Fachhochschule Aachen
Campus Jülich

Fachbereich: Medizintechnik und Technomathematik
Studiengang: Scientific Programming

Entwicklung eines schnellen Algorithmus zur Suche von Teilchenspuren im „Straw Tube Tracker“ des $\overline{\text{P}}$ANDA-Detektors

Bachelorarbeit von Jette Schumann

Jülich, den 15. August 2013

Eigenständigkeitserklärung

Diese Arbeit ist von mir selbstständig angefertigt und verfasst. Es sind keine anderen als die angegebenen Quellen und Hilfsmittel benutzt worden.

Ort und Datum	Unterschrift

Diese Arbeit wurde betreut von:

Erstprüfer:	Prof. Dr. Andreas Terstegge
Zweitprüfer:	Günter Sterzenbach

Die vorliegende Bachelorarbeit wurde am Institut für Kernphysik der Forschungszentrum Jülich GmbH erstellt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Erforschung der inneren Struktur von Hadronen (wie z. B. Protonen) ist eines der Hauptthemen der Mittel- und Hochenergiephysik. Es soll geklärt werden, aus welchen Elementarteilchen sie aufgebaut sind und welche Kräfte zwischen diesen wirken. Um diese Fragestellungen zu untersuchen, werden geladene Teilchen in Teilchenbeschleunigern auf Energien von mehreren GeV bis TeV beschleunigt und anschließend aufeinander oder auf ruhende Targets geschossen. Durch den Zusammenstoß entstehen neue, kurzlebige Teilchen. Diese zerfallen nahezu sofort wieder in eine Vielzahl von Sekundär-Teilchen, für deren Messung ein komplexes Detektor-System erforderlich ist. Moderne Detektoren erlauben eine präzise Messung der Flugbahnen der Teilchen, ihrer Energien und Impulse sowie eine Identifikation der entstandenen Teilchen.

Die Detektoren liefern eine große Menge an Signalen für alle Teilchen, die aus dem Zusammenstoß resultieren. Die anfallende Datenmenge muss in Echtzeit reduziert werden, da nicht alle Informationen abgespeichert werden können. An dieser Stelle muss Software eingesetzt werden, die physikalisch interessante Ereignisse erkennt und nur diese abspeichert. Dazu sind Algorithmen nötigt, die die Gesamtmenge an Daten schnell auswerten und diese u. a. in Informationen über die Flugbahnen der einzelnen Teilchen überführen. Diese Aufgabe wird von sogenannten Trackfinding-Algorithmen übernommen. In der vorliegenden Arbeit wird ein derartiger Trackfinder für den zentralen Spurdetektor des $\overline{\text{P}}$ANDA-Experiments vorgestellt.

Mit dem $\overline{\text{P}}$ANDA-Experiment werden in Zukunft Zusammenstöße von Antiprotonen mit Protonen untersucht. Da es dieser Arbeit zugrunde liegt, wird es im Kapitel 2 als Teil der Beschleunigeranlage FAIR eingeführt. Der wesentliche Aufbau des $\overline{\text{P}}$ANDA-Detektors wird beschrieben, wobei gesondert auf den Straw Tube Tracker (STT) des $\overline{\text{P}}$ANDA-Detektors eingegangen wird. Der STT spielt eine wesentliche Rolle, da ein Trackfinding-Verfahren für dessen Signal-Auswertung vorgestellt wird.

Anschließend wird in Kapitel 3 analysiert, welche Anforderungen an solch ein Verfahren für den STT gestellt werden. Der Verlauf von Flugbahnen im STT wird untersucht und es wird herausgestellt, welche Besonderheiten es bei der Rekonstruktion dieser Flugbahnen zu beachten gibt. Zur Simulation des $\overline{\text{P}}$ANDA-Experiments wird das Framework „PandaRoot“ eingesetzt, das es erlaubt, die entwickelte Software zu testen und zu verbessern. Da das Verfahren in „PandaRoot“ integriert werden soll, wird die grundlegende Funktionsweise dieses Frameworks vorgestellt.

In Kapitel 4 wird das entwickelte Verfahren detailliert beschrieben. Zum Finden der ersten Wegstücke wird ein zellulärer Automat eingesetzt. Da dieser zum Auffinden von vollständigen Teilchenspuren nicht ausreichend ist, schließen sich weitere Schritte an, mit denen die Wegstücke vervollständigt werden. Dabei wird der Riemann-Fit zur Approximation von Kreisbahnen eingesetzt. Bei der Entwicklung des Verfahrens wurde großer Wert darauf gelegt, dass es sich parallelisieren lässt.

Die Implementierung des Algorithmus wird in Kapitel 5 beschrieben. Dabei handelt es sich vorläufig nicht um ein parallelisiertes Programm. Operationen, die parallel abgearbeitet werden könnten, wurden sequentiell gelöst. Die entwickelten Klassen mit den verwendeten Datenstrukturen und verfassten Methoden werden vorgestellt. Das Verfahren wurde so implementiert, dass es in das Simulations-Framework eingebunden und getestet werden kann.

Daraufhin wird im Kapitel 6 das Trackfinding-Verfahren bewertet. Es wird überprüft, wie weit die physikalischen Flugbahnen rekonstruiert werden. Zur automatischen Analyse der vom Verfahren generierten Flugbahnen wurde eine zusätzliche Klasse entwickelt. Es wird vorgestellt, welche Informationen zur Qualitätsanalyse zur Verfügung stehen, welche Kriterien untersucht worden sind und welche Ergebnisse die Tests geliefert haben.

Im letzten Kapitel wird ein Fazit gezogen und es werden mögliche Erweiterungen und Verbesserungen des Verfahrens in Aussicht gestellt.

2 Zugrundeliegendes Experiment

2.1 FAIR - Facility for Antiproton and Ion Research

FAIR ist eine internationale Forschungseinrichtung in Darmstadt in der mit Antiprotonen- und Ionenstrahlen experimentiert werden wird. Es handelt sich um eine komplexe Beschleunigeranlage, die zur Zeit in der Entstehungsphase ist und 2018 fertiggestellt werden soll. Es werden Komponenten der *Gesellschaft für Schwerionenforschung* (GSI) integriert, die als Vorbeschleuniger dienen. FAIR bietet den Vorteil, dass unterschiedliche Physikprogramme parallel betrieben werden können. Es wird ein Fokus auf diese fünf Forschungsbereiche gelegt: Kernstruktur-, Antiprotonen-, Kernmaterie-, Plasma- und Atomphysik. Das Ziel von FAIR ist es, grundlegende Fragen zum Verständnis des Universums zu beantworten. [1]

Herzstück der Beschleunigeranlage ist ein Doppelring mit einem Umfang von circa 1100 m, der 17 m unter der Erde verlegt wird. Eine schematische Darstellung dieser Anlage ist in Abbildung 2.1 zu sehen. Die bereits bestehenden Beschleuniger der GSI werden als Vorbeschleuniger genutzt. FAIR wird im Anschluss Protonen auf bis zu 29 GeV, leichte Kerne auf bis zu 14 GeV pro Nukleon und schwere Kerne auf bis zu 11 GeV pro Nukleon beschleunigen. Die hochenergetischen Teilchen können dann z. B. auf Materieproben wie dünne Metallfolien geschossen oder mit Stoßexperimenten untersucht werden. So soll der Aufbau der Materie insbesondere die starke Kraft zwischen Quarks und die Evolution des Universums erforscht werden.

Da sich diese Arbeit mit einer Software für ein Experiment mit Antiprotonen beschäftigt, wird an dieser Stelle nur auf die diesbezügliche Physik eingegangen. Antiprotonen kommen in der Natur für gewöhnlich nicht vor und müssen für die Experimente erst erzeugt werden. Zu diesem Zweck werden Protonen über ein mehrstufiges System (siehe Abb. 2.1: p-LINAC, SIS18 und SIS100) auf bis zu 29 GeV beschleunigt und auf ein Wolfram-Target geschossen. Bei den Kernreaktionen in diesem massiven Target entstehen u. a. die gewünschten Antiprotonen. Die Antiprotonen werden im *Collector Ring* (CR) gesammelt und anschließend in den *High Energy Storage Ring* (HESR) eingespeist. Der HESR beschleunigt die Antiprotonen auf einen Strahlimpuls von 1.5 bis 15 GeV/c

[1] Die hier aufgeführten Informationen wurden den Internetauftritten der GSI [4] und FAIR GmbH [9] entnommen.

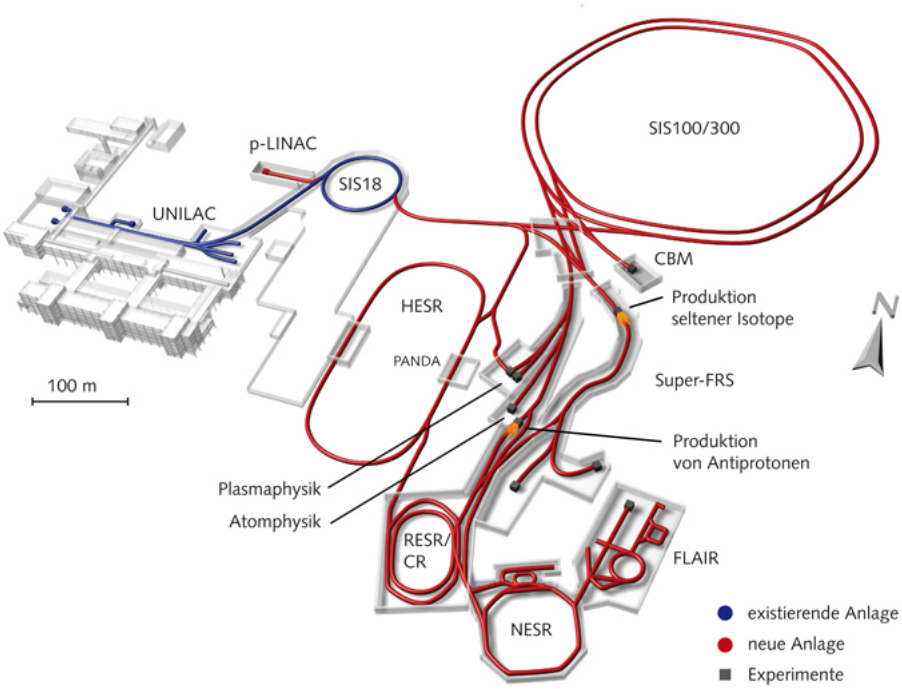

Abbildung 2.1: Schematische Ansicht der geplanten Beschleunigeranlage in Darmstadt. Die Komponenten von FAIR sind rot gekennzeichnet. [8]

und erlaubt eine Impulsauflösung von unter 10^{-4} $\frac{\Delta p}{p}$. An diesem Ring befindet sich das zugrundeliegende Experiment dieser Arbeit namens $\overline{\text{P}}$ANDA, auf welches im folgenden Abschnitt genauer eingegangen wird.

2.2 $\overline{\text{P}}$ANDA - AntiProton Annihilations at Darmstadt

Das $\overline{\text{P}}$ANDA-Experiment ist eine der wesentlichen Komponenten von FAIR. In der Zukunft werden mit $\overline{\text{P}}$ANDA Antiproton-Proton Annihilationen und Reaktionen von Antiprotonen mit schwereren Kernen untersucht. Als Annihilation wird der Prozess der Paarvernichtung bezeichnet. Dieser beschreibt den Zusammenstoß eines fundamentalen Elementarteilchens und des zugehörigen Antiteilchens. Im Folgenden wird auf das Ziel und den Ablauf des Experimentes sowie auf das Detektorsystem zur Erfassung der Daten eingegangen.

2.2.1 Das Experiment

Mit $\overline{\text{P}}$ANDA sollen grundsätzliche Fragen aus dem Bereich der Kern- und Teilchenphysik beantwortet werden. Das physikalische Programm lässt sich in vier wesentliche Teilgebiete untergliedern. Die Erforschung der *Nukleonenstruktur* soll Aufschluss über den genauen inneren Aufbau von Nukleonen geben. Mit der Untersuchung von *„in-medium effects“* soll die Frage beantwortet werden, warum Hadronen sich in gebundenen Zuständen anders verhalten als freie Teilchen. Im Bereich der *Hadronen Spektroskopie* sollen mögliche (exotische) Anregungszustände von Hadronen und deren Erzeugung und Zerfall untersucht werden. Außerdem werden *Hyperkerne* erforscht, welche Strange-Quarks enthalten. Die Einführung der Strangeness als weiterer Freiheitsgrad würde zu neuartigen Elementen im Periodensystem führen, die untersucht werden sollen. Einen vertiefenden Einblick in diese Gebiete bietet der Physik-Report von $\overline{\text{P}}$ANDA [1].

Der *High Energy Storage Ring* (HESR) liefert für diesen Zweck einen Antiprotonen-Strahl mit einer hohen Qualität und Intensität von bis zu 15 GeV. Der Speicherring schließt sich an die Beschleunigeranlage von FAIR an und hat die Form einer Rennbahn mit zwei parallelen geraden Streckenabschnitten. Die Gesamtlänge beläuft sich auf 575 m. Der so zur Verfügung gestellte Strahl wird in einem Vakuum mit Targets kollidieren. $\overline{\text{P}}$ANDA ermöglicht es, verschiedene Target-Systeme einzusetzen, sodass die Targets flexibel ausgetauscht werden können. Zu den wichtigsten Target-Systemen zählen das *Cluster Jet Traget*, das *Hydrogen Pellet Target* und *Nuclear Targets*. Angaben zu ihrem Aufbau und ihrer Funktionsweise können dem technischen Design-Report von $\overline{\text{P}}$ANDA [2] entnommen werden.

2.2.2 Der $\overline{\text{P}}$ANDA-Detektor

Innerhalb des $\overline{\text{P}}$ANDA-Detektors wird der Antiprotonenstrahl aus dem HESR auf ein stationäres Target gerichtet. Um alle relevanten Informationen zu erfassen, ist ein komplexes Detektorsystem erforderlich, welches in Abbildung 2.2 dargestellt ist.[2]

Das gesamte Detektorsystem setzt sich aus zwei Magnetspektrometern zusammen: aus einem *Target Spectrometer* und dem *Forward Spectrometer*. Das *Target Spectrometer* wird sich direkt um den Stoßpunkt befinden und Teilchen nahe der Kollisionsregion registrieren. Die zugehörigen Detektor-Komponenten werden zwiebelschalenartig um den Stoßpunkt aufgebaut. Das *Forward Spectrometer* wird alle Teilchen erfassen, die sich mit einem vertikalen bzw. horizontalen Winkel von bis zu $\pm 5^\circ$ bzw. $\pm 10^\circ$ vom Stoßpunkt in Richtung des Antiprotonenstrahls entfernen.

[2] Grundlage folgender Ausarbeitung ist der technische Design-Report von $\overline{\text{P}}$ANDA [2].

Zur Messung der Impulse von geladenen Teilchen wird im *Target Spectrometer* ein 2 T Solenoid-Magnetfeld[3] und im *Forward Spectrometer* ein 2 Tm Dipol-Magnetfeld erzeugt.

Magnetspektrometer nutzen die Wirkung der Lorentz-Kraft aus, welche auf geladene Teilchen wirkt, die sich in einem Magnetfeld bewegen. Die Krafteinwirkung führt zu einer Ablenkung der Teilchen von ihrer eigentlichen Flugbahn. Die Lorentzkraft ist von der Ladung und Geschwindigkeit des Teilchens sowie von der Stärke des Magnetfeldes abhängig. Dies erlaubt es, von der Ablenkung bzw. von der Flugbahn des Teilchens im Magnetspektrometer auf den Impuls zu schließen.

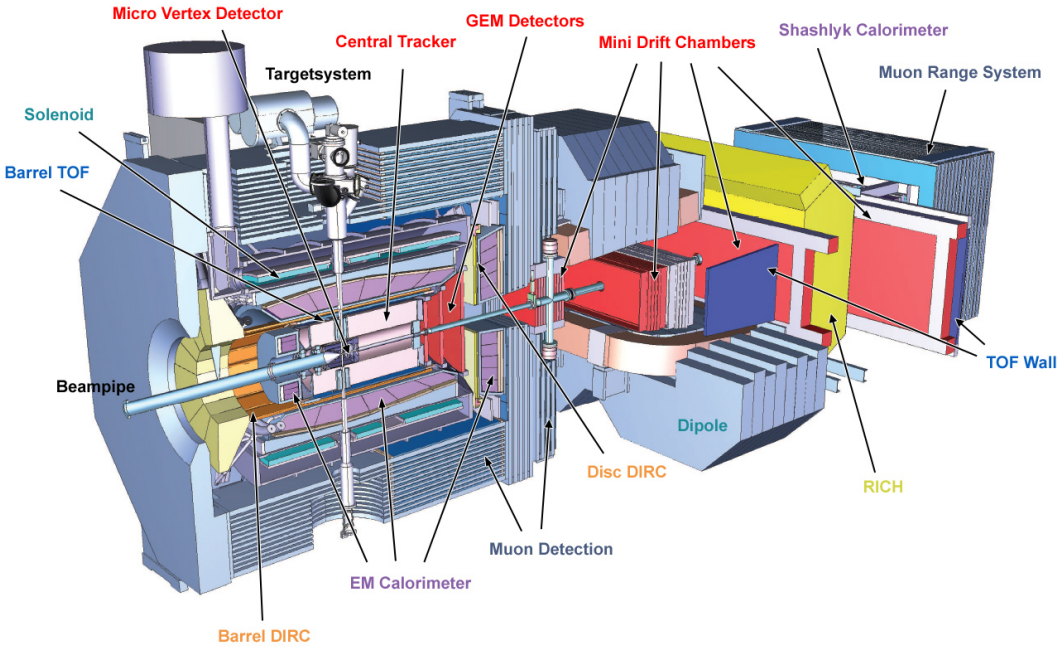

Abbildung 2.2: Geplanter Aufbau des $\overline{\text{P}}$ANDA-Detektors entlang des Antiprotonen-Strahls. [3]

Die einzelnen Detektor-Komponenten lassen sich in Bezug auf ihre primären Ziele der Messung in spurgebende Detektoren, Detektoren zur Teilchenidentifikation und zur Energiebestimmung einteilen.

Spurgebende Detektoren dienen der Auflösung der Flugbahnen der Teilchen. Zu ihnen zählen der *Micro Vertex Detector* (MVD), der *Straw Tube Tracker* (STT), der *Gas Electron Multiplier Detector* (GEM) sowie der *Forward Tracker* (FT). Die drei ersteren sind Bestandteile des *Target Spectrometers*. Der MVD erlaubt eine präzise Spurvermessung von geladenen Teilchen nahe am

[3]Ein Solenoid ist eine Zylinderspule, die es ermöglicht, ein nahezu vollständig homogenes Magnetfeld zu erzeugen.

Wechselwirkungspunkt. Der STT wird auch als „Central Tracker“ bezeichnet und ist im Abschnitt 2.3 genauer beschrieben. Teilchen, deren Flugbahn einen kleinen Winkel zur Strahlachse haben werden vom GEM bzw. vom FT gemessen.

