möglich Rechenjobs einer bestimmten Maschine zuzuweisen. Jeder der Pedejobs dieser Validierungsstudie wird daher an die Maschine mit dem Namen lxbst2131 übergeben. Auf der Kehrseite dauert das Durchführen aller nötigen Pedejobs aber noch länger, da immer nur einer bearbeitet werden kann. Ein weiteres Problem ist die Anzahl der Prozessor-Threads auf denen der Pedejob ausgeführt wird. Ein Standardwert sind zehn Threads, mit denen ein Pedejob des Ausmaßes von mp1885 bis zu einen Tag dauern kann. Es ist daher aus Zeitgründen nicht möglich nur einen Thread zu verwenden. Das Problem der Berechnung

mit mehreren Threads ist, dass es zu Abweichungen der Ergebnisse kommen kann. Eine Vermutung ist, dass dann zufällig verschiedene Rechenschritte auf einem der Kerne schneller fertig sind als auf einem anderen und sich bei der Interaktion der Threads dann Unterschiede bilden. Als Analogie kann man die Reihenfolge von Millejobs betrachten. Wenn die Mille-Ergebnisse derselben Alignmentkampagne in verschiedenen Reihenfolgen bei zwei Pedejobs verwendet werden, unterscheiden sich auch hier die Ergebnisse, wenn auch nur im sehr kleinen Maßstab. Wegen der unvermeidlichen Abweichungen ist es Teil der Validierungsstudie herauszufinden, inwiefern sich die Ergebnisse unterscheiden dürfen. Eine zu erwartende Größenordnung für akzeptable Abweichungen liegt bei etwa einem Mikrometer.

Zur Validierung bietet es sich an den Mille- und den Pede-Schritt zunächst separat zu betrachten. Die Ergebnisse des Mille-Schrittes lassen sich anhand von Kontrollverteilungen bezüglich der selektierten Teilchenspuren prüfen. Den Pede-Schritt kann man am direktesten durch einen Vergleich der Alignmentparameteränderungen prüfen.

Da alle Pedejobs auf demselben Rechner ausgeführt werden sollen, muss auch das Alignment mp1885 wiederholt werden. Es werden mehrere Kampagnen benötigt.

- **mp1944**: Exakte Kopie von der Alignmentkampagne mp1885 basierend auf dem ursprünglichen Arbeitsablauf mit dem `setup_align.pl`-Skript.
- **mp1945**: Reproduktion von mp1885 über den neuen Arbeitsablauf mit dem `mps_alisetup.py`-Skript. Hier wird der Pede-Schritt separat getestet, indem die binären Milledateien von mp1944 verwendet werden.
- **mp1977**: Reproduktion von mp1885 über den neuen Arbeitsablauf mit dem `mps_alisetup.py`-Skript. Anhand dieses Projekts wird einerseits der Mille-Schritt separat überprüft. Andererseits wird auch der gesamte Prozess aus Mille und Pede getestet.

7.1. Validierung des Mille-Schrittes

Bei jedem Millejob werden neben der binären Datei für Pede auch Kontrollhistogramme erstellt, die Informationen über die selektierten Teilchenspuren preisgeben. Enthalten sind beispielsweise Histogramme zu Impuls, Transversalimpuls und Pseudorapidität der rekonstruierten Teilchen. Zum Vergleich der Alignmentkampagnen mp1944 und mp1977 werden jeweils die Histogramme von Millejobs gleicher Datensatz-Sorten kombiniert. Gleiche Kontrollhistogramme sind streng gesehen kein Garant für identische Binärdateien, aber trotzdem ein starkes Indiz für die korrekte Selektion der Teilchenspuren.

Nach einer Angleichung einiger Selektionsparameter im Trackselection-Refitting-Tool an aktuelle Standards ergeben sich die Kontrollhistogramme von mp1977 und mp1944 als deckungsgleich. Die angeglichene Version des Programms ist nun, gemeinsam mit der aktualisierten Version des MPS, in CMSSW vorhanden. Einige ausgewählte Histogramme von mp1977 sind im Anhang A zu finden.

7.2. Validierung des Pede-Schrittes

Die Ergebnisse eines Alignments werden von Pede als Tabelle aller Alignmentparameteränderungen in der Datei namens `millepede.res` ausgegeben. Davon ausgehend errechnet CMSSW eine Spurdetektorgeometrie und speichert sie in Form von sqlite-Dateien ab. Für einen möglichst direkten Vergleich wird für diese Validierung von der `millepede.res`-Datei ausgegangen. In dieser Datei ist jeder Alignmentparameteränderung eine

Kennzeichnungszahl zugewiesen, anhand derer sich bestimmen lässt, zu welchem Subdetektor das entsprechende Modul gehört und ob es sich um einen Translations-, Rotations- oder Deformationsparameter handelt. Dabei sind Translationen in cm und Rotationen in rad angegeben. Die Deformationsparameter sind dimensionslos.

Zum Vergleich zweier Alignments werden die entsprechenden Alignmentparameteränderungen aus den beiden `millepede.res`-Dateien voneinander subtrahiert und der Betrag gebildet. Diese Werte werden dann in ein Histogramm gefüllt. Dabei muss in Translationen, Rotationen und Deformationen differenziert werden, da sich deren Werte aufgrund der verschiedenen Einheiten in unterschiedlichen Größenordnungen bewegen.

Um zu ermitteln, in welchem Ausmaß die Unterschiede zwischen den Ergebnissen normal sind, wird der Pedejob der Alignmentkampagne mp1944 zwei mal ausgeführt. Der Vergleich der Ergebnisse der beiden Pedejobs führt zu den Histogrammen in Abbildung 7.1, welche als Maßstab für erlaubte Abweichungen betrachtet werden können. Obwohl dieselben binären Milledateien und derselbe Rechner des Batch-Systems verwendet werden, ergeben sich Unterschiede in den Ergebnissen für die Alignmentparameter. Die meisten Abweichungen liegen bei den Translationen unter einem Mikrometer, aber es gibt auch Ausreißer mit bis zu 8 µm Abweichung. Im Vergleich zwischen Translationen, Rotationen und Deformationen zeichnet sich kein Unterschied aus.

Als nächstes soll der Pede-Schritt über den neuen Arbeitsablauf separat vom Mille-Schritt getestet werden. Dafür werden im Pedejob der Alignmentkampagne mp1945 die binären Milledateien von mp1944 verwendet. Der Vergleich zu dem Ergebnis des ersten Pedejobs von mp1944 liefert die Histogramme in Abbildung 7.2. Auch hier sind die Abweichungen erwartungsgemäß nicht Null, sondern liegen auf einem Niveau mit denen in Abbildung 7.1. Wie vermutet zeigt der Pede-Schritt keine Auffälligkeiten.

Zuletzt soll die komplette Alignmentkampagne mp1977 getestet werden. Dafür wird der Pedejob des Alignments mp1977 mit beiden Pedejobs von mp1944 verglichen. Die Ergebnisse sind in Abbildung 7.3 dargestellt. Auch hier zeigt sich ein ähnliches Bild wie in Abbildung 7.1.

Anhand der Histogramme lassen sich keine signifikanten Unterschiede zwischen dem alten Arbeitsablauf mit dem `setup_align.pl`-Skript und dem neuen Ablauf mit dem `mps_alisetup.py`-Skript erkennen. Die Abweichungen im niedrigen Bereich sowie die Ausreißer scheinen zu fluktuieren. Die meisten Abweichungen stammen von Modulen der Endkappen des Pixeldetektors (PXF) und den äußeren Endkappen (TEC), aber es treten auch Ausreißer von Modulen des TIB auf. Da in dieser Studie nur eine Alignmentkampagne reproduziert wird, lässt sich jedoch keine fundierte Aussage darüber treffen, weshalb besonders diese Detektorteile von Abweichungen betroffen sind.

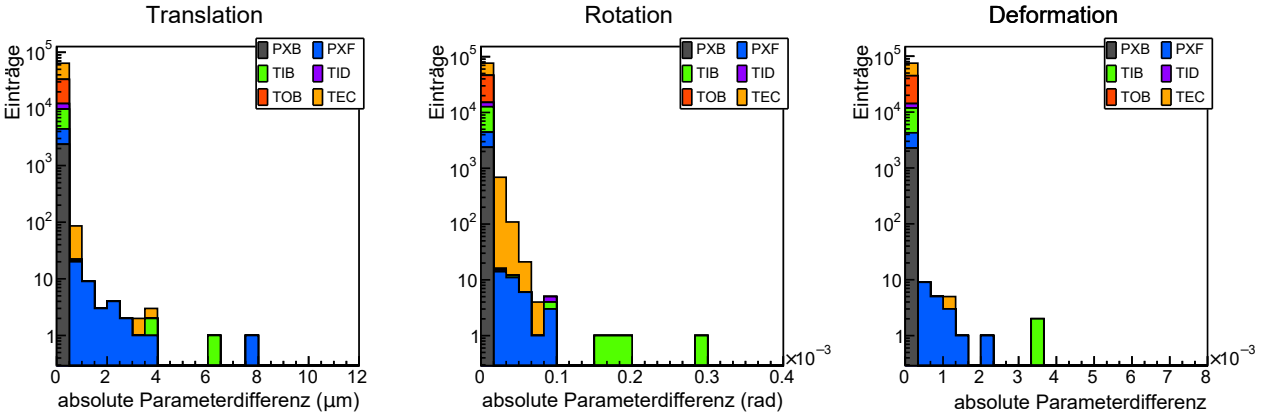


Abbildung 7.1.: **mp1944**: Vergleich der beiden identischen Pedejobs der Alignmentkampagne mp1944.