Zur Identifizierung der Teilchen werden ein *Detection of internally Reflected Cherenkov light-* (DIRC), ein *Time of Flight-* (TOF), ein Myonen- und ein *Ring Imaging Cherenkov*-Detektor (RICH) eingesetzt. Der DIRC-Detektor befindet sich im *Target Spectrometer* und der RICH-Detektor im *Forward Spectrometer*. Die Komponenten des TOF- und Myonen-Detektors sind in beiden Spektrometern wiederzufinden. Der DIRC- und RICH-Detektor tragen durch das Nutzen des Cherenkov-Effekts zur Bestimmung der Teilchenart bei, während der TOF-Detektor eine präzise Zeitmessung ermöglicht, die zur Teilchenidentifikation genutzt wird.

Elektromagnetische Kalorimeter in beiden Magnetspektrometern ermöglichen eine Energiebestimmung von Teilchen, die elektromagnetisch wechselwirken (Elektronen, Positronen und Gamma-Teilchen).

Zur Reduktion der anfallenden Datenmenge in einem solchen Experiment ist eine Vorselektion der physikalisch interessanten Ereignisse notwendig. Dazu wird üblicherweise ein mehrstufiges System eingesetzt, dessen erste Stufe ein Hardware-Trigger ist. Beim Einsatz eines Hardware-Triggers wird nur ein kleiner Bruchteil der gesamten anfallenden Datenmenge eines Ereignisses ausgelesen. Diese Daten werden analysiert und anhand der Ergebnisse wird entschieden, ob das Ereignis physikalisch interessant ist. Ist dies der Fall, werden die gesamten Detektordaten zu diesem Ereignis ausgelesen und weiter untersucht. Andernfalls werden die Daten gelöscht. Für $\overline{\text{P}}$ANDA trifft das aber nicht zu. Alle Detektorkomponenten messen ununterbrochen, was in einer riesigen Datenmenge an Signalen resultiert. Bei einer Ereignisrate von $2 * 10^7/s$ fallen 100 GByte/s an Daten an. Aus diesem Grund wird zum Verarbeiten der Signale schnelle Software benötigt, die die Ereignisse nach bestimmten Kriterien filtert, sodass nur noch 100 MByte/s anfallen, die gespeichert werden müssen.

2.3 STT - Straw Tube Tracker

Der Straw Tube Tracker ist der zentrale Spurdetektor von $\overline{\text{P}}$ANDA. In Abbildung 2.2 ist er in rot mit „Central Tracker“ beschriftet. Wie zu sehen, ist er eine Komponente des *Target Spectrometers* und umschließt den Micro Vertex Detektor.[4]

Der Detektor besteht aus 4636 einzelnen *Straw Tubes* (Driftröhrchen), welche in Abbildung 2.3 dargestellt sind. Bei den Straw Tubes handelt es sich um gasgefüllte Röhrchen in der Form eines Zylinders, die durch einen Überdruck

[4]Die folgenden Informationen über den Aufbau und die Funktionsweise des STTs stammen aus dem technischen Design-Report vom STT [2].

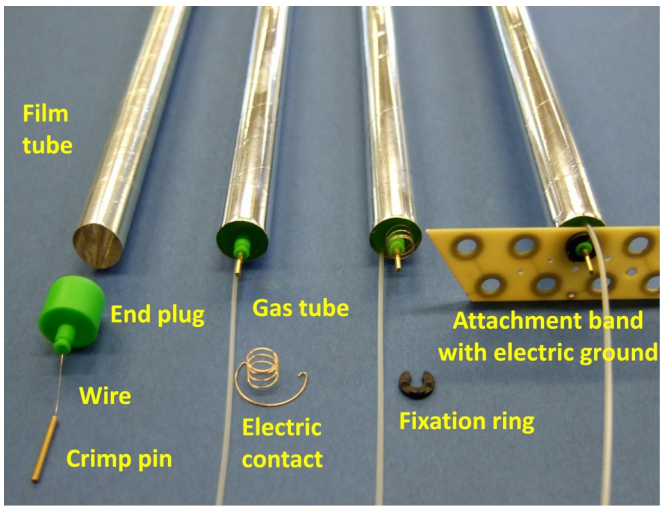

Abbildung 2.3: Komponenten der Straw Tubes [11]

von 1 bar stabilisiert werden. Sie besitzen eine leitende Innenbeschichtung als Kathode und einen Draht entlang der Zylinder-Achse, der als Anode fungiert. Die Wand des Zylinders besteht aus einer aluminiumbeschichteten Mylar-Folie. Die Straw Tubes sind 1500 mm lang und haben einen (inneren) Durchmesser von 10 mm. An den Enden der Röhrchen ist jeweils ein Endstopfen aus Thermoplaste angebracht, welcher es ermöglicht alle Straw Tubes in einem Rahmen neben- bzw. übereinander anzuordnen.

Zwischen dem Draht und der Hülle der Straw Tubes besteht ein elektrisches Feld. Passiert ein geladenes Teilchen den gasgefüllten Innenraum, kommt es zur Ionisation. Den Gas-Molekülen wird ein Elektron entfernt und es bleibt ein positives Ion zurück. Aufgrund des elektrischen Feldes wandern die Elektronen zum positiv geladenen Draht und die positiven Ionen bewegen sich zur Kathode (also zum Rand der Straw Tubes). In den Straw Tubes ist das elektrische Feld um den Draht herum stark genug, um weitere Ionisationen durch das Elektron auszulösen. Dadurch wird das Signal verstärkt und kann gemessen werden. Durch das Messen der sogenannten Drift-Zeit des Elektrons, das als erstes den Draht erreicht, kann der minimale Abstand der Teilchenspur von dem Draht berechnet werden. Es lassen sich Isochrone um den Draht herum bestimmen, die alle Punkte beinhalten, von denen die gleiche Drift-Zeit ausgeht. Die Teilchenspuren können dann basierend auf den Isochronen benachbarter Straw Tubes durch den am besten passenden Fit rekonstruiert werden.

Die Straw Tubes werden in 27 ebenen Schichten angeordnet und in einer hexagonalen Form um den MVD platziert. Dies kann in Abbildung 2.4 nachvollzogen werden. Acht dieser Schichten sind schräg mit einem Winkel von $\pm 2.9°$ angeordnet. Dabei sind die Reihen paarweise entgegengesetzt ausgerichtet. Auf diese Weise wird eine Information über die z-Komponente der passierenden Teilchen (also in Richtung des Strahls) erhalten. Wird von zwei

übereinanderliegenden entgegengesetzten Straw Tubes ein Signal ausgelöst, lässt sich aus der Kombination der beiden Signale die z-Koordinate der Teilchenspur berechnen. Die Straw Tubes der restlichen Schichten ermöglichen lediglich die Messung einer x- und y-Komponente senkrecht zur z-Achse.

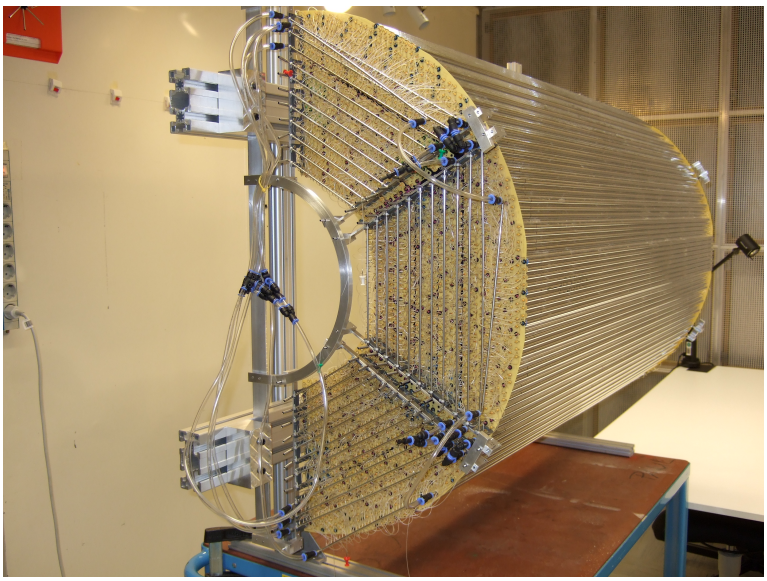

Abbildung 2.4: Halbschale eines $\overline{\text{P}}$ANDA-STT-Prototypen. In der Mitte des Hexagons wird sich der MVD befinden. [19]

3 Analyse des Problems

Mit $\overline{\text{P}}$ANDA werden in der Zukunft Stoßexperimente durchgeführt, deren Daten von einer komplexen Detektoranlage erfasst werden. Mit den Signalen der Detektoren sollen die Spuren der geladenen Teilchen rekonstruiert und die Art der entstandenen Teilchen bestimmt werden. Um die Flugbahnen von Teilchen nachzubilden, müssen sämtliche Informationen spurgebender Detektoren verarbeitet und zusammengeführt werden. Dies erlaubt die Bestimmung der Impulse der Teilchen, sowie eine hohe Ortsauflösung des Ursprungs von Teilchenspuren (Vertex) und der Nebenvertices beim Zerfall von Teilchen. In dieser Arbeit steht das Finden von Spuren der Partikel innerhalb des Straw Tube Trackers im Fokus. Im Folgenden wird diese Aufgabe als erstes genauer definiert und analysiert. Anschließend wird das Programm PandaRoot eingeführt, welches zur Simulation von $\overline{\text{P}}$ANDA-Experimenten dient.

3.1 Problemstellung

Es soll ein Verfahren entwickelt werden, das die Signale des Straw Tube Trackers den verschiedenen Spuren der Teilchen zuordnet. Das heißt, nach Anwendung des Verfahrens liegen die einzelnen Signale der Straw Tubes nach Spuren (Tracks) sortiert vor. Das Verfahren ist in die Kategorie der Trackfinding-Algorithmen einzuordnen. Es wird die Menge aller Signale der Straw Tubes (Hits) zur Verfügung gestellt und der Algorithmus soll aus diesen Hits die Tracks der Teilchen rekonstruieren. Zur Berechnung liegen nur die Hit-Informationen vor, demzufolge gibt es keine Angaben zur Anzahl oder Art der Tracks. Es werden allerdings nicht die genauen Flugbahnen der Teilchen berechnet, wie es beim Trackfitting der Fall ist. Dieser Schritt setzt erst nach dem Trackfinding ein und setzt die Gruppierung der Hits zu potenziellen Tracks voraus.

Das Verfahren muss die Tracks schnell ermitteln, da es als ein Online-Trackfinder eingesetzt werden soll. Online-Algorithmen müssen in Echtzeit einen kontinuierlichen Datenstrom verarbeiten. Das bedeutet, dass während des Experimentablaufs stets die Daten der STT-Hits übermittelt werden. Dies ist erforderlich, weil $\overline{\text{P}}$ANDA ohne einen Hardware-Trigger arbeitet. Die Detektor-Komponenten messen fortlaufend Ereignisse und liefern eine riesige zu verarbeitende Datenmenge. Aufgrund des begrenzten Speicherplatzes müssen die Daten online gefiltert werden. Dafür ist eine Zusammenführung der Signale

eines Ereignisses erforderlich, welche dann bewertet wird. Nicht interessante Ereignisse werden nicht abgespeichert. Damit dieser Vorgang schnell erfolgen kann, sollte das Verfahren möglichst parallelisierbar sein. Dies ermöglicht eine effiziente Berechnung z. B. auf einer Graphics Processing Unit (GPU).

Um das Verfahren anwenden zu können, sollen entsprechende Programmmodule entwickelt werden, die sich in das derzeit zur Experiment-Simulation eingesetzte Framework PandaRoot einbinden lassen und dessen Richtlinien folgen. Das Verfahren muss noch nicht mit einem sich fortlaufend ändernden Datenstrom als Eingabe arbeiten können. Dies ist aktuell in PandaRoot noch nicht vorgesehen und in der Entwicklung. Ein besonderes Augenmerk wird daher vorerst auf die Parallelisierbarkeit gelegt. Außerdem soll der Trackfinding-Algorithmus geeignet getestet werden, um die Güte des Verfahrens bewerten zu können. Dafür erforderliche Bewertungs-Kriterien müssen definiert werden.

3.2 Problemanalyse

Damit ein Verfahren entwickelt werden kann, muss als erstes geklärt werden, was genau unter einem Track verstanden wird und welche Anforderungen an den Algorithmus gestellt werden. Zur Veranschaulichung der verschiedenen Verläufe von Tracks ist eine Projektion der Hits auf die x-y-Ebene sinnvoll. Es wird der Querschnitt durch den Straw Tube Tracker betrachtet, wie er schematisch in der Abbildung 3.1 dargestellt ist. Der Querschnitt des gesamten Detektors lässt sich in sechs Sektoren unterteilen. In der Mitte befindet sich eine senkrechte Lücke, welche u. a. vom Target-System ausgefüllt wird. Für die zum Strahl parallelen Straw Tubes ergeben sich durch eine Projektion Kreise. Bei den gedrehten Straw Tubes kommt es bei der Projektion vor allem am Rand der Sektoren zu einer Reihe von Überlagerungen. Als Punkt für die Projektion kann der Punkt des Drahtes eines gedrehten Röhrchens, der sich in der Mitte des Zylinders befindet, genutzt werden. Bei den normalen Straw Tubes ist dieser Punkt in der Projektion für alle z gleich, da die Driftröhrchen entlang der z-Achse verlaufen.

Der transversale Anteil der Teilchenbahn wird aufgrund der Lorentz-Kraft zu einer Kreisbahn verformt. Zusammen mit der longitudinalen Komponente ergibt sich im Idealfall eine Helix, sodass sich in der Projektion Kreise ergeben würden. In der Realität wechselwirkt das Teilchen allerdings mit dem Detektormaterial, was zur Vielfachstreuung und zum Energieverlust der Teilchen führt. Durch den Energieverlust nimmt die Geschwindigkeit der Teilchen kontinuierlich ab, was zu einer spiralförmigen Flugbahnen mit einem immer kleiner werdenden Radius führt. Dieser Effekt ist impulsabhängig und für Teilchen mit kleinen Impulsen stärker. In der Projektion ergeben sich „Schneckenlinien“. Je nach Stärke der Ablenkung lassen sich stark gekrümmte oder auch fast gerade Flugbahnen erkennen. Es ist möglich, dass der STT nahezu komplette Teil-

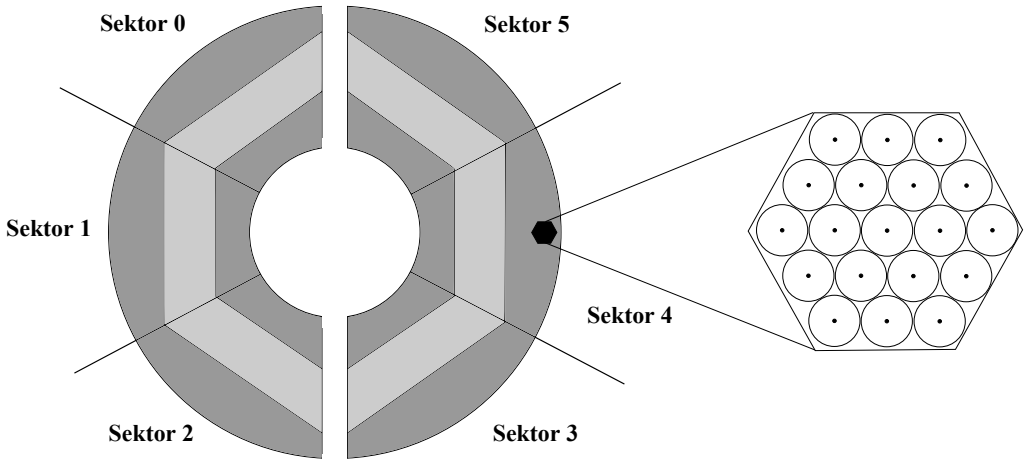

Abbildung 3.1: Schematischer Querschnitt durch den STT und seine Aufteilung in Sektoren. Der hell markierte Streifen beinhaltet die gedrehten Straw Tubes. Im dunklen Bereich befinden sich die parallelen Röhrchen, deren Querschnitt rechts skizziert ist (Umrandung und Mittelpunkte).

chenspuren erfasst oder auch nur ein Bruchstück dieser, sodass der restliche Weg von anderen Detektoren aufgezeichnet wird. Ist ab nun von einem Track die Rede, ist damit ein Teil einer Teilchenspur gemeint, der vom STT erfasst wurde. Bei der Suche nach Teilchenspuren im STT sind folgende Punkte zu beachten:

- Primäre Tracks beginnen im Regelfall in der innersten Reihe des STTs. Es gibt keine Tracks, die mitten im STT beginnen, da die Teilchen aus der Richtung des Stoßpunktes kommen. Dies gilt nicht für die Straw Tubes, die sich an der vertikalen Lücke befinden, weil es dort einen Bereich gibt, der nicht vom Detektor abgedeckt wird.

- Ein Track muss nicht zwingend von den inneren Reihen an Straw Tubes nach außen verlaufen. Er kann sich wieder zurück bewegen und dabei auch mehrere Sektoren durchqueren.

- Teilchen können so stark abgelenkt werden, dass sie im STT wiederholt mit einem immer kleiner werdenden Radius kreisen. Das hat zur Folge, dass ein großes Gebiet benachbarter Straw Tubes einen Hit meldet. Wird dieser Bereich dann noch von einer anderen Flugbahn gekreuzt, lässt sich der Verlauf der Tracks nur noch schwer nachvollziehen.

- Die Tracks können abrupt enden, wenn die Teilchen aus dem STT austreten. Somit sind auch sehr kurze Tracks und vereinzelte Hits möglich.

- Es besteht die Möglichkeit, dass einzelne Straw Tubes kein Signal geben, obwohl sie von einem Teilchen durchquert wurden. Dies kann aus einer zu schwachen Signalstärke resultieren und hat zur Folge, dass Tracks durch fehlende Hits unterbrochen sein können.

- Es ist denkbar, dass ein Teilchen zwischen zwei Straw Tubes hindurchfliegt ohne ein Signal bei einer der beiden auszulösen. Durch die Anordnung der Röhrchen ergibt sich aber, dass dann eine Straw Tube in der nächsten Reihe/Spalte ein Signal melden muss. Auch hier würde ein Hit zur Rekonstruktion des Tracks fehlen.

- Es ist sehr unwahrscheinlich, dass sich Tracks im Raum kreuzen und durch den gleichen Punkt verlaufen. Bei der Projektion in die x-y-Ebene ist allerdings zu beachten, dass Kreuzungen oft auftauchen können, da es zur Überlagerung kommt.

- Teilchen können in Sekundärteilchen zerfallen. Daher können Gabelungen (Sekundärvertices) auftauchen, sodass ein Track in zwei verzweigt.

Der Algorithmus soll alle normalen Tracks und möglichst viele Tracks mit den oben genannten Besonderheiten erkennen. Unter den normalen Tracks sind diejenigen zu verstehen, die sich vom Inneren des STTs nach außen bewegen und in der Projektion nicht von anderen Tracks gekreuzt werden. Auch unterbrochene Tracks sollen rekonstruiert werden können, sodass ein fehlender Hit nicht ausschlaggebend dafür ist, dass der Track nicht gefunden wird. Eine maximale Anzahl an fehlenden Hits, die auftauchen kann, ist dabei nicht gegeben. Allerdings werden mehrere direkt aufeinanderfolgende fehlende Signale als sehr unwahrscheinlich bewertet.

Das Verfahren muss schnell arbeiten und muss daher nicht zwingend alle Sonderfälle betrachten, wenn dies die Rechenzeit deutlich erhöht. Zeitintensives Rechnen muss vermieden werden. Aus diesem Grund ist eine Einbeziehung der Isochron-Radien der STT-Signale zu Ermittlung der Tracks nicht angebracht. Es müsste ein aufwendiger Fit durchgeführt werden, der anhand der Isochrone die bestmögliche Flugbahn berechnet.

Die Rekonstruktion von Flugbahnen in Form von „Schneckenlinien“ nimmt keinen hohen Stellenwert ein. Diese Tracks äußern sich in der x-y-Projektion durch eine große Menge benachbarter Straw Tubes, die alle ein Signal gegeben haben. Solch ein möglicher Track ist in Abbildung 3.2 skizziert. Eine schnelle Rekonstruktion derartiger Tracks wird als problematisch eingestuft. Zudem ist es auch schwierig, solche Flugbahnen gut durch eine Funktion zu approximieren. Zusammengefasst ergeben sich folgende Anforderungen:

- Der Algorithmus soll die Hits zu Tracks gruppieren. Dabei steht die Rekonstruktion normaler Tracks im Vordergrund.

- Es ist wichtiger alle physikalischen Tracks zu finden, als die Zahl falsch gebildeter Tracks zu minimieren.

- Fehlende Signale sind zu überbrücken.

- Das Verfahren muss schnell arbeiten, da es in der Zukunft als Online-Trackfinder eingesetzt werden soll.

- Der Algorithmus sollte gut parallelisierbar sein.

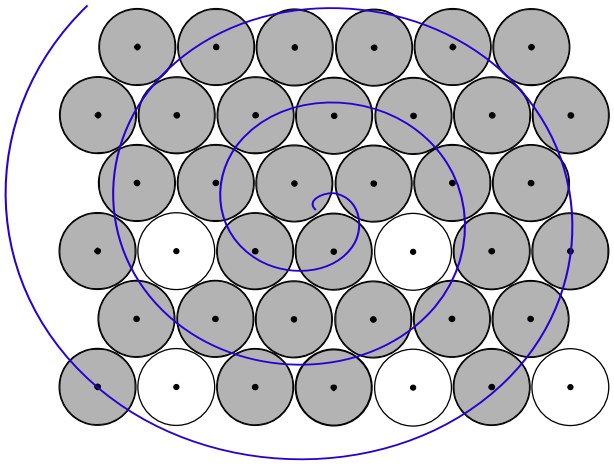

Abbildung 3.2: Veranschaulichung der Flugbahn eines stark abgelenkten Teilchens (blau) durch die Straw Tubes in der x-y-Projektion. Alle grau-ausgefüllten Straw Tubes melden ein Signal. Es entsteht eine große zusammenhängende Menge benachbarter Straw Tubes, die das Teilchen passiert. Eine eindeutige Rekonstruktion ist hier nicht möglich.

Zur Problemlösung ist ein Trackfinding-Algorithmus zu entwickeln, der diesen Anforderungen genügt und in PandaRoot integriert werden kann. Dazu müssen alle Hit-Informationen der Straw Tubes eingelesen werden. Anschließend muss ermittelt werden, welche Hits zu einem Track gehören. Die Informationen über die Tracks müssen abgespeichert und verifiziert werden. Daher soll sich eine Analyse des Verfahrens anschließen. In dieser ist z. B. zu ermitteln, wie viele der physikalischen Tracks gefunden werden, ob diese vollständig gefunden worden sind oder ob auch Tracks gefunden werden, die eigentlich nicht vorhanden sind. Weitere solcher Bewertungs-Kriterien sollen aufgestellt werden und sind zu überprüfen.

3.3 Das Framework PandaRoot

FAIR und somit auch $\overline{\text{P}}$ANDA befinden sich noch in der Entwicklung. Um das Detektorsystem zu verbessern und geeignete Verfahren zur Auswertung der Daten zu entwerfen, wird das $\overline{\text{P}}$ANDA-Experiment derzeit mit einem Programm namens *PandaRoot* simuliert. PandaRoot ist ein Framework, das speziell zur Simulation und Analyse von $\overline{\text{P}}$ANDA-Experimenten dient und auf den Programmiergerüsten ROOT und FairRoot basiert. ROOT stellt das Basis-Framework dar, während FairRoot und PandaRoot gesonderte Funktionen für die FAIR- bzw. $\overline{\text{P}}$ANDA-Komponenten anbieten. Da keine besonderen Informationen über FairRoot zum Verständnis des Simulationsprozesses benötigt werden, werden in diesem Abschnitt nur die Frameworks ROOT und PandaRoot eingeführt. Zudem wird genauer auf das Trackfinding in PandaRoot eingegangen, weil das zu entwickelnde Verfahren an dieser Stelle ergänzt werden wird.

3.3.1 ROOT

ROOT ist ein objektorientiertes Framework, welches an CERN (Europäische Organisation für Kernforschung) entwickelt wurde. Es bietet Physikern die Möglichkeit, Experimente aus dem Bereich der Hochenergiephysik zu simulieren und die Daten auszuwerten. Mit ROOT werden Dateien und Objekte in einer eigenen Datenstruktur (ROOT-Dateien) gespeichert. Diese ist an eine Baumstruktur angelehnt und zeichnet sich durch einen sehr schnellen Zugriff auf riesige Datenmengen aus. Mit Hilfe vieler mathematischer und statistischer Methoden können die Daten verarbeitet werden. Das Framework bietet u. a. Tools zum Erstellen von Histogrammen, Streudiagrammen und zum Zeichnen von Funktionen. Auf diese Weise können die Daten ausgewertet und die Ergebnisse der Simulationen dokumentiert werden. [15]

ROOT wurde mit C++ implementiert und stellt einen speziellen Interpreter namens CINT zur Verfügung. Der Einsatz eines Interpreters erlaubt eine schnelle Entwicklung/Änderung der Simulations- und Analyseprozesse, da die Zeit zum Kompilieren und Linken reduziert wird. Die Programme, mit denen diese Prozesse gesteuert werden, werden als Makros bezeichnet, weil sie nur mit CINT interpretiert werden müssen. ROOT lässt sich mit einem Command Line Interface ansteuern. Neben diesem interaktiven Betrieb wird für die Simulation und Analyse von Experimenten in der Regel der Batchbetrieb vorgezogen, in dem die erwähnten Makros den steuernden Input darstellen.

Im Folgenden soll kurz erläutert werden, wie die Daten für die Simulation erzeugt werden. Um das Verhalten eines Systems zu simulieren, können analytische oder stochastische Mittel eingesetzt werden. Erstere führen zu komplexen Bewegungs-Gleichungen, die gelöst werden müssen. Dies findet z. B. bei der Berechnung des Bahnverlaufs eines geladenen Teilchens im magneti-

schen Feld Anwendung. Um die Wechselwirkung von Teilchen mit Materie zu simulieren, wird der stochastische Ansatz gewählt. Es gibt eine große Anzahl möglicher physikalischer Vorgänge, die ablaufen können. Aus diesem Grund wird so ein Prozess mit Pseudozufallszahlen simuliert. Ein derartiges Vorgehen zählt zu den Monte-Carlo-Methoden. ROOT stellt eine Schnittstelle (Virtual Monte Carlo) zur Verfügung, mit der Monte-Carlo-Programme wie Geant3, Geant4 und Fluka genutzt werden können. Die Monte-Carlo-Simulation erzeugt u. a. für jedes Teilchen einen Weg durch die Materie/Detektorkomponenten und passt die Eigenschaften des Teilchens (z. B. Energie) an. Dabei werden Schritt für Schritt probabilistisch Wechselwirkungen und physikalische Vorgänge miteinbezogen. Nach dieser Stufe schließt sich die Simulation der Auslese-Elektronik an, die sogenannte Digitalisierung. [16]

3.3.2 PandaRoot

Im Rahmen des FAIR-Projektes wurde das Framework *FairRoot* entwickelt, um ein einheitliches Programmiergerüst für die Experimente von FAIR zu schaffen. Als Teil dessen ist *PandaRoot* entstanden, das auf ROOT basiert und Geant4 benutzt.

Der Simulationsprozess

Die Simulationen mit PandaRoot lassen sich nach [12] in drei grobe Schritte einteilen: das Generieren eines Events, das Übertragen dieses Events auf den Detektor und die Digitalisierung und Analyse der Daten. Allerdings bietet es sich an, die Digitalisierung und Analyse als separate Schritte aufzufassen. Sie werden in der Regel auch in verschiedenen Makros definiert. Daher ist nachstehend von vier Teilschritten die Rede, deren Ablauf in der Abbildung 3.3 nachvollzogen werden kann.[1]

Unter einem Event wird ein Ereignis des $\overline{\text{P}}$ANDA-Experimentes verstanden. Dazu zählt das Wechselwirken eines Teilchens des Strahls mit dem Teilchen des Targets und die daraus resultierenden Vorgänge und Sekundärteilchen. Mit Hilfe von *Event-Generatoren* wird demnach der zu betrachtende physikalische Vorgang definiert bzw. simuliert. Die Event-Generatoren erlauben die Angabe der Teilchen, die miteinander wechselwirken und die Benennung von Sekundärteilchen, die dabei entstehen. Es können die Eigenschaften der Teilchen und auch Zerfallsketten definiert werden. Die Event-Generatoren erzeugen eine Ausgabedatei, die Informationen über die entstehenden Teilchen nach der Wechselwirkung enthält (Energie, Impuls, ...).

Diese Datei dient dann als Eingabe für das sogenannte *Transport-Modell*. Dieses überträgt die Informationen über die Teilchen auf die Detektorgeome-

[1] Die Informationen zu den Simulationsschritten wurden trotzdem sinngemäß [12] entnommen.

trie und benutzt eines der in 3.3.1 beschriebenen Monte-Carlo-Programme. Die Flugbahnen der Teilchen werden berechnet und es wird ermittelt, an welchen Punkten die Bestandteile der einzelnen Detektorkomponenten passiert werden. Dabei werden Wechselwirkungen wie z. B. mit dem magnetischen Feldern oder den Detektormaterialien und auch Teilchenzerfälle berücksichtigt. Dieser Schritt setzt eine detaillierte Definition des gesamten $\overline{\text{P}}$ANDA-Detektors voraus. Die dafür benötigten Klassen sind bereits implementiert. Die vom Monte-Carlo-Programm erzeugten Daten werden in einer ROOT-Datei abgespeichert.

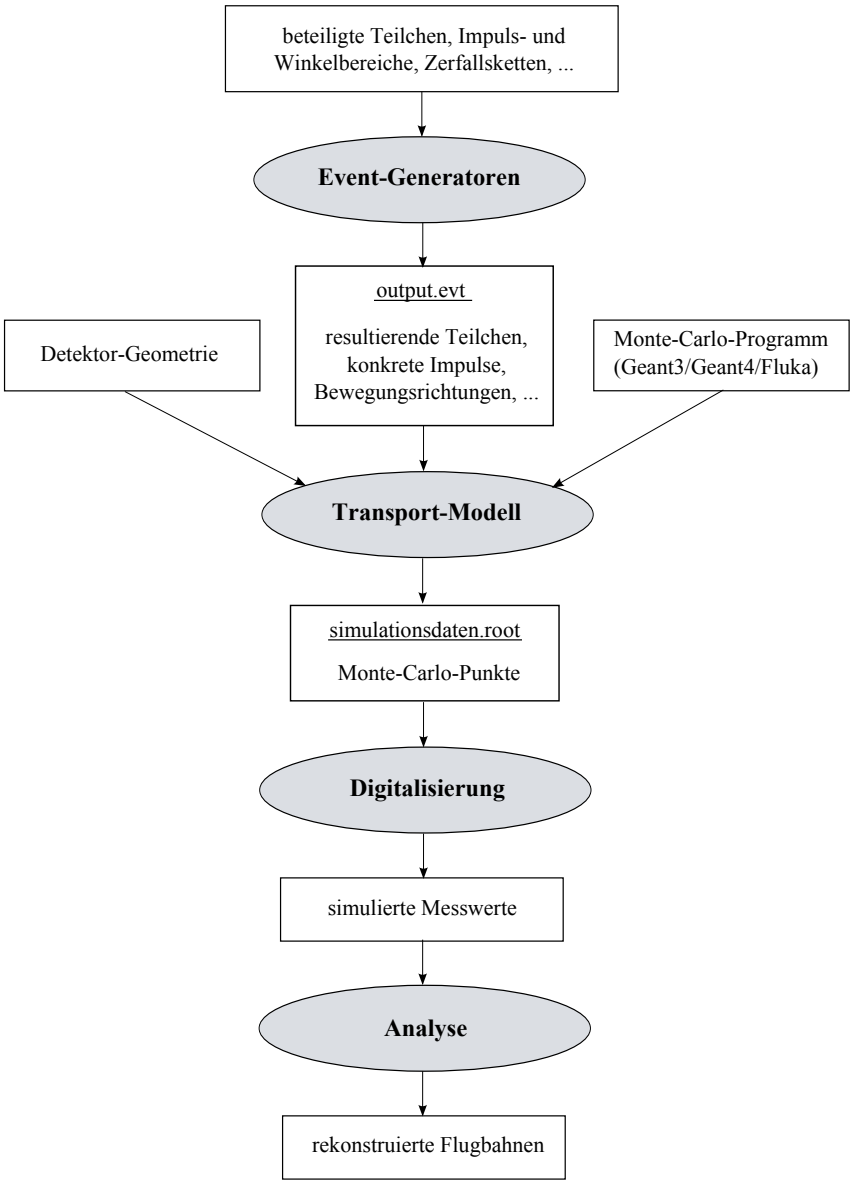

Abbildung 3.3: Schematische Darstellung der Simulationsschritte von PandaRoot (Abarbeitungsrichtung von oben nach unten).

Durch das Ausführen sogenannter *Tasks* können die Daten digitalisiert und die Teilchenspuren rekonstruiert werden. Die Digitalisierung erstellt aus den berechneten Monte-Carlo-Punkten die Signale, die von den Detektoren wirklich ausgelöst werden würden. So kann es zu Verzögerungen und Messungenauigkeiten kommen. Es kann sogar vorkommen, dass manche Punkte, die berechnet worden sind, gar nicht in ein Signal umgewandelt werden, da z. B. ein Schwellenwert nicht erreicht wurde. Diese Maßnahmen simulieren analoge Daten zu den Messdaten des realen Experiments.

Im Anschluss werden die Messwerte analysiert. Es werden Algorithmen zur Rekonstruktion der Flugbahn der Teilchen angewendet. Außerdem muss ermittelt werden, welche Teilchen an dem Event beteiligt waren. Aufgrund der Event-Simulation kann die Qualität der eingesetzten Methoden zur Rekonstruktion und Identifikation bewertet werden, da die ursprünglichen Informationen über die Teilchen und Flugbahnen bekannt sind. Nach Anwendung geeigneter Verfahren liegen als Ergebnis genaue Informationen über die Flugbahnen und den zugehörigen Teilchen vor.

Die Anzahl der auszuführenden Events, die Auswahl des Monte-Carlo-Programms und andere Einstellungen können durch das Ändern von Flags oder vereinzelten Zeilen in den Makros nach Belieben angepasst werden ohne das Programm erneut kompilieren zu müssen.

Die Event-Rekonstruktion in PandaRoot

Die Entwicklung von Verfahren zur Rekonstruktion der Teilchenspuren ist einer der Schwerpunkte der $\overline{\text{P}}$ANDA-Kollaboration. Es gibt schon eine Vielzahl von Algorithmen. Jedoch werden diese stets weiterentwickelt oder neue Ansätze erprobt.

Um das Stoßexperiment in der Zukunft anhand von realen Detektorsignalen nachbilden zu können, müssen Algorithmen entwickelt werden, die feststellen, welche Hits zu einer Teilchenspur gehören. In PandaRoot werden ausgehend von den Monte-Carlo-Punkten Detektorsignale (Hits) erzeugt, die als Eingabewerte für die Event-Rekonstruktion dienen. Diese riesige Menge an Hits muss aufgetrennt und zu einzelnen Tracks gegliedert werden. Das sogenannte *Trackfinding* sortiert die einzelnen Hits zu Gruppen, die von der gleichen Teilchenspur stammen könnten (Track-Kandidaten). In Bezug auf den STT ergibt sich eine Menge von STT-Hits der entsprechenden Straw Tubes, die das Teilchen passiert hat.

Zur genauen Beschreibung der Flugbahnen muss die Bahn gefunden werden, die alle Hits am besten approximiert. Dabei müssen aufgetretene Messungenauigkeiten in die Berechnung einbezogen werden. Hierbei handelt es sich um das *Trackfitting*, welches u. a. mit dem Kalman-Filter durchgeführt wird. Das Kalman-Filter ist ein rekursives Verfahren, das es ermöglicht aus den Messwerten den wahren Zustand des Systems zu schätzen. Außerdem ermöglicht es

eine Vorhersage des weiteren Verlaufs einer Flugbahn. Dies ist vor allem beim Zusammenführen der Track-Informationen verschiedener Detektoren hilfreich, da diese Daten abschließend noch zu einem *globalen Track* zusammengeführt werden müssen.

Um eigenständig in den Prozess des Trackfindings bzw. Trackfittings einzugreifen, müssen Tasks definiert werden. Bei der Ausführung einer Task kann auf die gewünschten Datenstrukturen zugegriffen werden. In diesem konkreten Fall werden die STT-Hits benötigt. Diese Daten können verarbeitet werden und zu neuen Datenstrukturen führen (in diesem Fall Tracks). Die so ermittelten Informationen werden nach Beendigung der Task in einer ROOT-Datei abgespeichert. Die Ausführung der Tasks ist dann im entsprechenden Makro zu ergänzen und kann so in die Simulation miteinbezogen werden.