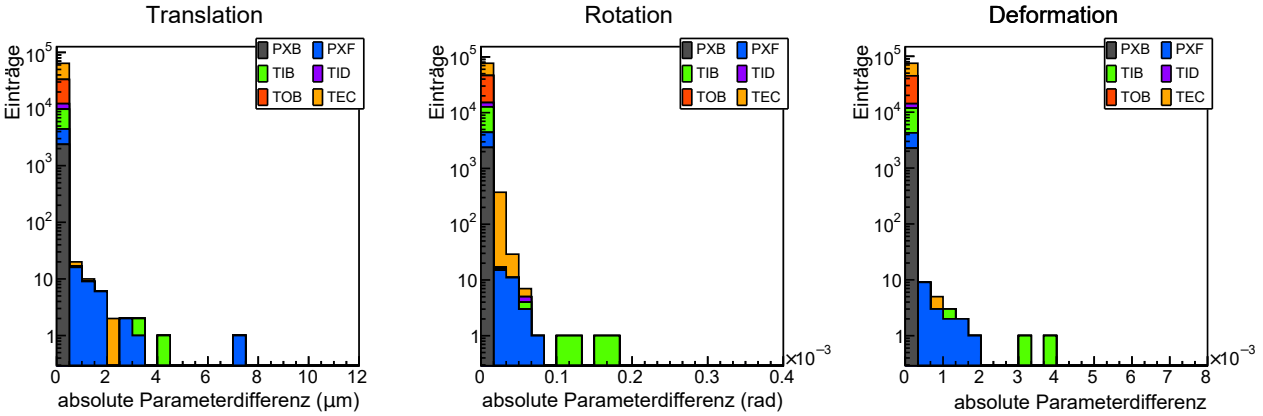
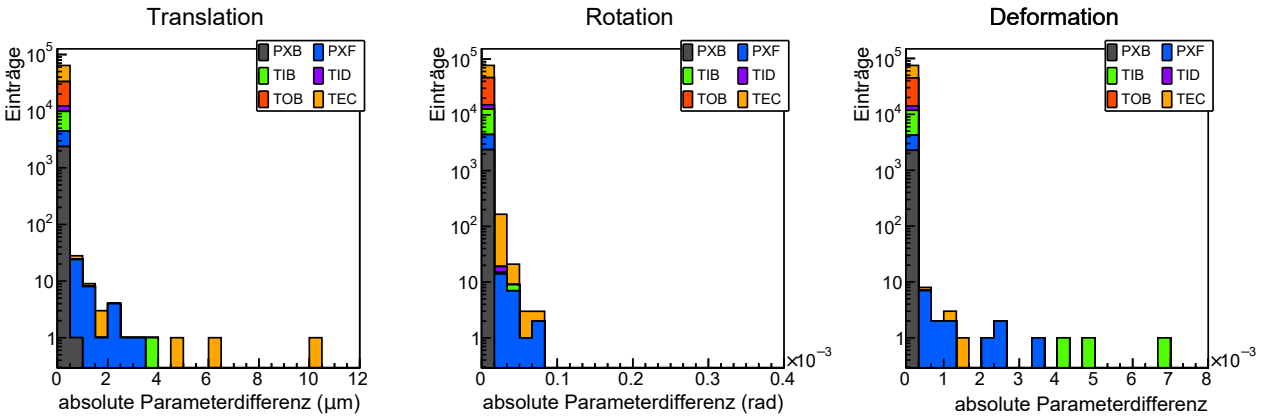
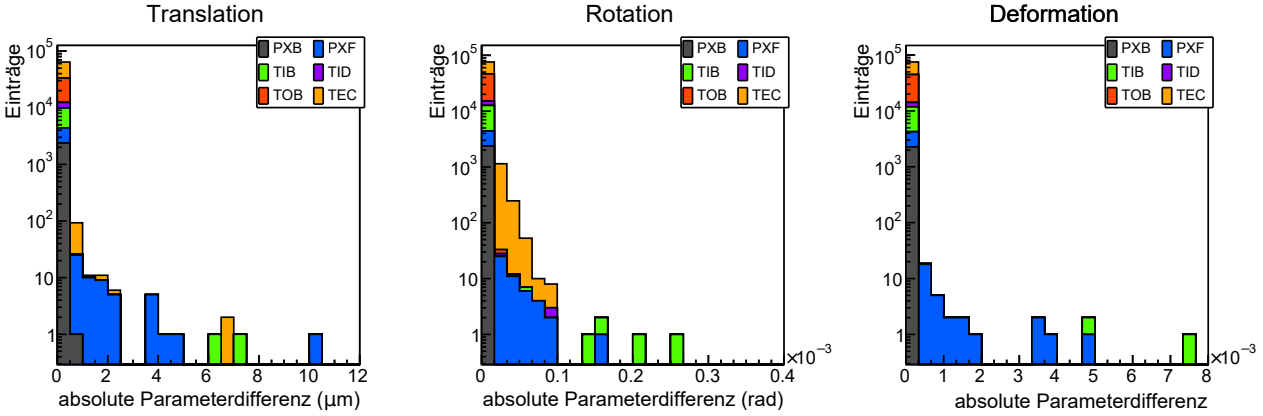


Abbildung 7.2.: **mp1944 und mp1945**: Vom Mille-Schritt unabhängiger Test des Pedejobs, welcher über den neuen Arbeitsablauf mit `mps_alisetup.py` erstellt wurde.



(a) Vergleich von mp1977 mit dem ersten Pedejob von mp1944.



(b) Vergleich von mp1977 mit dem zweiten Pedejob von mp1944.

Abbildung 7.3.: **mp1944 und mp1977**: Vollständiger Vergleich des neuen Arbeitsablaufes mit dem bisherigen Ablauf.

8. Fazit und Ausblick

Das Alignment des zentralen Spurdetektors stellt eine komplexe Aufgabe dar und ist von entscheidender Bedeutung für das CMS-Experiment. Eine genaue Bestimmung der Spurdetektorgeometrie erlaubt z.B. präzisere Rekonstruktionen von Teilchenspuren sowie Primär- und Sekundärvertizes. Im Zuge der Datennahme muss die Geometrie des Spurdetektors wegen äußerer Einflüsse wie Temperaturänderungen oft neu ermittelt werden. Millepede II ist für diese Aufgabe ein mächtiger Algorithmus, der auf elegante Weise mit einer großen Anzahl an Alignmentparametern zurechtkommt, und er wird daher routinemäßig am CMS-Experiment verwendet.

Alignmentkampagnen mit Millepede II sind fast alltägliche Arbeit, und dies soll sich auch in einer angemessenen Arbeitsweise beim Ausführen der Software widerspiegeln. Die im Rahmen dieser Arbeit überarbeitete Version des *Millepede Production Systems* (MPS) greift an der problematischsten Stelle des Arbeitsablaufes an, nämlich dem Aufsetzen der Software-Einstellungen von Alignmentkampagnen. Die Neuheiten des MPS sind stark an der Nutzerfreundlichkeit und den Interessen der Millepede-Alignmentgruppe orientiert, und es sollen durch die vorangetriebene Integration in das *CMS-Software-Framework* (CMSSW) weitere kollaborative Verbesserungen ermöglicht werden.

Mit den neuen Werkzeugen ist der Arbeitsablauf wesentlich effizienter gestaltet und der Arbeitsaufwand damit drastisch gesenkt. Dabei ist die Übersichtlichkeit, besonders durch die Reduktion der involvierten Dateien, gesteigert. Es ist jetzt einfacher zu verstehen, was im Hintergrund geschieht, und das System ist insgesamt weniger anfällig für Bedienungsfehler. Während alle Funktionen des alten Systems erhalten werden, sind nun auch neue Funktionen implementiert. So ist jetzt beispielsweise das Arbeiten mit statistischen Gewichten für den globalen Fit der Alignmentparameter, was zuvor komplett manuell konfiguriert werden musste, erheblich vereinfacht.