Visualisierung mit Eve

Eve steht für *Event Visualization Environment* und ist ein Framework von ROOT, mit dem die Events dargestellt werden können. Es bietet u. a. Klassen zur Darstellung von Baumstrukturen, von Listen und der verschiedenen Geometrien für die Detektoren. Die Flugbahnen der Teilchen können in einer 3D-Ansicht aus verschiedenen Blickwinkeln betrachtet werden. Für die 3D-Ansicht werden automatisch 2D-Projektionen erzeugt. Mit Eve können die Monte-Carlo-Punkte, Hits und Teilchenspuren für die verschiedenen Detektoren dargestellt werden. [17]

Zur Überprüfung der Module und der Güte des zu entwickelnden Verfahrens werden die Ergebnisse mit Eve dargestellt. In Abbildung 3.4 ist ein Event im STT zu sehen, wie es mit Eve visualisiert wird. Es handelt sich um eine Projektion auf die x-y-Ebene. Andere Detektoren und deren Signale wurden der Übersichtlichkeit halber ausgeblendet. In der Mitte der Darstellung befindet sich der Stoßpunkt. In diesem Fall gehen alle Flugbahnen der Teilchen vom Zentrum aus. Sie sind durch die grauen Linien gekennzeichnet. Die Umrandung des STTs ist grün gekennzeichnet. Sie ist von vielen kleinen Punkten ausgefüllt, die die Mittelpunkte der Straw Tubes darstellen. An den Enden der Reihen mit gedrehten Straw Tubes innerhalb der Sektoren kommt es zur Überlagerung dieser Punkte. Durch die schräge Anordnung der Röhrchen müssen die entstehenden Lücken am Rand mit kürzeren Straw Tubes aufgefüllt werden, sodass sich mehrere Mittelpunkte in der Projektion dicht beieinander befinden. Bei den pinkfarbenen Punkten handelt es sich um die Monte-Carlo-Punkte des STTs. Diese liegen demzufolge auf einer Linie. Die hervorgehobenen (hellen) Punkte stellen die zugehörigen STT-Hits dar. Da das kleinste Pixel des STTs eine Straw Tube ist, liegen die Hits nicht mehr auf einer glatten Linie. Zudem liegen die Signale gedrehter Straw Tubes weiter von den Teilchenspuren entfernt, was an der Darstellung in der x-y-Projektion liegt.

Bei dem ersten Beispiel handelt es sich um ein sehr übersichtliches Event. Die

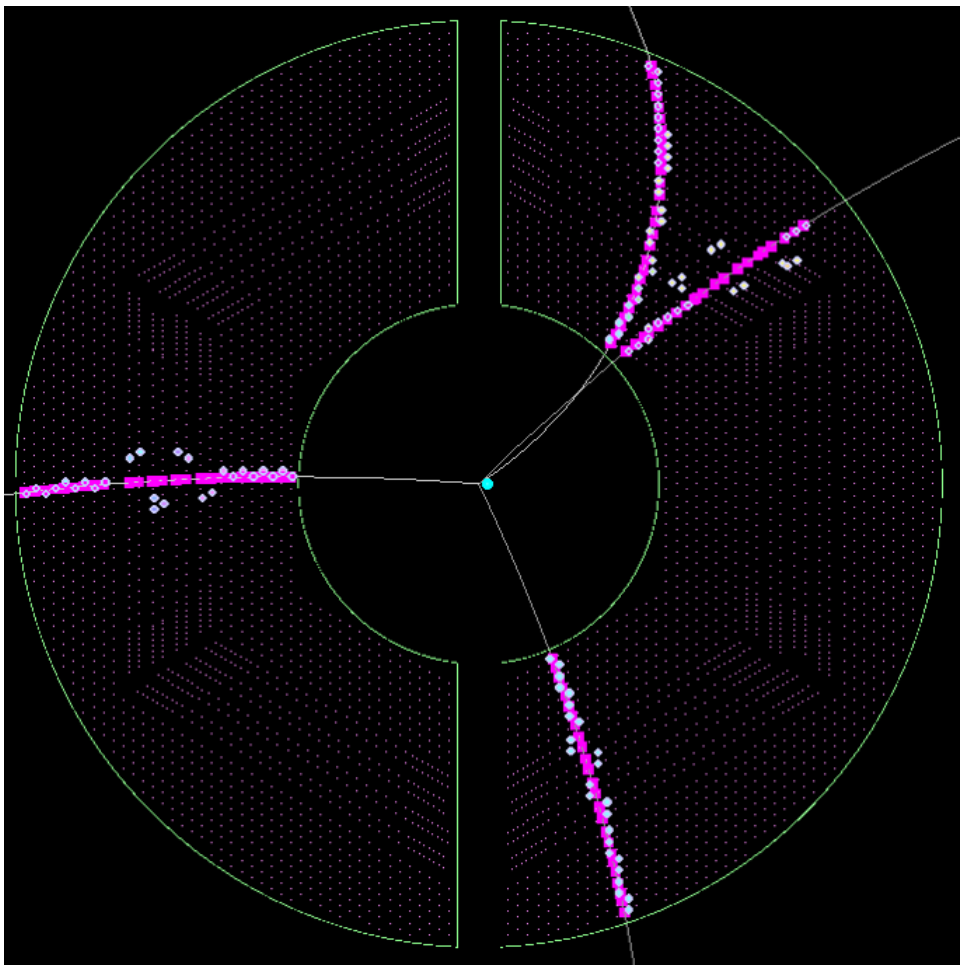

Abbildung 3.4: Ausschnitt der Darstellung eines Events im STT in der x-y-Projektion mit Eve. Die Signale zu einer Flugbahn können abrupt enden, wenn das Teilchen den STT in z-Richtung verlässt.

Signale des STTs können durch eine reine Betrachtung gut zu verschiedenen Flugbahnen gruppiert werden. Um die Problematik der Track-Rekonstruktion zu verdeutlichen, ist in Abbildung 3.5 ein weiteres Event dargestellt. Nicht alle Teilchen-Flugbahnen lassen sich leicht durch eine reine Betrachtung der Signale rekonstruieren.

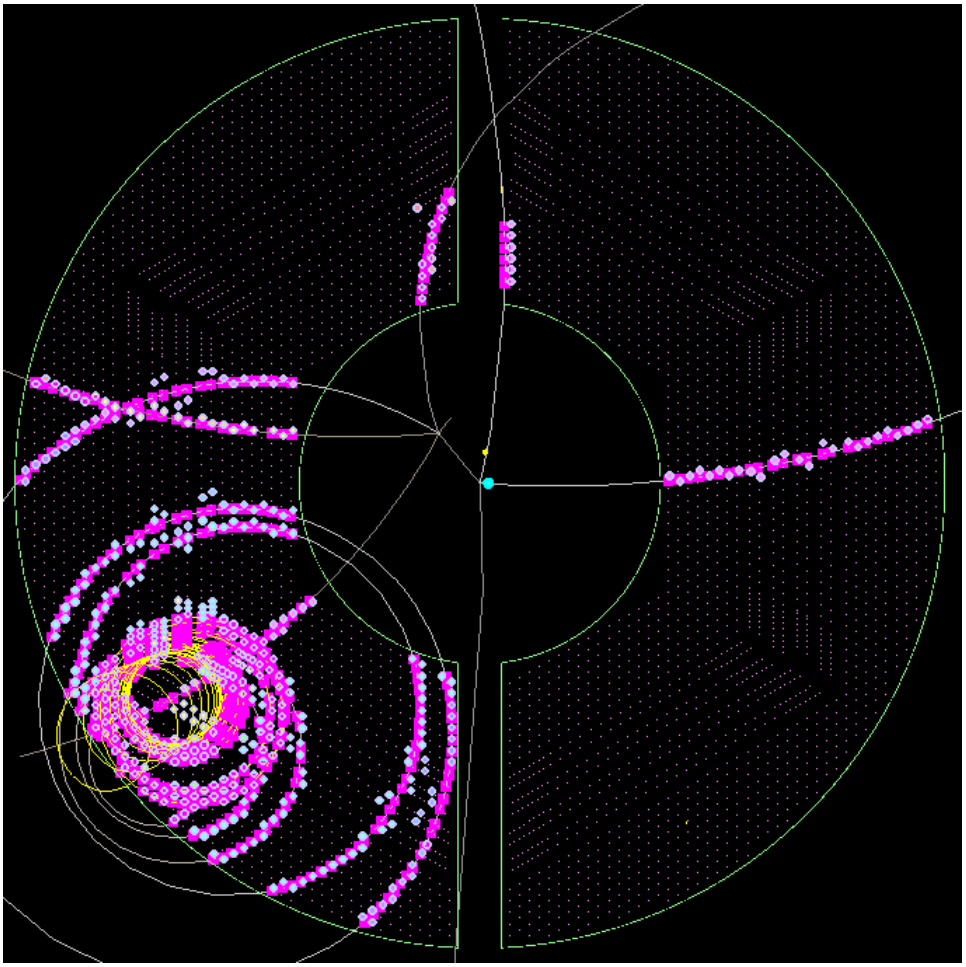

Abbildung 3.5: Darstellung eines komplexen Events im STT. Es gibt mehrere kreisende Flugbahnen, die einander überkreuzen. Ein freigesetztes Elektron (gelbe Flugbahn) erzeugt eine große und dichte Menge an STT-Hits.

4 Verfahrensbeschreibung zum Finden der Teilchenspuren

In diesem Kapitel wird das Verfahren beschrieben, das zum Finden von Teilchenspuren im STT entwickelt wurde. Vorab werden „Zelluläre Automaten“ eingeführt, deren Funktionsweise auf die Problemstellung übertragen wird. Außerdem ist das Berechnen einer Kreisfunktion zu einer gegebenen Menge von Punkten ein wichtiger Aspekt, weshalb darauf auch gesondert eingegangen wird. Darauf aufbauend wird der Trackfinding-Algorithmus detailliert beschrieben. Die genaue Implementierung und die verwendeten Datenstrukturen werden erst in Kapitel 5 erläutert.

4.1 Zellulärer Automat

Ein zellulärer/zellularer Automat ist ein abstraktes, diskretes Modell, das vor allem zur Modellierung von dynamischen Systemen der Natur dient. Ein Zellularautomat besteht aus einer Menge von Zellen, deren Zustände sich in einer definierten Nachbarschaft nach bestimmten Übergangsregeln zeitlich getaktet verändern. In der Physik, Biologie, Chemie und anderen Bereichen treten selbst-organisierende Systeme auf, bei denen durch Interaktion der Komponenten aus einem ungeordneten System geordnete Zustände entstehen. Ein Beispiel dafür ist die symmetrische Struktur von Schneeflocken, die durch die eigenständige Ausrichtung der Moleküle zu Stande kommt. Das Ziel zellulärer Automaten ist es, solche Verhalten zu simulieren und zu untersuchen. [20]

4.1.1 Eigenschaften und einführendes Beispiel

Der zelluläre Automat wird fortan als „CA“ für „Cellular automaton“ bezeichnet. Nach [10] zeichnet er sich durch diese Eigenschaften aus:

- Er betrifft eine abzählbare Menge von Zellen, die miteinander in Beziehung stehen (*Zellraum*). Dabei handelt es sich oft um ein n-dimensionales Gitter. Für die Dimension 2 kann dies z. B. eine Art Schachbrett sein, bei dem die Kästchen (Zellen) neben- und übereinander miteinander verbunden sind.

- Es ist eine diskrete Menge von *Zuständen* definiert, die eine Zelle des CA annehmen kann (Status). Aus der Gesamtheit der einzelnen Zustände der Zellen ergibt sich der Zustand des CA.
- Die Definition der *Nachbarschaften* ist gegeben. Für jede Zelle sind topologische Nachbar-Zellen definiert, die die Änderung des Zustands beeinflussen.
- Die Definition von *Randbedingungen* kann gegeben sein. Der Zellraum ist begrenzt und weist Rand-Zellen auf, die gesondert behandelt werden müssen. Sie haben eine andere Nachbarschafts-Umgebung als die restlichen Zellen. Als Abhilfe dienen oft periodische Randbedingungen, die Rand-Zellen miteinander verketten. Im eindimensionalen CA wird dies durch die Bildung eines geschlossenen Rings erreicht. Auf diese Weise lässt sich ein unendlicher Zellraum simulieren.
- Klar definierte *Regeln* beschreiben den Übergang des aktuellen Zustands einer Zelle in den neuen ausschließlich basierend auf dem aktuellen Zustand der Zelle und den Zuständen der Nachbar-Zellen.
- Die Zustände der Zellen verändern sich *simultan*, d. h. sie werden gleichzeitig von allen Zellen angepasst. Dieser Aktualisierungsvorgang wird als ein Iterationsschritt aufgefasst.

Ein großes Interesse an zellulären Automaten entstand nach der Entwicklung des „Game of Life“ von John Conway im Jahr 1970.[1] Bei dem „Game of Life“ handelt es sich um ein Spiel, das Analogien zum Aufkommen, Rückgang und der Veränderung einer Gesellschaft lebender Organismen aufweist. Als Zellraum dient ein zweidimensionales Feld mit einem Schachbrettmuster. Gestartet wird mit einer einfachen Verteilung der lebenden Zelle in diesem Feld. Jede Zelle hat acht Nachbarn (jeweils vier orthogonale und diagonale). Cornway hat folgende Übergangsregeln definiert:

1. Überleben: Eine lebende Zelle mit zwei oder drei lebenden Nachbarn bleibt am Leben.
2. Tod: Eine lebende Zelle mit vier oder mehr lebenden Nachbarn stirbt aufgrund von Überbevölkerung. Lebende Zellen mit weniger als zwei lebenden Nachbarn sterben an Einsamkeit.
3. Geburt: Eine tote Zelle mit genau drei lebenden Nachbarn wird neu geboren.

[1] Die Informationen zu diesem CA sind auf [7] zurückzuführen.

Durch Anwenden der Regeln wird nach sehr vielen Iterationsschritten einer dieser drei Endzustände angenommen: die Population klingt komplett ab, es entwickelt sich eine stabile Konstellation, die sich nicht mehr verändert oder es ergibt sich ein endloser Kreislauf in dem sich die Verteilungen wiederholen.

4.1.2 Übertragung auf den STT

Für den STT soll ein Verfahren entwickelt werden, das an zelluläre Automaten angelehnt ist und das Trackfinding unterstützt. Dies wird durch die Anforderung der Parallelisierbarkeit an das Verfahren und die vorhandenen Nachbarschaftsbeziehungen der Straw Tubes motiviert. In diesem Abschnitt wird erklärt, wie erste „Tracklets“ als Wegstücke eines physikalischen Tracks erzeugt werden.

Zellen und ihre Nachbarschaften

Die Gesamtheit aller Straw Tubes des STTs bildet die Grundmenge für den Zellraum. Dementsprechend handelt es sich bei einer Zelle um eine Straw Tube. Das Gitter, das die Nachbarschaftsbeziehungen repräsentiert, ist zwei- bzw. drei-dimensional. Für die parallelen Straw Tubes, die nicht an gedrehte Driftröhrchen grenzen, lassen sich die *Nachbarn* im zweidimensionalen Raum betrachten. Jede dieser Straw Tubes ist von maximal sechs anderen umgeben. Am Rand des STTs schrumpft die Anzahl der Nachbarn bis auf zwei. Ein Auszug dieses Gitters ist in Abbildung 4.1 zu sehen. Für die gedrehten Driftröhrchen und die Straw Tubes, die an solche grenzen, ergeben sich weitaus mehr Nachbarn. Durch die versetzte Anordnung der gedrehten Röhrchen überqueren diese eine große Anzahl von anderen Straw Tubes. So kann es sein, dass eine Straw Tube alle Straw Tubes der nächsten Reihe berührt. Da die gedrehten Röhrchen teilweise hintereinander angeordnet sind, lassen sich diese Beziehungen nicht mehr zweidimensional darstellen.

Es werden keine besonderen Randbedingungen definiert. Die Straw Tubes am Rand besitzen weniger Nachbarn, die betrachtet werden müssen. Es ist keine Ausdehnung des Zellraums z. B. durch Verkettung der Rand-Zellen erwünscht.

Zustandsdefinition

Der Zustand einer Zelle des CA ist eine Track-ID in Form einer ganzen Zahl, die den Track identifiziert zu dem die Zelle gehört. Vor der Anwendung des CA wird jeder Zelle eine eindeutige Track-ID zugewiesen. Nach der Anwendung des CA gehören die Zellen, die dieselbe Track-ID besitzen, zum gleichen physikalischen Track.

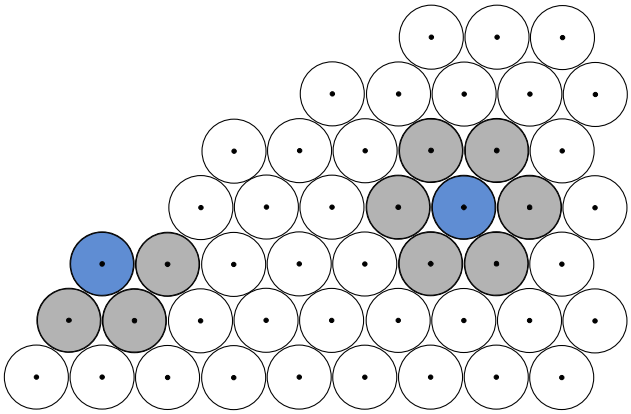

Abbildung 4.1: Darstellung der Nachbarschaftsbeziehungen der parallelen Straw Tubes im STT. Die betrachteten Zellen sind blau und die angrenzenden Nachbarn grau markiert.

Einschränkung des Zellraums

Straw Tubes, die keinen Hit gemeldet haben, können zu keinem Track gehören und benötigen daher keine Track-ID. Selbst wenn ein Teilchen eine Straw Tube passiert ohne ein messbares Signal auszulösen, kann die Straw Tube keine Informationen zu dem zu rekonstruierenden Track beitragen. Aus diesem Grund müssen vom CA nur Straw Tubes betrachtet werden, die ein Signal gemeldet haben. Diese werden als aktive Zellen bezeichnet.

Übergangsregeln

Die Grundidee zur Anpassung der Track-ID besteht darin, die Zahl solange mit dem Minimum aus der eigenen und den benachbarten Track-IDs zu ersetzen, bis sich im folgenden Iterationsschritt der Zustand keiner Zelle mehr ändert. Die Ermittlung und Anpassung der Track-IDs wird wie folgt vorgenommen:

1. Jede Zelle besitzt zu Beginn eine eindeutige Track-ID, welche sonst keiner anderen Zelle zugewiesen wurde.
2. Es werden die Track-IDs der benachbarten aktiven Zellen ermittelt.
3. Es wird das Minimum dieser betrachteten Track-IDs (inklusive der eigenen) ermittelt.
4. Im Iterationsschritt wird die Track-ID auf das Minimum gesetzt. Folglich ändert sich der Zustand nur, wenn es eine benachbarte aktive Zelle mit einer kleineren Track-ID gibt.

Mit diesem Verfahren werden alle aktiven Zellen, die benachbart sind, mit der gleichen Track-ID versehen. Dies hat zur Folge, dass auch Tracks, die sich in

der Projektion kreuzen zu einem gemeinsamen Track zusammengefügt werden und nicht voneinander unterschieden werden können. Um sicher zu gehen, dass Zellen mit der gleichen Track-ID nur zu einem Track gehören, müssen Ambiguitäten wie Kreuzungen oder Verzweigungen ausgeschlossen werden. Da nur reine Tracklets erwünscht sind, die ausschließlich Hits von ein und demselben physikalischen Track enthalten, müssen Ambiguitäten gesondert behandelt werden. Eine Zelle kann nicht eindeutig einem Track zugeordnet werden, wenn sie mehr als zwei aktive Nachbarn hat. Es kann sich hierbei um einen Kreuzungs- oder Verzweigungspunkt handeln. Die Zelle kann auch Teil eines Tracks in Form eines „Wirbels“ sein, der einen größeren Bereich benachbarter Straw Tubes passiert. Derartige Zellen werden von der Minimums-Findung ausgeschlossen, da es keine eindeutige Zuordnungsmöglichkeit zu einem Track gibt.

Hat eine Zelle genau zwei aktive Nachbarn, wird angenommen, dass es sich um ein Verbindungsstück innerhalb eines Tracks handelt. Gibt es nur eine aktive Nachbarzelle, handelt es sich um einen Endpunkt eines Tracks bzw. eines Teilstück eines Tracks. Diese Zellen können eindeutig einem Track/Tracklet zugewiesen werden.

Für den CA ergibt sich schließlich nur eine Regel:

> Hat eine Zelle einen oder zwei aktive Nachbarn, ermittele das Minimum aus den Track-IDs *gleichartiger* Nachbarn und der eigenen. Setze die Track-ID der Zelle auf dieses Minimum.

Unter *gleichartigen* Nachbar-Zellen sind diejenigen zu verstehen, die auch nur ein oder zwei aktive Nachbarn besitzen. Hier kommt ein weiterer Filter zum Einsatz, der Zellen mit Ambiguitäten ausblendet. Die Menge der Zellen, die von der Änderung des Zustands betroffen ist, wurde damit weiter eingeschränkt, was sich positiv auf die Rechenzeit des Verfahrens auswirkt.

Die Zustände der aktiven Zellen werden basierend auf der oben genannten Regel simultan angepasst. Ändert sich in zwei aufeinanderfolgenden Iterationsschritten kein Zustand mehr, wurde die Track-ID aller eindeutig zuweisbaren Zellen endgültig gesetzt. Dies erlaubt die Gruppierung der Zellen bzw. STT-Hits mit gleicher Track-ID zu einem Tracklet. Mit dieser Vorgehensweise lassen sich sehr schnell die einzelnen Zellen anhand von Nachbarschaftsbeziehungen zu Tracklets oder sogar zu vollständigen Tracks zusammenführen. Vollständige Tracks werden nur dann rekonstruiert, wenn alle Zellen des Tracks keine Ambiguitäten aufweisen.

Beispiel

Die nachfolgend Abbildung 4.2 veranschaulicht, wie der Zellraum gefiltert wird und welches Ergebnis der CA liefert. Es ist zu sehen, welcher Zellraum betrachtet wird, wenn die Hits der beiden skizzierten Tracks als Eingabewerte dienen.

Alle inaktiven Zellen (ohne ein Signal) werden ausgeblendet. Die Zustände der aktiven Zellen werden mit eindeutigen Zahlen versehen (Track-ID). Diese wurden beliebig vergeben. Es schließt sich das Herausfiltern von Ambiguitäten an. Die Zustände werden nur für Zellen geändert, die ein oder zwei aktive Nachbarn haben. Alle anderen Zellen werden nicht betrachtet. Die Übergangsregel wird angewendet bis sich im dritten Iterationsschritt keine Track-ID mehr ändert, was zum Abbruch des Verfahrens führt. Es liegen vier Tracklets vor, die sich aus jeweils drei Zellen bzw. Hits zusammensetzen.

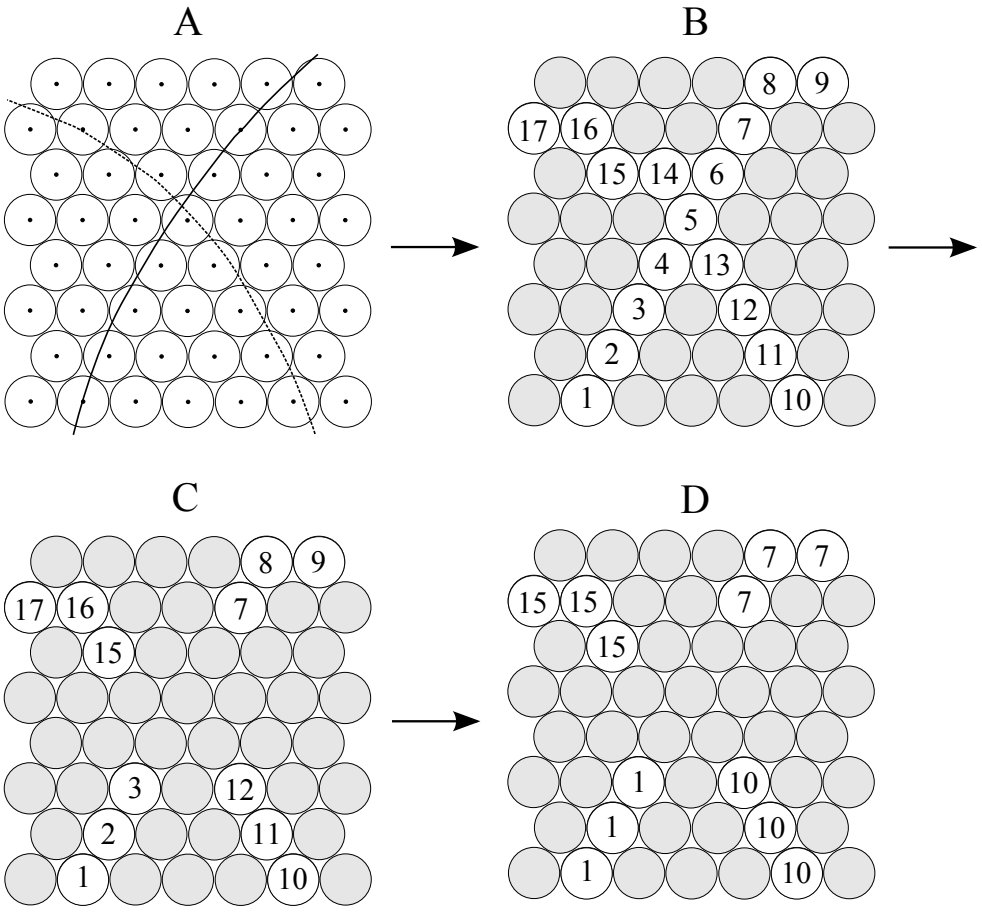

Abbildung 4.2: A: zwei sich kreuzende Tracks im Querschnitt des STTs, B: Ausblenden aller inaktiven Zellen des CA und Vergabe der Start-Track-IDs, C: Ausblenden von Ambiguitäten, D: End-Track-IDs nach Anwendung des CA

4.1.3 Beurteilung des Prinzips

Das im vorigen Abschnitt beschriebene Verfahren erlaubt es einfach und schnell STT-Hits in Tracklets zu gliedern. Allerdings werden nur Zellen in Betracht

gezogen, die ein oder zwei aktive Nachbar-Zellen besitzen. Nur diese Zellen können durch die Anwendung des CA zu reinen Tracklets gruppiert werden. Ein Tracklet wird als „rein“ verstanden, wenn es sich ausschließlich aus Hits zusammensetzt, die zu ein und demselben physikalischen Track gehören. Überschneiden sich also zwei Tracks, wird der Kreuzungsbereich ausgeblendet und die ursprünglichen Tracks zerfallen in mehrere Tracklets. Vereinzelte Hits, aktive Zellen ohne aktive Nachbarn, werden von dem CA ebenfalls nicht angetastet. Zur vollständigen Rekonstruktion von Tracks müssen sich weitere Schritte anschließen, in denen die Tracklets und nicht betrachteten Hits zusammengeführt werden.

Der CA bietet keine Möglichkeit, anhand von Nachbarschaftsbeziehungen genauere Aussagen über die Art der Zellen zu machen, die mehr als zwei aktive Nachbarn besitzen. Hat eine Zelle drei aktive Nachbarn, kann es sich um eine Zelle vor einer Kreuzung, in einer Verzweigung oder in der Nähe von zwei beieinander liegenden Tracks handeln, wie es in Abbildung 4.3 skizziert ist. Hat eine aktive Zelle vier aktive Nachbarn, liegt die Vermutung nahe, dass es sich um einen Kreuzungspunkt handelt. Aber auch hier lässt sich dies nicht mit Sicherheit sagen, wie das dritte Beispiel zeigt. Daher sind keine weiteren allgemeingültigen Aussagen möglich, die das Trackfinding vervollständigen.

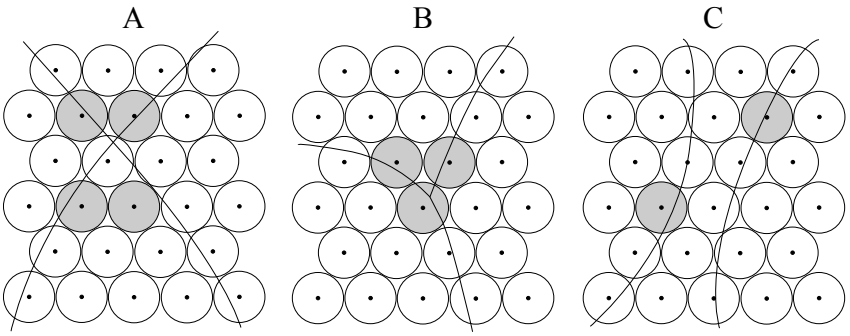

Abbildung 4.3: A: zwei sich kreuzende Tracks, B: Verzweigung eines Tracks, C: zwei aneinander vorbeilaufende Tracks. Zellen mit drei aktiven Nachbarn sind grau markiert.

Die Rekonstruktion von Tracks ist ein sehr zeitintensiver Schritt bei der Analyse der Simulationsdaten. Der Einsatz eines CA hat den Vorteil, dass das Evaluieren der Zustände der einzelnen Zellen parallelisierbar ist. Es muss nur sichergestellt werden, dass das Aktualisieren zeitgleich erfolgt und alle Zellen mit den Daten des aktuellen Iterationsschrittes arbeiten. Zur Änderung der Zustände muss nur mit lokalen Daten gearbeitet werden. Zur Berechnung des neuen Zustands wird lediglich die Track-ID der betrachteten Zelle und ihrer Nachbarn benötigt. Es müssen wenig Daten verarbeitet und keinerlei Kombinationen oder ähnliches gebildet werden. Zudem müssen für jede Zelle ausschließlich die Nachbarn und der Zustand in Form einer Integerzahl gespeichert

werden, was den benötigten Speicherplatz stark einschränkt. Ein weiterer Vorteil des Verfahrens ist, dass es sich auf andere Detektoren übertragen lässt, für die eine Nachbarschafts-Geometrie vorliegt. Das grundlegende Prinzip kann wiederverwendet werden.

4.2 Kreisfit

Aus dem vorherigen Abschnitt geht hervor, dass das alleinige Anwenden des CA nicht ausreicht, um die Tracks vollständig zu rekonstruieren. Die Tracklets müssen im Anschluss noch miteinander kombiniert werden. Als Kriterium für das Bilden von Kombinationen werden Kreisparameter zu den Hits der Tracklets berechnet und bewertet. Im Anschluss wird die grundlegende Idee der Anwendung einer Kreisapproximation motiviert und das dazu eingesetzte Verfahren beschrieben. Als Letztes werden dabei entstandene Probleme erläutert.

4.2.1 Grundlegende Idee

Die Flugbahnen der Teilchen haben unter optimalen Bedingungen die Form einer Helix, die in der Projektion Kreise ergibt. Dies resultiert aus der wirkenden Lorentz-Kraft auf die geladenen Teilchen im Magnetfeld. Im Regelfall kommt es zu Wechselwirkungen der Teilchen, die zur Folge haben, dass die Flugbahnen verzerrt werden und der Radius abnimmt. Da es sich aber meist um hochenergetische Teilchen handelt, die nur wenig wechselwirken, kann von einer Flugbahn in Form eines projizierten Kreises ausgegangen werden. Diese Information soll zum Verbinden der Tracklets genutzt werden.

Es werden Tracklets miteinander kombiniert. Zu der Menge der Mittelpunkte aller Straw Tubes dieser Kombination wird ein Kreis berechnet, der die Punkte bestmöglich approximiert. Anschließend wird die Qualität des berechneten Kreises bewertet und anhand dessen entschieden, ob die Kombination der Tracklets sinnvoll ist. Ursprünglich sollten zu jedem Tracklet, das im ersten Schritt des Verfahrens generiert wurde, einmalig die Kreisparameter bestimmt werden. Die Kreisapproximation müsste für zusammengehörige Tracklets nahezu gleiche Kreisparameter ergeben, da sie auf der selben Kreisbahn liegen. Tracklets, für die ein Kreis mit sehr ähnlichem Radius und Mittelpunkt berechnet wurden, gehören vermutlich zum gleichen physikalischen Track. Basierend auf den Kreisparametern sollten Kombinationen von Tracklets gebildet werden. Dies hätte den Vorteil, dass der Kreis-Fit nur einmalig für die Start-Tracklets durchgeführt werden muss und das zeitintensive „Ausprobieren durch Kombinieren“ entfällt. Diese Vorgehensweise konnte nicht umgesetzt werden, da der Kreis-Fit für einige Tracklets nicht den Kreis im Sinne der Flugbahn der Teilchen berechnet wie es in Abschnitt 4.2.3 erläutert ist. Aus diesem Grund wird erst kombiniert und dann die Güte der Kombination durch die Bewertung des

(gemeinsamen) Kreis-Fits bestimmt.

Zur Bewertung des Kreisapproximation reicht es nicht aus, den mittleren Abstand aller Punkte zur Kreisbahn zu berechnen. Von den Tracklet-Kombinationen ist nicht zwingend diejenige die beste, deren Kreis-Approximation den kleinsten mittleren Abstand zu den Mittelpunkten der Straw Tubes hat. Gibt es einen Ausreißer, einen Punkt, der weit entfernt von der Kreisbahn liegt, während alle anderen Punkte sehr gut approximiert werden, ist der mittlere Abstand klein, aber die Kombination/Approximation eventuell unbrauchbar. Dem liegt die Überlegung zugrunde, dass alle Punkte einen Abstand von der Kreisbahn haben dürfen, der maximal dem Radius R einer Straw Tube entspricht. Die Kreisbahn stellt die angenäherte Flugbahn der Teilchen dar. Verfäuft diese durch eine Straw Tube, muss sich im Umkreis mit dem Radius R der Mittelpunkt der Straw Tube befinden. Es muss ein Signal ausgelöst worden sein, dessen Position durch diesen Mittelpunkt lokalisiert wird. Wird ein Mittelpunkt gefunden, der weiter als R von der Kreisbahn entfernt ist, kann dieser nicht zu einer Straw Tube gehören, die von der Kreisbahn passiert wird. Die Approximation ist ungenügend oder die Kombination der Tracklets war nicht sinnvoll. In beiden Fällen muss die Kombination mit der Approximation der zugehörigen Kreisbahn verworfen werden.

Zur Approximation muss ein Verfahren angewendet werden, das schnell eine Lösung liefert, da es einen Teil eines Online-Trackfinders darstellt. Aus diesem Grund sind iterative Annäherungsverfahren ungeeignet. Ein Standardverfahren zur Berechnung einer Ausgleichskurve zu einer gegebenen Menge von Punkten ist die Methode der Kleinsten Quadrate, mit welcher die Summe der quadratischen Abstände der Punkte von der Kurve minimiert wird. Handelt es sich bei der Ausgleichskurve um eine Linearkombination der gesuchten Parameter, lässt sich das Problem analytisch über das Bilden von Ableitungen lösen. Die allgemeine Kreisfunktion $(x-a)^2 + (y-b)^2 = r^2$ mit (a, b) als Mittelpunkt und r als Radius erfüllt diese Bedingung jedoch nicht und kann nicht analytisch gelöst werden. Es muss ein alternatives Verfahren eingesetzt werden, dass eine explizite Lösung der Kreisapproximation bietet.

4.2.2 Riemann-Fit

In diesem Abschnitt wird ein Verfahren vorgestellt, dass die Approximation eines Kreises in ein lineares Problem überführt und eine explizite Lösung bietet. Der Riemann-Fit wurde bereits in PandaRoot implementiert und ermöglicht die Kreisapproximation ohne ein iteratives Vorgehen oder den Einsatz numerischer Verfahren. Dies war ausschlaggebend für die Wahl des Algorithmus, dessen Funktionsweise kurz erläutert wird.

Der sogenannte *Riemann-Fit* projiziert die zweidimensionalen Punkte (u_i, v_i), die durch einen Kreis anzunähern sind, auf eine Oberfläche im Raum, sodass Kreise in der Ebene (u, v) auf eine eindeutige Schnitt-Ebene durch diese

Oberfläche abgebildet werden. Für die Projektionspunkte wird dann die bestmögliche Ausgleichs-Ebene berechnet. Aus den Parametern der Ebene lassen sich Rückschlüsse auf die gewünschten Kreisparameter ziehen. Das quadratische Problem der Kreisapproximation im Zweidimensionalen wird in ein lineares Problem der Berechnung einer Ebene im Dreidimensionalen überführt.[2]

Die Punkte werden auf eine Riemannsche Fläche projiziert, woraus sich die Namensgebung ergibt. Dabei handelt es sich in diesem Fall um einen zirkularen Paraboloiden, der sein Minimum im Koordinatenursprung hat und dessen Symmetrie-Achse entlang der z-Achse verläuft. Die wesentlichen Schritte des Verfahrens sind in der Abbildung 4.4 schematisch dargestellt.

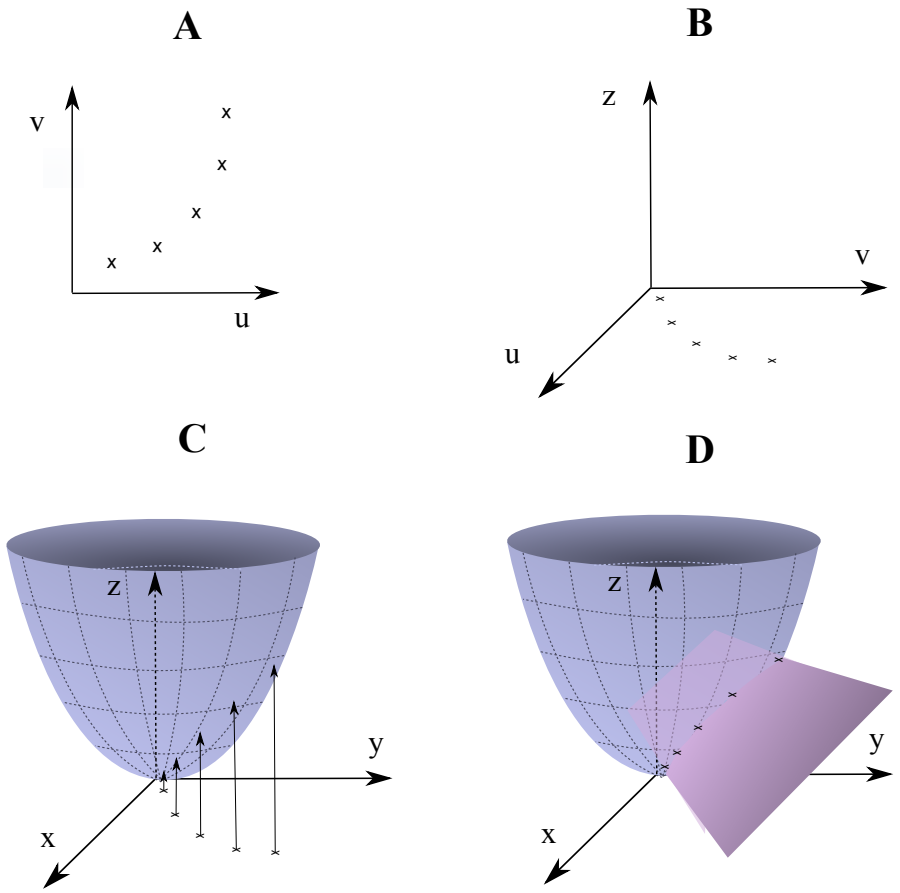

Abbildung 4.4: Skizzierte Schritte des Riemann-Fits: A - Punkte, die durch einen Kreis anzunähern sind, B - Hinzunahme einer weiteren Dimension, C - Projektion der Punkte auf den Paraboloiden, D - Ebene durch die Projektionspunkte

[2]Sämtliche Formeln und Informationen über die einzelnen Schritte des Verfahrens wurden aus [13] und [14] entnommen.