Zur Einübung des neuen Arbeitsablaufes wurde am 4. Februar 2016 innerhalb der Millepede-Alignmentgruppe ein Tutorium veranstaltet. Es wurden dabei weitere Ideen vorgebracht, die dann noch implementiert wurden. Die Änderungen am MPS wurden mit positiver Rückmeldung entgegengenommen und finden bereits Anwendung in aktuellen Alignmentkampagnen.

Mit der Validierungsstudie aus Kapitel 7 ist demonstriert, dass der neue Arbeitsablauf Ergebnisse liefert, die mit denen des bisherigen Ablaufes übereinstimmen.

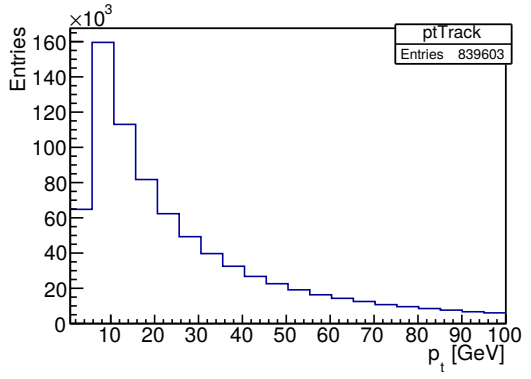
Mit der Überarbeitung des MPS im Bezug auf das Aufsetzen von Alignmentkampagnen wird das System den aktuellen Anforderungen besser gerecht. Die mit dieser Arbeit ein-

geführten Programmteile werden in Zukunft von der Millepede-Alignmentgruppe weiterentwickelt. Eine Idee zur weiteren Verbesserung ist, die Verwendung der ini-Konfigurationsdatei auszubauen. Es könnten z.B. das Setzen der Startgeometrie und der Pede-Einstellungen aus der universellen CMSSW-Konfigurationsvorlage in die ini-Konfigurationsdatei umgelagert werden. So wäre es auf elegante und effiziente Weise möglich, beim Aufsetzen einer Alignmentkampagne mehrere Pedejobs mit unterschiedlichen Einstellungen zu erstellen, während die universelle CMSSW-Konfigurationsvorlage noch statischer wird. Weiteres großes Verbesserungspotential gibt es bei der Ausgabe des Mille- und des Pede-Schrittes. Es wäre beispielsweise praktisch, wenn bei der Ausgabe des Mille-Schrittes automatisch die Anzahl der verwendeten Teilchenspuren extrahiert wird, und bei der Ausgabe des Pede-Schrittes wäre eine Überarbeitung der automatisch erstellten Kontrolldiagramme sinnvoll. Zuletzt wäre es wünschenswert den Rest der MPS-Steuerungsskripte in Python zu übersetzen, was jedoch im Vergleich zu den anderen bereits genannten Verbesserungsvorschlägen niedrige Priorität hat.

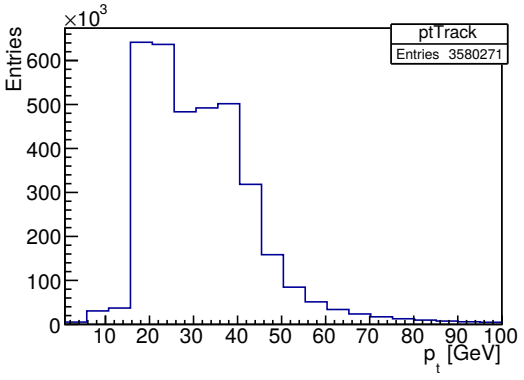
Anhang

A. Kontrollverteilungen des Mille-Schrittes der Alignmentkampagne mp1977

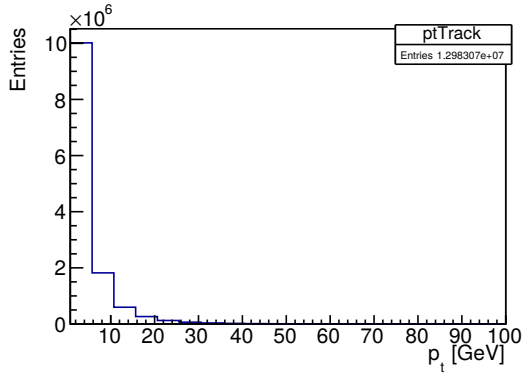
Im folgenden sind einige der Kontrollverteilungen abgebildet, auf die in Kapitel 7.1 hingewiesen wird. Da die Kontrollverteilungen von mp1977 exakt mit denen von mp1944 übereinstimmen, sind letztere hier nicht zusätzlich aufgeführt. Die Histogramme werden nicht tiefgehend ergründet, da zur Validierung lediglich die Deckungsgleichheit unter den Alignmentkampagnen relevant ist.