Ausgehend von den Messpunkten (u_i, v_i), die in diesem Fall die Mittelpunkte der Straw Tubes präsentieren, erfolgt durch

$$x_i = u_i \tag{4.1}$$
$$y_i = v_i \tag{4.2}$$
$$z_i = u_i^2 + v_i^2 \tag{4.3}$$

eine Transformation in dreidimensionale Punkte. Die Punkte (x_i, y_i, z_i) stellen die Projektionspunkte auf dem Paraboloiden dar. Wird von einem Punkt (u_i, v_i) eine zur u, v-Ebene senkrechte Gerade gezogen und der Schnittpunkt der Gerade mit dem Paraboloiden bestimmt, wird ebenfalls der transformierte Punkt erhalten.

Die Kreisgleichung in der u, v-Ebene sei als

$$(u - u_0)^2 + (v - v_0)^2 = \rho^2 \tag{4.4}$$

mit dem Radius ρ und dem Mittelpunkt (u_0, v_0) definiert. Durch den Einsatz der Formeln aus 4.1 ergibt sich auf dem Paraboloiden die Ebene

$$z - 2v_0 y - 2u_0 x + u_0^2 + v_0^2 - \rho^2 = 0. \tag{4.5}$$

Um eine Ebene bestmöglich an die transformierten Messwerte anzupassen, wird die Summe der quadratischen Abstände zu dieser Ebene minimiert. Dazu wird die Hesse-Normalform einer Ebene im Raum betrachtet. Diese lautet

$$c + n_1 x + n_2 y + n_3 z = 0, \tag{4.6}$$

wobei c den Abstand der Ebene zum Ursprung und $n^T = (n_1, n_2, n_3)$ den (normalisierten) Normalenvektor definiert. Dementsprechend ist das Minimum von

$$S = \sum_{i=1}^{N} \frac{(c + n_1 x_i + n_2 y_i + n_3 z_i)^2}{\sigma_i^2} = \sum_{i=1}^{N} \frac{d_i^2}{\sigma_i^2} \tag{4.7}$$

bezüglich der unbekannten Parameter c und n zu ermitteln. N stellt die Anzahl der Messwerte dar. Die einfließenden Abstände d_i werden mit der Varianz der Messfehler, σ_i^2, in der (u, v)-Ebene gewichtet. σ_i^2 ist durch die Fehler der Detektoren, die bei der Messung der Spurpunkte gemacht werden, gegeben. Die Einbeziehung der Messfehler sorgt dafür, dass Messwerte mit einer hohen Genauigkeit einen größeren Einfluss auf das Ergebnis, als ungenauere Werte haben. Eine Berechnung des Minimums bezüglich c über das Lösen von $\frac{\partial S}{\partial c} = 0$ ergibt

$$c = -n^T r_{cg} \tag{4.8}$$

mit dem Schwerpunktvektor $r_{cg}^T = (x_{cg}, y_{cg}, z_{cg})$. An dieser Stelle fließen die Messwerte erneut gewichtet ein. Die einzelnen Komponenten des Schwerpunktes

$$x_{cg} = \sum_{i=1}^{N} w_i x_i \tag{4.9}$$

$$y_{cg} = \sum_{i=1}^{N} w_i y_i \tag{4.10}$$

$$z_{cg} = \sum_{i=1}^{N} w_i z_i \tag{4.11}$$

$$\tag{4.12}$$

werden mit dem Gewicht

$$w_i = \frac{1/\sigma_i^2}{\sum\limits_{j=1}^{N} 1/\sigma_j^2} \tag{4.13}$$

entsprechend ihrer Messgenauigkeit in den Schwerpunkt miteinbezogen. Durch das Einsetzen der Bedingung 4.8 in 4.7 lässt sich S in der Form

$$S = n^T A n \tag{4.14}$$

angeben werden wobei

$$A = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\sigma_i^2} (r_i - r_{cg})(r_i - r_{cg})^T \tag{4.15}$$

mit $r_i^T = (x_i, y_i, z_i)$. A stellt die Kovarianzmatrix der Variablen x_i, y_i und z_i dar. Um S bezüglich n zu minimieren, wird n durch die Eigenvektoren u_j von A dargestellt: $n = \sum\limits_{j=1}^{3} a_j u_j$. Da die Kovarianzmatrix symmetrisch ist, kann in die Form $A = UDU^T$ überführt werden (Diagonalisierung). D enthält auf der Diagonalen die Eigenwerte λ_j und U die Eigenvektoren u_j. Dies liefert den Ausdruck

$$S = \sum_{j=1}^{3} a_j^2 \lambda_j. \tag{4.16}$$

S wird minimiert, wenn n dem zum kleinsten Eigenwert von A gehörenden Eigenvektor entspricht. Das Problem wurde auf das Finden eines Eigenwertes und Eigenvektors einer 3x3-Matrix reduziert und kann analytisch gelöst werden. Durch die Bestimmung von n und Einsetzen in 4.8 kann auch c direkt ermittelt werden. Die Abbildung der Parameter der Ebene auf die Kreisparameter lässt sich mit Hilfe der Hesse-Normalform 4.6 aus 4.5 ermitteln. Die Kreisparameter ergeben sich wie folgt:

$$u_0 = -\frac{n_1}{2n_3} \tag{4.17}$$

$$v_0 = -\frac{n_2}{2n_3} \tag{4.18}$$

$$\rho^2 = \frac{1 - n_3^2 - 4cn_3}{4n_3^2} \tag{4.19}$$

Mit anschließenden Schritten können auch Flugbahnen in Form einer Helix approximiert werden. Da für die Spur-Rekonstruktion im STT in diesem Fall ausschließlich die x-y-Projektion von Flugbahnen betrachtet wird, ist ein derartiger Helix-Fit nicht erforderlich.

Um den zwei-dimensionalen Kreis-Fit zu einem Helix-Fit zu erweitern, wird nach der Bestimmung der Kreisparameter die Bogenlänge jedes Messpunktes auf dem gefitteten Kreis berechnet. Die Bogenlänge wird gegen die gemessenen z-Koordinaten der Messpunkte aufgetragen. Dies sollte eine Gerade ergeben, die im nächsten Schritt über eine lineare Regression angepasst wird. Weiterführende Informationen dazu sind in [5] zu finden.

4.2.3 Bewertung des Verfahrens

Im Folgenden werden die Kreisbahnen untersucht, die der Riemann-Fit für die Start-Tracklets berechnet. Es wird gerechtfertigt, warum erst ein Kombinieren und eine anschließende Bewertung über die Kreisapproximation zum Zusammenführen von Tracklets erforderlich ist.

Von dem Verfahren können vorerst nur die Signale bzw. Mittelpunkte von nicht gedrehten Straw Tubes betrachtet werden. Zur Approximation eines Kreises wird mit den x- und y-Koordinaten des Mittelpunktes der Straw Tubes gearbeitet. Bei den zur Strahl-Achse parallelen Röhrchen sind diese Koordinaten für alle z gleich. Für die gedrehten Straw Tubes müssen die x- und y-Koordinaten unter Berücksichtigung der aktiven Nachbarn noch berechnet werden, da diese entlang des Strahls verschiedene Werte annehmen. Ansonsten würden in der Referenzebene die Koordinaten des Mittelpunktes auf halber Höhe der Straw Tubes in den Kreisfit eingehen, was die Lösung verfälschen würde.

In Abbildung 4.5 sind die Mittelpunkt der Straw Tubes der im ersten Schritt erzeugten Tracklets und die entsprechenden approximierten Kreise dargestellt. Es wurden nur die Punkte eingetragen, die in den Kreis-Fit einfließen. Dementsprechend sind keine Mittelpunkte von gedrehten Straw Tubes zu sehen. Für Tracklets mit weniger als drei Hits können keine Kreisparameter bestimmt werden. Besitzt ein Tracklet genügend Hits und ist trotzdem kein Kreis zu sehen, liegen die Hits nahezu auf einer Geraden, was in einem Kreis mit einem sehr großen Radius resultiert, der nicht eingezeichnet wurde. Das zugehörige Event mit den zu rekonstruierenden Flugbahnen ist in Abbildung 4.6 zu sehen.

Das Problem des Verfahrens wird gut im unteren linken Quadranten des STTs sichtbar. Die Tracklets auf ungefähr gleicher Höhe gehören zum gleichen physikalischen Track. Das Verfahren liefert nicht die gewünschten Kreise, die die Flugbahn approximieren sollen. Für die vier Tracklets werden Kreise berechnet, die viele Hits der Tracklets einschließen. Die Kreisparameter der Tracklets lassen sich nicht miteinander in Beziehung setzen, da sie sich nicht ähneln. Es werden Kreise benötigt, deren Kreisbahn entlang der Hits verläuft,

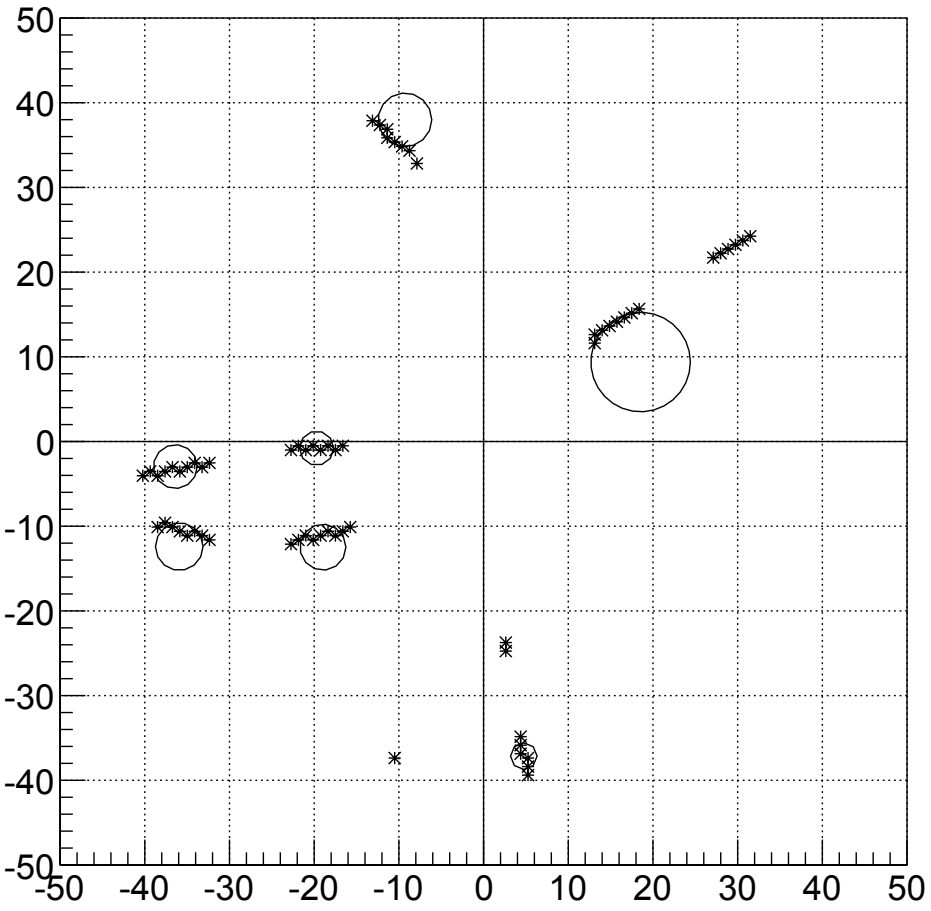

Abbildung 4.5: Darstellung der Mittelpunkte der Straw Tubes der Start-Tracklets und der resultierenden Kreisapproximationen. Im Ursprung befindet sich das Zentrum des STTs.

sodass alle Hits einen kleineren Abstand als R zu dieser Kreisbahn haben (siehe Abschnitt 4.2.1). Auch die Tracklets im oberen rechten Quadranten sind dem gleichen physikalischen Track zuzuordnen. Der Kreis für das äußere Tracklet ist nicht zu sehen, weil die Punkte nahezu auf einer Geraden liegen und der Mittelpunkt des Kreises sehr weit entfernt liegt. Der Kreis des inneren Tracklets ist dagegen sehr klein und die Kreisparameter stehen in keiner Relation.

Das aufgetretene Problem lässt sich auf die „Ungenauigkeit“ der STT-Hits zurückführen. Die Mittelpunkte der Straw Tubes eines Tracklet können aufgrund der Geometrie des STTs einer „Zick-Zack-Linie“ folgen (siehe Abb. 4.7). Dies führt dazu, dass der bestmögliche Kreis einen kleinen Radius hat und einige STT-Hits des Tracklets einschließt. Es wird nicht wie erwartet eine Kreisbahn berechnet, die die „Zick-Zack-Linie“ ausgleicht. Das Verfahren müsste die Randbedingung beachten, dass alle Messwerte einen maximalen Abstand von R zur Kreisbahn haben sollen. Es müsste das Minimum ermitteln, dass diese Randbedingung erfüllt und andere Approximationen verwerfen.

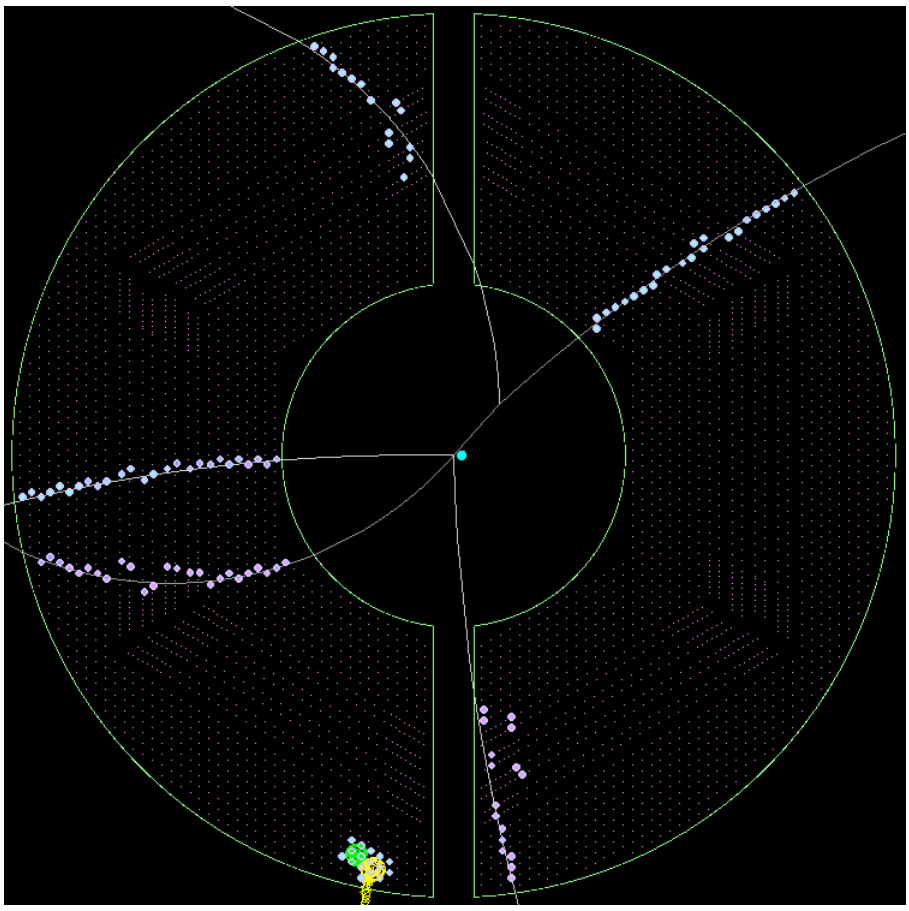

Abbildung 4.6: Event dessen Tracklets mit Hilfe des Riemann-Fits verbunden werden sollen.

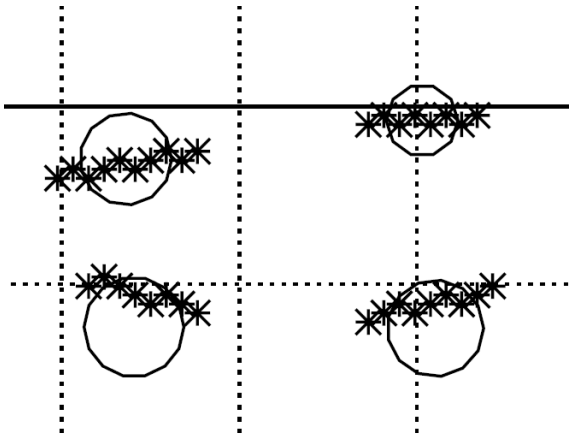

Abbildung 4.7: Vergrößerter Ausschnitt der Tracklets in Form von „Zick-Zack-Linien“ aus Abb. 4.5

Um diese komplexe Aufgabe zu umgehen, werden erst Kombinationen von Tracklets gebildet und dann die Kreisparameter berechnet. Zur Approximation stehen dann mehr Messpunkte zur Verfügung, sodass eine Approximation der gewünschten Kreisbahn wahrscheinlicher ist. Nicht alle Trackles folgen einer „Zick-Zack-Linie“. Durch die Kombination mit einem andersartigen Tracklet und einen anschließenden Fit, kann das Problem verringert werden. Außerdem erschwert der Ausschluss der gedrehten Driftröhrchen die Approximation, da weniger Informationen genutzt werden, als eigentlich vorhanden sind. Mit der Einbeziehung dieser Hits ist eine bessere Approximation wahrscheinlicher.

4.3 Entwickelter Trackfinding-Algorithmus

In diesem Abschnitt wird das zum Trackfinding eingesetzte Verfahren beschrieben. Als Erstes werden mit Hilfe eines zellulären Automaten Tracklets generiert, wobei Ambiguitäten außer Acht gelassen werden. Dabei kann ein physikalischer Track in mehrere Tracklets zerfallen. Diese werden im nächsten Schritt durch das Bilden von geeigneten Kombinationen mit Hilfe des Riemann-Fits zusammengeführt. Dabei werden die Kombinationsmöglichkeiten der Tracklets stark eingeschränkt, um ein schnelles Verfahren zu realisieren. Als Letztes werden die fehlenden Hits, die im ersten Schritt nicht beachtet worden sind, den Tracklets zugewiesen. Es folgt eine Erläuterung dieser Stufe und eine Bewertung ihrer Parallelisierbarkeit.