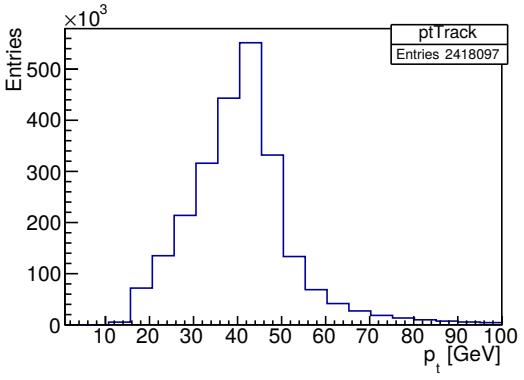
(a) Kosmische Strahlung



(b) Isoliertes Myon



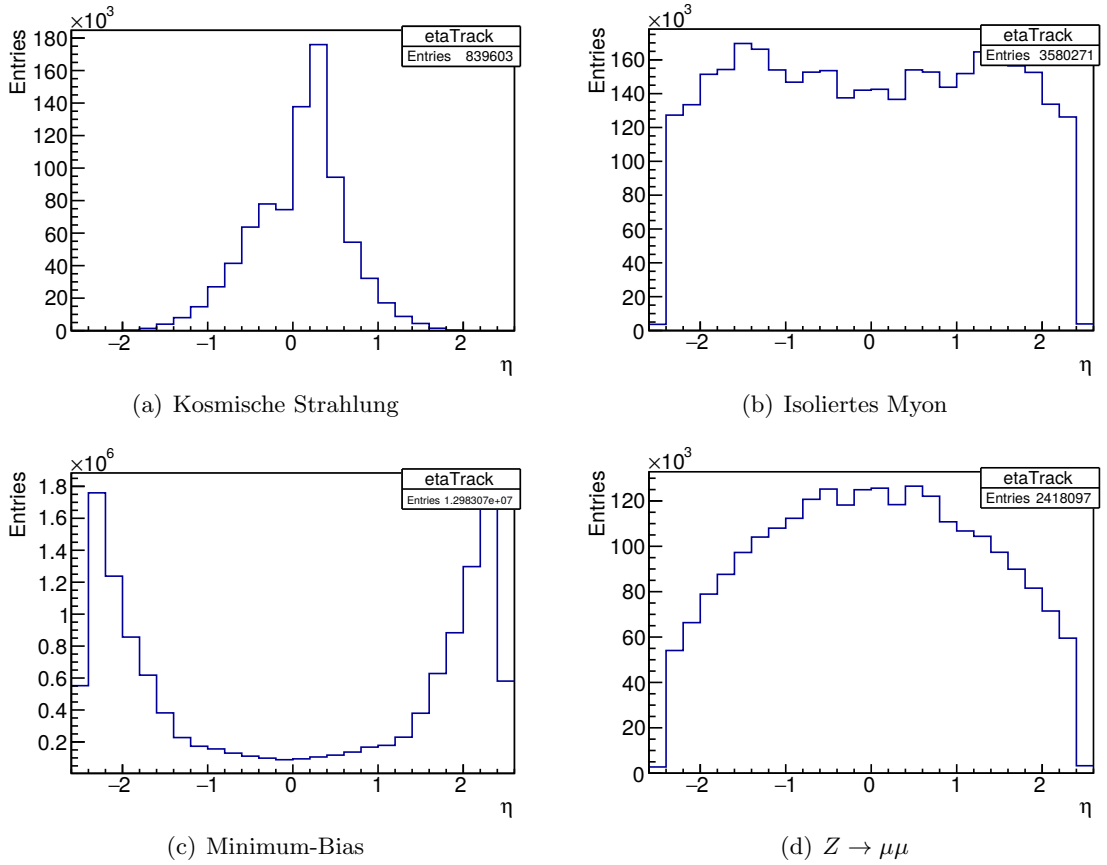
(c) Minimum-Bias



(d) $Z \to \mu\mu$

Abbildung A.1.: Transversalimpulsverteilungen (p_t) der selektierten Teilchenspuren bei der Alignmentkampagne mp1977.

Zu Abbildung A.1: Die Minimum-Bias-Datensätze liefern mit Abstand die größte Anzahl an Teilchenspuren. Im p_t-Spektrum herrscht große Vielfalt, doch die meisten Teilchenspuren liegen im niedrigen p_t-Bereich. Obwohl hier nur der Transversal- und nicht der Gesamtimpuls betrachtet wird, liegt das Maximum $Z \to \mu\mu$ Verteilung in der Nähe von 45 GeV. Dies entspricht näherungsweise der Hälfte der Z-Boson-Masse.



(a) Kosmische Strahlung

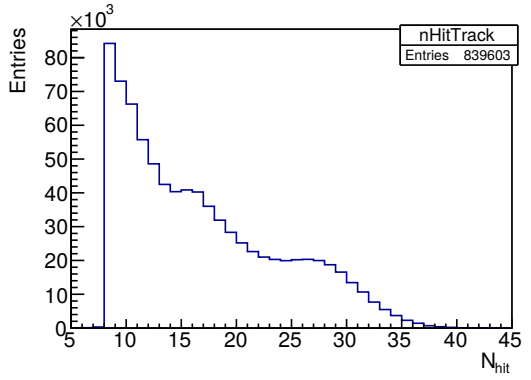
(b) Isoliertes Myon

(c) Minimum-Bias

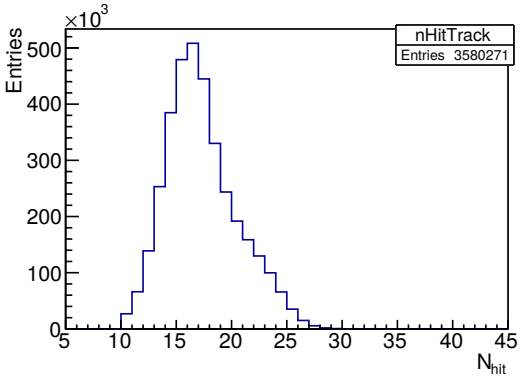
(d) $Z \to \mu\mu$

Abbildung A.2.: Pseudorapiditätsverteilungen (η) der selektierten Teilchenspuren bei der Alignmentkampagne mp1977.

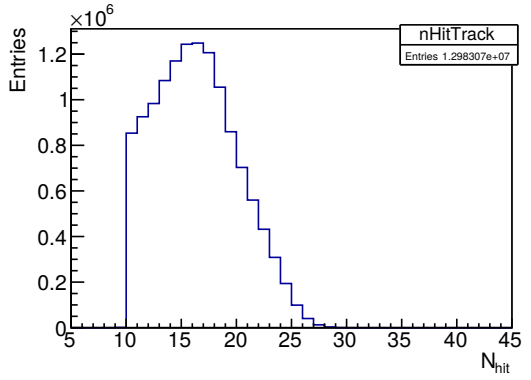
Zu Abbildung A.2: Interessanterweise ist die η-Verteilung der kosmischen Strahlung asymmetrisch. Dies rührt daher, dass durch den Schacht auf der einen Seite der CMS-Kaverne mehr kosmisches Strahlung zum Detektor vordringt. Der Betrag der Pseudorapidität ist hier insgesamt eher klein, da kosmische Teilchen von oben kommen. Der Großteil der Minimum-Bias-Teilchenspuren weist eine betragsmäßig hohe Pseudorapidität auf, da weiche Stoßprozesse mit kleinen Streuwinkeln wesentlich häufiger auftreten. Durch die hohe Anzahl der Einträge sind absolut gesehen dennoch sehr viele Teilchenspuren mit niedriger Pseudorapidität enthalten.



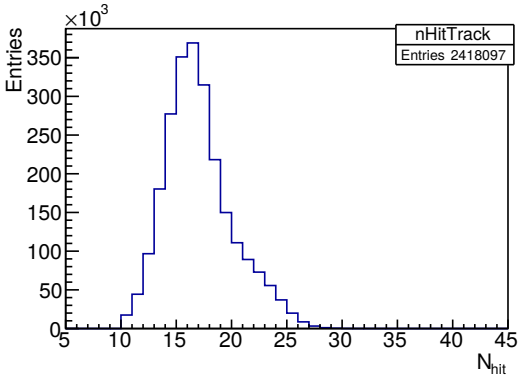
(a) Kosmische Strahlung: Es werden bei der Selektion mindestens acht Spurpunkte verlangt, weshalb die Verteilung links eine Kante aufweist.



(b) Isoliertes Myon



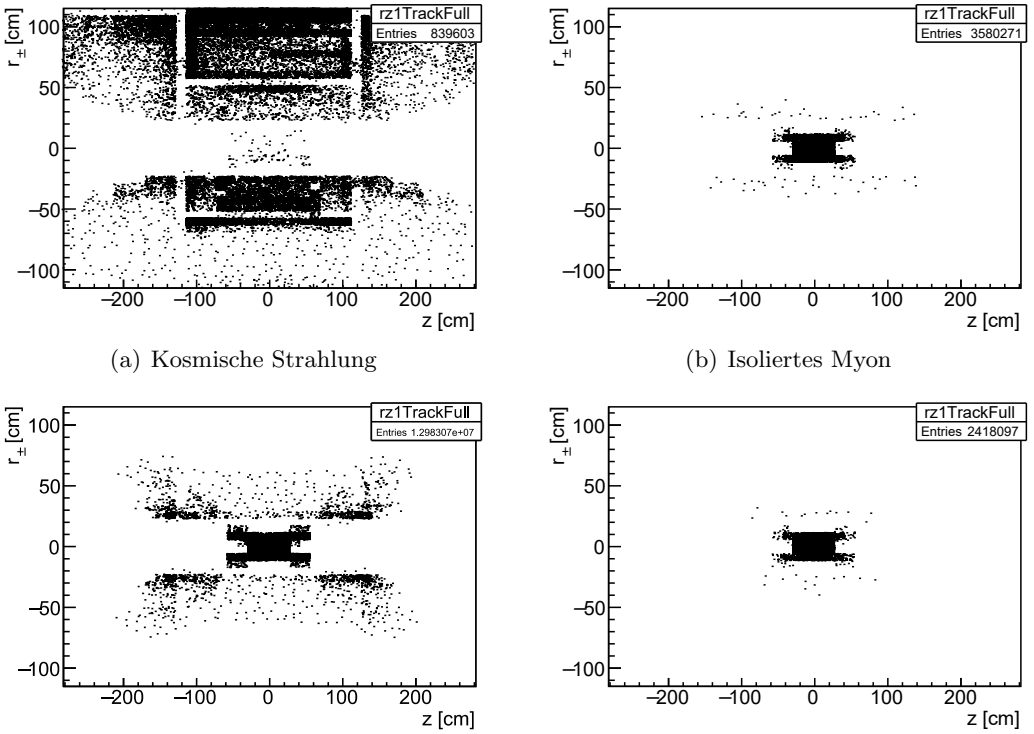
(c) Minimum-Bias: Es werden bei der Selektion mindestens zehn Spurpunkte verlangt, weshalb die Verteilung links eine Kante aufweist.



(d) $Z \to \mu\mu$

Abbildung A.3.: Anzahl der Spurpunkte, die von den selektierten Teilchenspuren im Spurdetektor hinterlassen wurden.

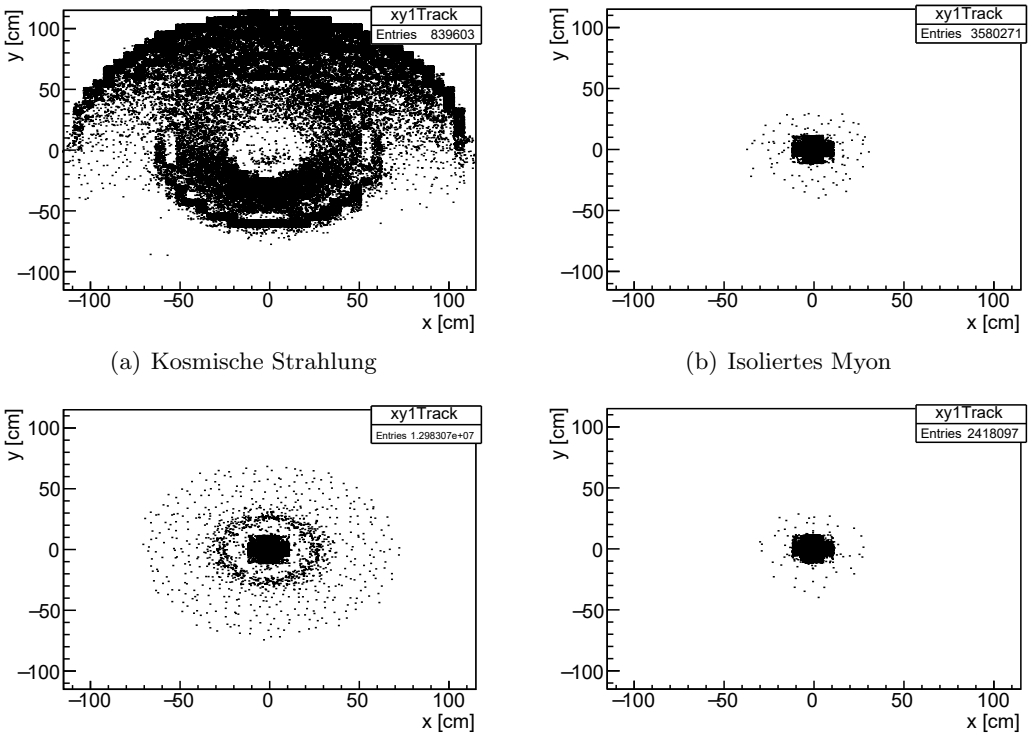
Zu Abbildung A.3: Im Vergleich zu den anderen Verteilungen reicht die Verteilung der kosmischen Teilchenspuren zu den höchsten Anzahlen an Spurpunkten. Dies entspricht der Erwartung, da kosmische Teilchen den gesamten Querschnitt des Detektors durchlaufen können, weil sie nicht vom Kollisionspunkt ausgehen.