Nach Anwendung des Verfahrens ergeben sich Track-Kandidaten, die die potentiellen Tracks darstellen. Der bisherige Entwicklungsstand des Verfahrens liefert in der Regel mehr Track-Kandidaten als eigentlich vorhanden sind. Das bedeutet, dass bei Unsicherheiten mehrere potentielle Tracks vermerkt werden, um das fälschliche Verwerfen von richtigen Möglichkeiten zu vermeiden. Außerdem enthalten die Track-Kandidaten unter Umständen weniger Hits, da die Signale von gedrehten Straw Tubes noch nicht vollständig in das Verfahren integriert worden sind.

4.3.1 Finden der ersten Tracklets

Im ersten Schritt des Verfahrens werden die STT-Hits zu ersten Wegstücken vereint. Dieser Schritt folgt dem Ansatz eines zellulären Automaten, wie er in Abschnitt 4.1 beschrieben wurde. Zu dessen Umsetzung wird zunächst geklärt, mit welchen Informationen gearbeitet werden kann.

Die Hits werden in einem eindimensionalen Feld zur Verfügung gestellt, welches ab sofort als „Hit-Feld“ bezeichnet wird. Jeder Hit kann mit der Straw Tube, die dieses Signal ausgelöst hat, in Verbindung gebracht werden. Zu jeder Straw Tube gibt es eine eindeutige Nummer, eine „Tube-ID“, über die sie identifiziert werden kann. Anhand dieser Tube-ID lassen sich die Nachbar-

schaftsbeziehungen ermitteln. Sie gibt Aufschluss über die genaue Lage der Straw Tube im STT. Die Driftröhrchen sind von innen nach außen gegen den Uhrzeigersinn durchnummeriert. Die Nummerierung beginnt bei 1 und endet mit 4542.[3] Im Folgenden ist daher von Tube-IDs statt Hits bzw. Zellen die Rede. Ein Tracklet wird als eine Folge von Tube-IDs angesehen. Zuordnungen und Operationen, die sich auf Zellen beziehen, werden mit Hilfe der Tube-IDs realisiert.

Um den zellulären Automaten anwenden zu können, muss als erstes der *Zustand* der Zellen initialisiert werden. Diese Operation muss nur für die Tube-IDs des Hit-Feldes durchgeführt werden, da sich nur die Zustände von aktiven Zellen verändern. Für jede Tube-ID wird zu Beginn der Status auf die Tube-ID dieser Straw Tube gesetzt, sodass nicht zusätzlich unterschiedliche Zahlen erzeugt werden müssen. Das hat zur Folge, dass nach dem Ablauf des Algorithmus der Status der Zellen, die zu einem Tracklet gehören, der Tube-ID des Tracklets entspricht, die dem Zentrum des STTs am nächsten ist. Bei der Anpassung der Zustände nehmen die Zellen eines Tracklets nach und nach die kleinste Tube-ID der Hits des Tracklets als Status an.

Als Nächstes werden die *Nachbarschaftsbeziehungen* für die Zellen des Hit-Feldes ermittelt. Auch hierfür müssen wieder nur Tube-IDs vermerkt werden, die im Hit-Feld vorhanden sind, da zur Evaluierung der Zustände nur aktive Nachbarzellen miteinbezogen werden. Welche Zellen benachbart sind, ist in einer „Straw Map“ vermerkt. Bei dieser handelt es sich um einen schematischen Querschnitt durch den STT in dem alle Straw Tubes mit ihrer Tube-ID versehen sind. Die Zuordnung von den Tube-IDs der aktiven Nachbarn zu den Tube-IDs aus dem Hit-Feld ist essentiell für den Algorithmus. Diese Information muss jederzeit vorliegen und auf sie sollte schnell zugegriffen werden können. Es ergeben sich Zuordnungen der Form: „<Tube-ID> <Tube-IDs der aktiven Nachbar-Zellen>“.

Der Status wird nur für aktive Zellen mit ein oder zwei aktiven Nachbarn aktualisiert. Daher werden die ermittelten Zuordnungen im Anschluss in Bezug auf die Anzahl der Hit-Nachbarn *gruppiert*. Für jede aufgestellte Zuordnung werden die aktiven Nachbarn gezählt. Auf diese Weise werden die Tube-IDs aus dem Hit-Feld in die Gruppen 0, 1, ..., 7 unterteilt, wobei alle Straw Tubes mit mehr als 6 Nachbarn in der letzten Gruppe 7 zusammengefasst werden.

Es folgt die *Anpassung der Zustände* für die Zellen der Gruppen 1 und 2, indem die Regeln des zellulären Automaten angewendet werden. Für jede Zelle dieser Gruppen werden die aktiven Nachbarn bestimmt, die ebenfalls in Gruppe 1 oder 2 enthalten sind, da zur Bestimmung des Minimums nur gleichartige Nachbarn betrachtet werden. Zur simultanen Änderung der Zustände muss der neue berechnete Zustand zwischengespeichert werden. Nachdem dieser für alle berechnet worden ist, wird der Gesamtzustand des Systems aktualisiert. Alle

[3] PandaRoot verwendet hier ein neues Detektor-Layout, das weniger Straw Tubes enthält, als im technischen Design-Report des STTs vorgesehen sind.

Zuordnungen der Form „<Tube-ID> <Track-ID>“ werden nach der Bestimmung der Minima durch eine neue Liste von Zuordnungen ersetzt. Dies wird solange wiederholt, bis sich kein Status mehr ändert.

Durch das Gruppieren der Zellen nach ihrem Status, ergeben sich die ersten Wegstücke der Tracks. Damit ist das Generieren der Tracklets abgeschlossen. Als Ergebnis können Tracklets vorliegen, die schon alle Tube-IDs eines kompletten Tracks umfassen. Es können aber auch Tracklets mit nur einem Hit zu Stande kommen.

4.3.2 Zusammenführen der Tracklets

Bei den im ersten Schritt erzeugten Tracklets handelt es sich nur teilweise um vollständig rekonstruierte Tracks. Bei Ambiguitäten zerfällt ein Track in mehrere Tracklets, die wieder zusammengeführt werden müssen. Dazu wird das in Abschnitt 4.2 beschriebene Verfahren eingesetzt. Die Tracklets werden miteinander kombiniert und für diese Kombination wird eine Kreisbahn approximiert. Genügt die Approximation den Anforderungen, wird die Kombination akzeptiert. Dabei werden Hits von gedrehten Straw Tubes nicht berücksichtigt.

Begrenzung der Kombinationsmöglichkeiten

Da das Kombinieren eines jeden Tracklets mit allen anderen Tracklets nicht sinnvoll und sehr zeitintensiv ist, wurde nach Möglichkeiten gesucht, dies einzuschränken. Die Einschränkung darf aber nur in einem Maß erfolgen, das das Auffinden der meisten Tracks noch zulässt. An dieser Stelle wird nun erklärt, wie die Kombinationsmöglichkeiten begrenzt werden.

Es werden nur Tracklets kombiniert, die in Verbindung einen sinnvollen Track ergeben können, der vom Inneren des STTs nach außen verläuft. Zu diesem Zweck wird für jedes Tracklet ermittelt, ob es sich von der Mitte des STT strikt nach außen bewegt oder wieder zur Mitte führt. Demzufolge werden stark gekrümmte Tracklets an dieser Stelle von dem Kombinieren ausgeschlossen. Diese Einschränkung lässt sich mit Hilfe der Tube-IDs der Hits eines Tracklets realisieren. Verläuft ein Tracklet nach außen, so besitzt die Straw Tube am äußeren Ende die maximale Tube-ID aller IDs, die dem Tracklet zugewiesen wurden. Da die Tube-IDs der Straw Tubes von Reihe zu Reihe nach außen immer größer werden, lässt sich das auf diesem Weg leicht prüfen. Befinden sich mehrere Hits in der äußersten Reihe eines Tracklets, so kann keine Aussage darüber gemacht werden, ob es eine Tendenz nach außen oder innen hat. In diesem Fall werden solche Tracklets als „nicht nach außen verlaufend“ gewertet. Die Information für den Verlauf der Tracks muss nur einmalig ermittelt werden und fällt daher nicht stark ins Gewicht.

Eine derartige Bestimmung dieser Eigenschaft ist sehr einfach, dafür jedoch für einige Fälle fehlerhaft. Es gibt Tracklets, die Straw Tubes enthalten, die

der Zählrichtung der Tube-IDs folgen. Das heißt, es gibt mehrere aufeinander folgende Tube-IDs, die sich in einer Reihe befinden. Befindet sich die abschließende Straw Tube jedoch alleine in der nächsten Reihe, wodurch sie eine größere Tube-ID hat, wird dieses Tracklet fälschlicherweise als „nach außen verlaufend“ angesehen. Diese Fälle werden noch nicht gesondert behandelt. Es sind aber auch nicht viele Tracklets zu erwarten, auf die diese Besonderheit zutrifft.

Verläuft ein Tracklet nach außen, so muss es nur mit Tracklets kombiniert werden, die hinter der letzten Reihe des zu kombinierenden Tracklets beginnen. Soll Tracklet A mit Tracklet B kombiniert werden, lässt sich folgende Information nutzen: Die Kombination ist nur sinnvoll, wenn der Status von Tracklet B größer als die Tube-ID des abschließenden Hits des Tracklets A ist. Der Status eines Tracklets entspricht der Tube-ID, die dem Zentrum des STT am nächsten ist und somit der minimalen Tube-ID des Tracklets. Durch das Überprüfen, ob der Status von B größer als die End-ID von A ist, wird sichergestellt, dass nur hintereinanderliegende Tracklets kombiniert werden. Dadurch werden viele Kombinationsmöglichkeiten ausgeschlossen, die nicht nützlich sind, z. B. die Kombination zweier Tracklets, die nebeneinander verlaufen.

Vorerst wird nur nach Kombinationen zwischen einem nach außen verlaufenden Tracklet und anderen Tracklets gesucht. Im STT stark gekrümmte Flugbahnen werden dementsprechend nicht betrachtet. Außerdem werden nur Tracklets miteinander kombiniert, die sich aus mindestens drei Tube-IDs von nicht gedrehten Straw Tubes zusammensetzen. So werden nur größere Tracklets aneinander gefügt, weil diese eine bessere Grundlage für einen guten Kreisfit bieten. Die Rekonstruktion von Tracks, die sich von einer Hälfte des STTs in die andere bewegen, müssen gesondert behandelt werden und werden vorerst nicht berücksichtigt. Durch die große Lücke zwischen den Hälften, ergibt sich ein großer Spielraum für die Kombinationen. Die Kombinationen der Tracklets werden wie folgt eingeschränkt:

- Die Kombinationen erfolgen nur in derselben Detektor-Hälfte. Tracklets von einem Track, der beide Hälften passiert, werden dementsprechend nicht kombiniert.

- Kombiniert werden nur Tracklets, die mehr als drei Hits von nicht gedrehten Straw Tubes beinhalten.

- Es werden nur an nach außen verlaufenden Tracklets weitere angefügt. Die angefügten Tracklets müssen diese Eigenschaft aber nicht besitzen. Dies kann dazu führen, dass im STT zurück-kreisende Flugbahnen nicht rekonstruiert werden.

- Durch Berücksichtigung der Tube-IDs werden nur hintereinanderliegende Tracklets kombiniert.

Fortführendes Kombinieren

Wurden derartige Zweier-Kombinationen gebildet, werden diese auf mögliche Dreier-Kombinationen untersucht. Angenommen Tracklet A wurde mit Tracklet B und Tracklet B mit Tracklet C kombiniert. Dann ist es möglich die drei Tracklets A, B und C zu einem großen Tracklet zusammenzufügen. Die bisher gebildeten Kombinationen werden auf derartige Ketten untersucht und falls möglich zu größeren verbunden. Es ist auch vorstellbar Kombinationen aus einer größeren Anzahl von Tracklets zu bilden, bis keine mehr möglich sind. Da jedoch von vornherein eine Vielzahl von Tracklets mit zu wenig Signalen von nicht gedrehten Straw Tubes ausgeschlossen werden, sind diese Kombinationsmöglichkeiten eher unwahrscheinlich. Das Bilden von weiteren größeren Kombinationen wäre hauptsächlich zur Rekonstruktion von kreisenden Tracks („Wirbel“) nötig, da diese in viele kleine Tracklets zerfallen. Allerdings werden diese vom Verfahren ausgeschlossen. Dementsprechend sollte das Verfahren bei Einbeziehung von Hits der gedrehten Straw Tubes und stark gekrümmten Tracks, angepasst werden, sodass weiter kombiniert wird.

Filtern der Kombinationen

Trotz dieser starken Einschränkungen können immer noch Tracklets miteinander kombiniert werden, die eigentlich nicht zusammengehören. Daher muss im Anschluss geprüft werden, ob es sich um eine gute Kombination handelt. Die Anzahl der Mittelpunkte der Straw Tubes eines kombinierten Tracklets, die einen größeren Abstand als den Radius R eines Röhrchens zum Kreis haben, ist dafür ausschlaggebend, ob die Kombination akzeptiert oder verworfen wird (siehe Abschnitt 4.2.1). Beläuft sich die Anzahl auf Null, wird die Kombination akzeptiert. Da aber auch eine schlechte Approximation, dafür sorgen kann, dass der Abstand R überschritten wird, können richtige Kombinationen abgelehnt werden.

Es werden nur als gut befundene Kombinationen weiter kombiniert. Das heißt, die Dreier-Kombinationen werden nur aus bereits akzeptierten Zweier-Kombinationen gebildet.

Resultierende Tracklets

Als Ergebnis des zweiten Schritts entstehen im Regelfall größere Tracklets als jene, die in der ersten Stufe des Verfahrens erzeugt worden sind. Tracklets, die nicht kombiniert werden konnten, bleiben als solche erhalten und werden als Kandidaten für Tracks angesehen, wenn sie mindestens drei Hits besitzen. Kombinationen aus zwei Tracklets werden fortan ausschließlich in ihrer Kombination betrachtet. Das heißt, die ursprünglichen Tracklets (vor dem Kombinieren) werden nicht mehr als potentielle Tracks betrachtet. Werden aus Zweier-Kombinationen Dreier-Kombinationen gebildet, werden diese jedoch zusätzlich

erstellt und die Informationen über die alten Kombinationen bleiben erhalten. Als gut befundene Kombinationen werden stets als Track-Kandidaten angesehen. Dementsprechend kann ein Tracklet in mehreren Kombinationen auftauchen, falls mehrere passende ermittelt worden sind. Aus diesem Grund sind wesentlich mehr Track-Kandidaten zu erwarten, als es in Wirklichkeit gibt. Es müssen noch Entscheidungskriterien aufgestellt werden, die es ermöglichen, aus mehreren Kombinationen eines Tracklets die beste auszuwählen. Diese könnten sich aus der Kombination mit weiteren Informationen über Flugbahnen ergeben, die von anderen Detektoren ermittelt worden sind. Um das Verwerfen von richtigen Kombinationen zu vermeiden, wurde dies noch nicht umgesetzt.

4.3.3 Einbeziehen der fehlenden Hits

Nachdem das Kombinieren abgeschlossen wurde, werden noch nicht zugeordnete Hits zu den passenden Tracklets (falls vorhanden) hinzugefügt. Auch hier werden nur die Hits von nicht gedrehten Straw Tubes berücksichtigt, da eine Abstandsberechnung vorgenommen wird und die x- und y-Werte der Mittelpunkte dieser Straw Tubes noch geeignet berechnet werden müssen.

Die passenden Tracklets werden in zuvor gebildeten Kombinationen gesucht, aber auch in den Tracklets aus dem ersten Schritt, die nicht kombiniert worden sind und zu denen die Kreisparameter berechnet werden können. Die Suche wird auf dieselbe Detektor-Hälfte eingeschränkt, aus der der Hit stammt. Die Kreisparameter können bestimmt werden, wenn ein Tracklet mindestens drei Hits von nicht gedrehten Straw Tubes besitzt. Die Parameter für die Kombinationen wurden bereits berechnet und können hier wiederverwendet werden. Für Tracklets aus dem ersten Schritt muss die Approximation an dieser Stelle noch durchgeführt werden. Mit Hilfe dieser Kreisparameter wird der euklidische Abstand des Mittelpunktes der Straw Tubes der fehlenden Hits zur Kreisbahn berechnet. Es wird das Tracklet ermittelt, dass den geringsten Abstand zum noch nicht zugeordneten Hit/zur Straw Tube hat. Ist dieser Abstand maximal so groß wie der Radius einer Straw Tube, wird der Hit dem Tracklet zugeordnet. Wird auf diese Weise kein Tracklet gefunden, verbleibt der Hit ohne eine Zuweisung.

Als Erstes werden die Tube-IDs betrachtet, die drei oder vier aktive Nachbarn haben. Um welche Tube-IDs es sich dabei handelt wurde bereits im ersten Schritt des Verfahrens ermittelt. Die entsprechenden Hits wurden aufgrund von Ambiguitäten noch keinem Tracklet zugeordnet. Für diese Tube-IDs wird wie zuvor beschrieben nach passenden Tracklets gesucht. Anschließend werden die Hits von Start-Tracklets zugeordnet, die weniger als drei Hits besitzen und nicht kombiniert worden sind.

Es ist möglich, dass Hits falsch zugeordnet werden oder eine Zuordnung vollkommen ausbleibt. Eine Alternative wäre die Mittelpunkte der fehlenden Hits in die Kreis-Approximation aufzunehmen und diese zu bewerten. Dies

würde wahrscheinlich zu einem besseren Ergebnis führen, wäre aber nicht mehr so leicht parallelisierbar wie die ursprüngliche Methode und es muss wesentlich mehr berechnet werden. Die Kreisbahnen der Tracklets würde sich durch die Hinzunahme eines weiteren Punktes ändern. Daher müsste der Zugriff auf die Kreisparameter gesondert behandelt werden.

4.3.4 Parallelisierbarkeit

Eine besondere Anforderung an das Verfahren ist, dass es es gut parallelisierbar sein soll. Es soll zum Online-Trackfinding eingesetzt werden und möglichst schnell Informationen über die Tracks liefern.

Im ersten Schritt des Verfahrens wird ein Prinzip angewendet, das an die Funktionsweise eines zellulären Automaten angelehnt ist. Zelluläre Automaten sind für ihre hochgradige Parallelisierbarkeit bekannt. Auch in diesem Fall kann das Generieren der Tracklets gut parallelisiert werden. Die Nachbarschaftsbeziehungen müssen einmalig aufgestellt und können zeitgleich für mehrere Zellen ermittelt werden. Das anschließende Evaluieren der Zustände kann an mehrere Prozesse aufgeteilt werden. Es erfolgt unabhängig voneinander, da sich die neuen Zustände nur aus den Daten des vorherigen Iterationsschrittes ergeben. Jede Zelle benötigt nur die Informationen ihrer Nachbarzellen und passt ihren Status dementsprechend an. Es werden fast nur lokale Daten geändert. Lediglich die Berechnung der Summe der Track-IDs der Zellen zur Überprüfung der Abbruchbedingung muss speziell geregelt werden und sequentiell erfolgen. Ansonsten treten keine Situationen auf in denen versucht wird, einen Datensatz von mehreren Prozessen zeitgleich zu verändern.