(a) Kosmische Strahlung (b) Isoliertes Myon

(c) Minimum-Bias (d) $Z \to \mu\mu$

Abbildung A.4.: Position des ersten Spurpunktes jeder selektierten Teilchenspur in der rz-Ebene.



(a) Kosmische Strahlung (b) Isoliertes Myon

(c) Minimum-Bias (d) $Z \to \mu\mu$

Abbildung A.5.: Position des ersten Spurpunktes jeder selektierten Teilchenspur in der xy-Ebene.

B. Beispiel einer Alignment-Konfigurationsdatei (.ini)

Programmausdruck 8.1: Beispiel einer Alignment-Konfigurationsdatei (.ini) mit vier Datensätzen.

```
[general]
classInf       = cmscaf1nd:cmscafspec1nw
jobname        = MP2015
pedeMem        = 32000
datasetdir     = /afs/cern.ch/cms/CAF/CMSALCA/ALCA_TRACKERALIGN/MP/MPproduction/
    datasetfiles
configTemplate = universalConfigTemplate.py
globaltag      = 74X_dataRun2_Candidate_2015_11_03_11_22_18

[dataset:ZMuMu]
collection     = ALCARECOTkAlZMuMu
inputFileList  = ${datasetdir}/Collisions2015/Run2015D/DoubleMuon_Run2015D-
    TkAlZMuMu-PromptReco-v3_ALCARECO.txt
json           = /afs/cern.ch/cms/CAF/CMSALCA/ALCA_TRACKERALIGN/MP/MPproduction/
    datasetfiles/Collisions2015/Run2015D/json_DCSONLY_2015-11-05.txt

[dataset:Cosmics]
collection     = ALCARECOTkAlCosmicsCTF0T
inputFileList  = ${datasetdir}/Cosmics2015/interfill/Run2015D/Cosmics_Run2015D-
    TkAlCosmics0T-PromptReco-v4_ALCARECO_38T.txt
cosmicsDecoMode  = true
cosmicsZeroTesla = false
njobs          = 20

[dataset:IsoMu]
collection     = ALCARECOTkAlMuonIsolated
inputFileList  = ${datasetdir}/Collisions2015/Run2015D/SingleMuon_Run2015D-
    TkAlMuonIsolated-PromptReco-v3_ALCARECO_rsplit6pc.txt
json           = /afs/cern.ch/cms/CAF/CMSALCA/ALCA_TRACKERALIGN/MP/MPproduction/
    datasetfiles/Collisions2015/Run2015D/json_DCSONLY_2015-11-05.txt

[dataset:MinBias]
collection     = ALCARECOTkAlMinBias
inputFileList  = ${datasetdir}/Collisions2015/Run2015D/HLTPhysics_Run2015D-
    TkAlMinBias-PromptReco-v3_ALCARECO_rsplit2pc.txt
njobs          = 19
weight         = 0.02
json           = /afs/cern.ch/cms/CAF/CMSALCA/ALCA_TRACKERALIGN/MP/MPproduction/
    datasetfiles/Collisions2015/Run2015D/json_DCSONLY_2015-11-05.txt
```

Literaturverzeichnis

[1] L.R. Evans ; P. Bryant: LHC Machine. In: *J. Instrum.* 3 (2008), S08001. 164 p. http://cds.cern.ch/record/1129806. – This report is an abridged version of the LHC Design Report (CERN-2004-003)

[2] F. Marcastel: CERN's Accelerator Complex. La chaîne des accélérateurs du CERN. (2013), Oct. https://cds.cern.ch/record/1621583. – abgerufen am 10. Mai 2016

[3] *About CERN: CMS-Experiment*. http://home.cern/about/experiments/cms. – abgerufen am 14. Juni 2016

[4] CMS Collaboration: The CMS experiment at the CERN LHC. In: *Journal of Instrumentation* 3 (2008), Nr. 08, S08004. http://stacks.iop.org/1748-0221/3/i=08/a=S08004

[5] CMS collaboration: Alignment of the CMS tracker with LHC and cosmic ray data. In: *Journal of Instrumentation* 9 (2014), Nr. 06, P06009. http://stacks.iop.org/1748-0221/9/i=06/a=P06009

[6] D. Barney: *CMS Detector Slice*. https://cds.cern.ch/record/2120661. Version: Jan 2016. – abgerufen am 10. Mai 2016

[7] CMS Collaboration: *CMS Physics: Technical Design Report Volume 1: Detector Performance and Software*. CERN, 2006 (Technical Design Report CMS). http://cds.cern.ch/record/922757

[8] W. Adam u. a.: Alignment of the CMS silicon strip tracker during stand-alone commissioning. In: *Journal of Instrumentation* 4 (2009), Nr. 07, T07001. http://stacks.iop.org/1748-0221/4/i=07/a=T07001

[9] G. Flucke u. a.: CMS silicon tracker alignment strategy with the Millepede II algorithm. In: *Journal of Instrumentation* 3 (2008), Nr. 09, P09002. http://stacks.iop.org/1748-0221/3/i=09/a=P09002

[10] J. Draeger: *Track based alignment of the CMS silicon tracker and its implication on physics performance*, Hamburg U., Diss., 2011. http://www-library.desy.de/cgi-bin/showprep.pl?thesis11-026

[11] CMS Collaboration: Description and performance of track and primary-vertex reconstruction with the CMS tracker. In: *Journal of Instrumentation* 9 (2014), Nr. 10, P10009. http://stacks.iop.org/1748-0221/9/i=10/a=P10009

[12] P. Vanlaer u. a.: Impact of CMS Silicon Tracker Misalignment on Track and Vertex Reconstruction / CERN. Version: Jan 2006. https://cds.cern.ch/record/927378. Geneva, Jan 2006 (CMS-NOTE-2006-029. 1). – Forschungsbericht

[13] CMS COLLABORATION: Identification of b-quark jets with the CMS experiment. In: *Journal of Instrumentation* 8 (2013), Nr. 04, P04013. http://stacks.iop.org/1748-0221/8/i=04/a=P04013

[14] A. OSTAPTCHOUK u. a.: The Alignment System of the CMS Tracker / CERN. Version: Nov 2001. http://cds.cern.ch/record/687389. Geneva, Nov 2001 (CMS-NOTE-2001-053). – Forschungsbericht

[15] V. KARIMÄKI ; T. LAMPEN ; F. SCHILLING: The HIP Algorithm for Track Based Alignment and its Application to the CMS Pixel Detector / CERN. Version: Jan 2006. http://cds.cern.ch/record/926537. Geneva, Jan 2006 (CMS-NOTE-2006-018). – Forschungsbericht

[16] V. BLOBEL: *Millepede II Linear Least Squares Fits with a Large Number of Parameters*. Institut für Experimentalphysik Hamburg U., 2007. http://www.desy.de/~blobel/Mptwo.pdf

[17] V. BLOBEL ; E. LOHRMANN: *Statistische und numerische Methoden der Datenanalyse*. Stuttgart : Teubner, 1998 (Teubner-Studienbücher : Physik). http://www.desy.de/~blobel/eBuch.pdf. – ISBN 3–519–03243–0

[18] M. STOYE: *Calibration and alignment of the CMS silicon tracking detector*, Hamburg U., Diss., 2007. http://dx.doi.org/10.3204/DESY-THESIS-2007-026. – DOI 10.3204/DESY–THESIS–2007–026

[19] C.C. PAIGE ; M.A. SAUNDERS: Solution of Sparse Indefinite Systems of Linear Equations. In: *SIAM Journal on Numerical Analysis* 12 (1975), Nr. 4, 617-629. http://dx.doi.org/10.1137/0712047. – DOI 10.1137/0712047

[20] *CERN batch computing service*. http://information-technology.web.cern.ch/services/batch. – abgerufen am 10. Mai 2016

[21] *CMSSW Project Index*. http://cms-sw.github.io/index.html. – abgerufen am 10. Mai 2016

[22] *The CMS Offline WorkBook*. https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBook. – abgerufen am 10. Mai 2016

[23] *Python Programming Documentation*. https://www.python.org/doc/. – abgerufen am 10. Mai 2016

[24] V. BLOBEL ; C. KLEINWORT ; F. MEIER: Fast alignment of a complex tracking detector using advanced track models. In: *Comput. Phys. Commun.* 182 (2011), S. 1760–1763. http://dx.doi.org/10.1016/j.cpc.2011.03.017. – DOI 10.1016/j.cpc.2011.03.017

[25] *The MillePede Production System (MPS)*. https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideMillepedeProductionSystem. – abgerufen am 10. Mai 2016

[26] *Perl Programming Documentation*. http://perldoc.perl.org/. – abgerufen am 10. Mai 2016

[27] *Workflow for MillePede-Based Alignment*. https://twiki.cern.ch/twiki/pub/CMSPublic/SWGuideMillepedeProductionSystem/millepede-workflow-700.png. – abgerufen am 10. Mai 2016

[28] *Description of the cmsRun Python Configuration Syntax*. https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideAboutPythonConfigFile. – abgerufen am 10. Mai 2016

[29] *CERN AFS Service*. http://information-technology.web.cern.ch/services/afs-service. – abgerufen am 10. Mai 2016

[30] *CERN EOS Service*. http://information-technology.web.cern.ch/services/eos-service. – abgerufen am 10. Mai 2016

[31] *Zsh shell and scripting language*. http://www.zsh.org/. – abgerufen am 10. Mai 2016

[32] *MPS States*. https://twiki.cern.ch/twiki/pub/CMSPublic/SWGuideMillepedeProductionSystem/MPS-states.png. – abgerufen am 10. Mai 2016

[33] M.J. French u. a.: Design and results from the APV25, a deep sub-micron CMOS front-end chip for the CMS tracker. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 466 (2001), Nr. 2, S. 359 – 365. http://dx.doi.org/10.1016/S0168-9002(01)00589-7. – DOI 10.1016/S0168–9002(01)00589–7. – ISSN 0168–9002

[34] M. Raymond u. a.: Final Results from the APV25 Production Wafer Testing. (2005). https://cds.cern.ch/record/922784

[35] G. Mittag: *CMSSW Common Alignment Trackselection and Refitting Tool*. https://github.com/cms-sw/cmssw/blob/CMSSW_8_0_X/Alignment/CommonAlignment/python/tools/trackselectionRefitting.py. – abgerufen am 10. Mai 2016