Auch das Suchen und Testen der Kombinationen in der zweiten Phase des Verfahrens lässt sich gut parallelisieren. Das Berechnen der Kreisparameter für die Kombinationen erfolgt komplett unabhängig voneinander. Es müssen zwar teilweise die gleichen Daten gelesen werden, geschrieben werden aber nur lokale Daten (für eine Kombination).

Das abschließende Suchen der passenden Tracklets für die noch nicht zugeordneten Hits kann ebenfalls nebenläufig erfolgen, sodass der Vorgang pro Hit von einem Prozess ausgeführt werden kann.

5 Umsetzung des Algorithmus

Das im vorherigen Kapitel beschriebene Verfahren wurde mit C++ implementiert und in den bestehenden Quellcode von PandaRoot integriert. Zur Umsetzung des Verfahrens wurden neue Klassen entwickelt, deren Schnittstellen sich an bereits existierenden Trackfindern orientieren. Bereits implementierte Teilschritte wie der Kreisfit oder das Aufstellen der Nachbarschaftsbeziehungen der Straw Tubes konnten durch das Erzeugen entsprechender Objekte realisiert und mussten nicht selbst entwickelt werden. In diesem Kapitel wird darauf eingegangen, um welche Klassen es sich dabei handelt, welche Klassen zusätzlich definiert worden sind und wie diese miteinander in Beziehung stehen. Dieses Kapitel trägt nicht zum Verständnis des Verfahrens bei, ist aber wichtig, um die Implementierung nachzuvollziehen und gegebenenfalls erweitern zu können.

5.1 Programmierung in PandaRoot

In diesem Abschnitt werden kurz einige Richtlinien und Vorgehensweisen bezüglich der Programmierung in PandaRoot beschrieben. Diese betreffen auch ROOT und FairRoot, da ROOT das grundlegende Basis-Framework ist, FairRoot Funktionen für alle FAIR-Komponenten anbietet und PandaRoot spezielle Funktionalitäten von $\overline{\text{P}}$ANDA implementiert. Grundsätzliche Eigenschaften von ROOT und die einzelnen Schritte zur Simulation eines physikalischen Experimentes wurden bereits in Abschnitt 3.3 erläutert. Die folgenden Informationen beschränken sich speziell auf die Aspekte, die die Implementierung des Trackfinding-Verfahrens betreffen.[1]

5.1.1 Programmierkonventionen

Da PandaRoot von mehreren internationalen Arbeitsgruppen weiterentwickelt wird, gibt es Programmierrichtlinien, die zur Lesbarkeit des Programms beitragen. Davon sind im Folgenden einige aufgelistet, die vor allem für die Namensgebung der Klassen und ihrer Attribute ausschlaggebend waren.

- Alle Klassennamen des PandaRoot-Frameworks starten mit dem Präfix

[1] Sie wurden dem Benutzerhandbuch von ROOT [18] und dem PandaRoot-Wiki [6] entnommen.

„Pnd“, die FairRoot-Klassen mit „Fair“ und die Klassen von ROOT mit einem „T“.

- Der Name der Attribute einer Klasse beginnt mit einem „f“, um Namenskonflikte mit Zugriffsfunktionen zu vermeiden. „f“ steht dabei für „Field“ bei dem es sich um einen historischen Begriff für Membervariablen handelt.

- Dem Namen von `typedef`s und `struct`s wird die Endung „_t“ angehängt.

- Methodennamen starten mit einem Großbuchstaben.

- Um Maschinen-unabhängige Datentypen bereitzustellen, wurden Datentypen mit einer festen Größe wie `Int_t` oder `Double_t` definiert. Sollen die Daten nicht abgespeichert werden, können diese Typen gleichbedeutend zu den Standardtypen verwendet werden (z. B. `Int_t` und `int`).

5.1.2 Einbindung in PandaRoot

Um ein selbst entwickeltes Trackfinding-Verfahren in PandaRoot zu integrieren, muss eine *Task* erstellt werden. Von dieser Trackfinder-Task wird in einem Rekonstruktions-Makro ein Objekt erzeugt und der Laufzeit-Umgebung als eine abzuarbeitende Aufgabe bekannt gemacht. Tasks werden nicht nur für das Trackfinding erstellt, sondern auch z. B. für das anschließende Trackfitting. Daher gibt es eine Basis-Klasse namens `FairTask`, die das Grundgerüst einer Task definiert, welches entsprechend angepasst werden muss. In einer Task werden Arbeitsschritte definiert, die für jedes Event gleichartig abgearbeitet werden. Die wichtigsten Methoden, die zu dessen Umsetzung überschrieben werden müssen, sind im Folgenden kurz erläutert.

`Init()` Hier erfolgen Vorbereitungen zur Abarbeitung der Task. Es werden benötigte Daten und Objekte initialisiert bzw. eingelesen. Die Funktion wird einmalig nach dem Starten der Task aufgerufen.

`Exec()` Die Methode implementiert die abzuarbeitende Aufgabe und wird für jedes Event aufgerufen. Berechnete Ergebnisse sollten an dieser Stelle gespeichert werden.

`FinishEvent()` Die Funktion wird nach der Ausführung der Task für ein Event aufgerufen (also nach `Exec()`) und dient zum Leeren von Event-spezifischen Speicherpuffern. So wird eine klare Trennung der Daten je Event erreicht.

`Finish()` Eine Implementierung dieser Funktion dient zur Nachbereitung der Daten nach Abarbeitung der Task für alle Events (z. B. zur Analyse in

Form von Histogrammen). Sie wird nach Beendigung der vollständigen Task einmalig aufgerufen.

Der aufrufenden Umgebung sind die Anzahl der Events bekannt für die das Trackfinding durchgeführt werden soll. Die Auswahl und Reihenfolge der Tasks wird durch die Makros und das Rahmenprogramm gesteuert. Die `FairTask`-Klasse ermöglicht eine beliebige Anpassung und Erweiterung der Schleife über die Ereignisse und macht die FairRoot-Applikationen damit sehr flexibel.

Grundlage für eine erfolgreiche Ausführung ist, dass der Root-File mit den entsprechenden Simulationsdaten vorliegt. Das heißt, die Event-Generation, das Anwenden eines Transport-Modells und die Digitalisierung der Daten hat bereits stattgefunden. In einem ROOT-File müssen u. a. Informationen über die Monte-Carlo-Tracks und die aus ihnen erzeugten STT-Hits gespeichert sein.

5.1.3 Lesen und Schreiben von Daten

ROOT verwendet eine eigene Datenstruktur zum Speichern von Objekten. Diese ROOT-Dateien weisen eine Baumstruktur auf, in der Daten auf mehreren Ebenen abgelegt werden können.

Während der Programmausführung steht ein IO-Manager von FairRoot zur Verfügung. Dieser hat Zugriff auf die im aufrufenden Makro definierten ROOT-Dateien, die als Ein- und Ausgabedateien genutzt werden. Um auf bestimmte Zweige einer ROOT-Datei zuzugreifen, muss ein Branch-Name angegeben werden, der den Zweig identifiziert. Unter dem Zweig „STTHit“ sind z. B. alle Objekte zu den Hits des STTs zu finden. Zum Abspeichern von Daten müssen ein neuer Zweig und eine Datenstruktur beim IO-Manager registriert werden. Die Daten werden aus der angegebenen Datenstruktur entnommen und sind dann unter dem definierten Zweig zu finden.

In diesem Fall werden die zu speichernden Objekte in `TClonesArray`s abgelegt und als solche in der ROOT-Datei abgespeichert. Ein `TClonesArray` ist eine Realisierung eines Feldes von ROOT, die eigenständig und effizient den Speicherplatz verwaltet. Das Feld speichert ausschließlich gleichartige Objekte (gleicher Typ) und wurde speziell für ein wiederholtes Erstellen oder Löschen von gleichen Objekten entwickelt.

5.2 Konzeption der Klassen

5.2.1 Zusätzlich erstellte Klassen

Zur Entwicklung des Trackfinders wurden zwei neue Klassen erstellt: `PndSttCellTrackFinderTask` und `PndSttCellTrackFinder`. Um das Verfahren zu testen, wurde die Klasse `PndSttCellTrackFinderAnalysisTask` entwickelt, welche erst in Kapitel 6 erläutert wird. Nachfolgend wird die Funktionalität

der anderen beiden Klassen beschrieben. Auf die Implementierung und die verwendeten Datenstrukturen wird allerdings erst im nächsten Abschnitt eingegangen.

Die Klasse `PndSttCellTrackFinder` stellt Methoden zur Verfügung, die die verschiedenen Phasen des Trackfindings realisieren. Sie bietet Schnittstellen an, über die an die erzeugten Track-Kandidaten gelangt werden kann. Da zu den Tracklets und deren Kombinationen viele Informationen abgespeichert werden müssen (z. B. der Verlauf), werden die Strukturen `TrackletInf_t` und `Combination_t` zur Datenhaltung genutzt, wie es im Klassendiagramm (Abb. 5.1) zu sehen ist. In dem Diagramm werden nur die wichtigsten und zum Trackfinding benötigten Attribute und Methoden dargestellt.

Wie in Abschnitt 5.1.2 beschrieben, wird eine Task benötigt, die das Trackfinding initiiert. Diese Aufgabe wird von der Klasse `PndSttCellTrackFinderTask` übernommen. Während der Abarbeitung der Task wird für jedes Event eine Instanz des `PndSttCellTrackFinder`s erzeugt. Über diese wird an die generierten Track-Kandidaten gelangt. Die Task hat die Aufgabe, die für den Algorithmus benötigten Daten einzulesen. Dabei handelt es sich im konkreten Fall um die STT-Hits. Nachdem die Track-Kandidaten ermittelt worden sind, werden diese in einem ROOT-File gespeichert, sodass sie mit Eve visualisiert werden können und für die Analyse zur Verfügung stehen.

5.2.2 Benötigte Klassen von PandaRoot

PndSttTube

Ein Objekt der Klasse `PndSttTube` repräsentiert eine Straw Tube des STTs. Es enthält Attribute, die z. B. Auskunft über die Lage, Richtung und Tube-ID geben.

PndSttHit

`PndSttHit`-Objekte beschreiben die STT-Hits. Sie werden für jede Straw Tube erzeugt in der ein Signal ausgelöst wurde. Sie enthalten u. a. die Tube-ID und die Driftzeit. Objekte dieser Klasse stellen den Haupt-Input für den Trackfinder dar.

PndSttStrawMap und PndSttGeometryMap

Diese Klassen bieten Funktionen an, um an Daten über die Straw Tubes des STTs anhand der Tube-IDs zu gelangen.

Über ein `PndSttStrawMap`-Objekt werden Informationen über die Lage der Straw Tubes ermittelt. Mit Hilfe der Tube-ID kann z. B. herausgefunden werden, in welchem Sektor und welcher Reihe die zugehörige Straw Tube liegt oder

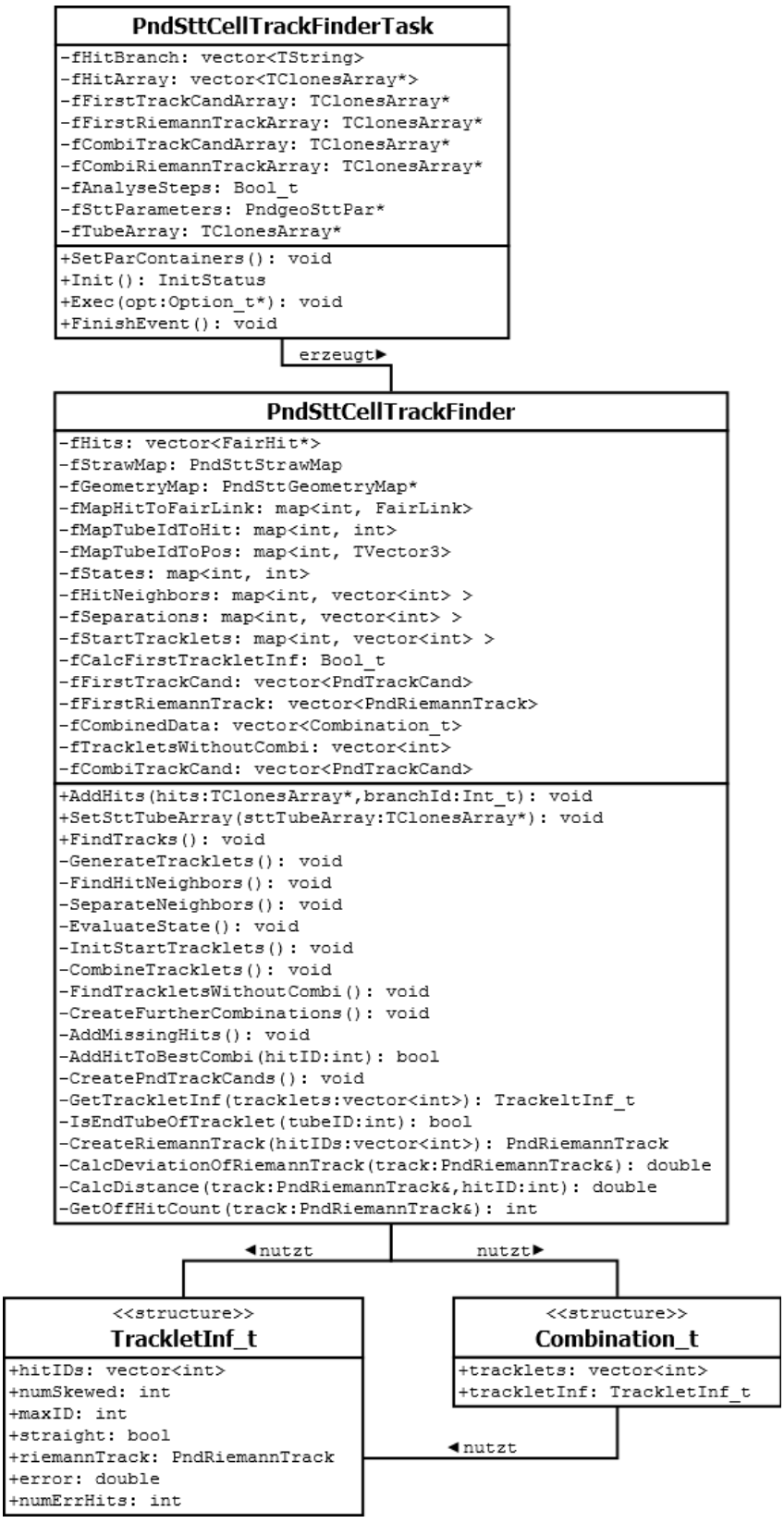

Abbildung 5.1: Klassendiagramm der zusätzlich entwickelten Klassen

ob es sich um ein gedrehtes oder zur Strahl-Achse paralleles Driftröhrchen handelt. Zu einer gegebenen Tube-ID kann das entsprechende `PndSttTube`-Objekt erzeugt werden.

Das `PndSttGeometryMap`-Objekt wird genutzt, um die Nachbarn zu einer Straw Tube zu finden. Zu einer Tube-ID werden über Abstandsberechnungen (auch für gedrehte Straw Tubes) alle Tube-IDs der Nachbarn ermittelt. Für Straw Tubes, die sich an der vertikalen Lücke des STTs befinden, werden keine Nachbarn in der anderen Detektor-Hälfte gesucht.

Sowohl die Klasse `PndSttStrawMap` als auch `PndSttGeometryMap` benötigt zur Initialisierung der Daten ein Array, das an der Stelle i das `PndSttTube`-Objekt mit der Tube-ID i gespeichert hat. Zu diesem Zweck muss ein Objekt der Klasse `PndGeoSttPar` erzeugt werden, das mit Informationen über die Geometrie des STTs (aus einer Runtime-Database) gefüllt wird. Mit diesem Objekt kann wiederum ein `PndSttMapCreator`-Objekt erzeugt werden. Über diese Instanz kann das benötigte Array ermittelt und den Konstruktoren von `PndSttStrawMap` und `PndSttGeometryMap` übergeben werden. Aus diesem Grund finden sich Attribute dieser Typen in dem Klassendiagramm (Abb. 5.1) wieder.

PndTrackCand

Mit dieser Klasse werden Track-Kandidaten beschrieben. Sie stellen eine einfache Liste von Hits dar ohne dass z. B. ein Fitting vorgenommen wurde. Um ein `PndTrackCand`-Objekt zu erzeugen, müssen dem Objekt die gewünschten Hits hinzugefügt werden. Dabei ist ein bestimmter Hit-Typ nicht festgelegt. `PndTrackCand`-Objekte sind der Haupt-Output des Trackfinders.

PndRiemannTrack

Diese Klasse implementiert die in Abschnitt 4.2 beschriebene Kreisapproximation und den fortführenden Helix-Fit. Es werden nur Funktionen genutzt, die die Kreisparameter berechnen. Einem Objekt dieser Klasse müssen `PndRiemannHit`s hinzugefügt werden, die aus `FairHit`-Objekten erzeugt werden können. Ist im Folgenden von einem Riemann-Track die Rede, ist damit die approximierte Kreisbahn durch die Punkte (Hits) dieses Tracks gemeint.

5.3 PndSttCellTrackFinderTask

Der grundlegende Aufbau und die Funktionsweise einer Task-Klasse in PandaRoot wurde im Abschnitt 5.1.2 erläutert. An dieser Stelle wird genauer darauf eingegangen, wie die Funktionen der Klasse `FairTask` in `PndSttCellTrackFinderTask` implementiert worden sind.

5.3.1 Attribute

Attribut	Beschreibung
`vector<TString>` `fHitBranch`	Namen der Zweige aus denen die Hits eingelesen werden sollen (in diesem Fall nur „STTHit“)
`vector<TClonesArray*>` `fHitArray`	Arrays mit den Hits pro Zweig in `fHitBranch`
`TClonesArray*` `fFirstTrackCandArray`	`PndTrackCand`-Objekte, die im ersten Schritt des Verfahrens erzeugt werden, Abspeicherung im Zweig „FirstTrackCand“
`TClonesArray*` `fFirstRiemannTrackArray`	`PndRiemannTrack`-Objekte (falls vorhanden) zu den Objekten in `fFirstTrackCandArray`, Abspeicherung im Zweig „FirstRiemannTrack“
`TClonesArray*` `fCombiTrackCandArray`	endgültige Tracklets, die generiert wurden, Abspeicherung im Zweig „CombiTrackCand“
`TClonesArray*` `fCombiRiemannTrackArray`	`PndRiemannTrack`-Objekte zu allen Tracklet-Kombinationen, Abspeicherung im Zweig „CombiRiemannTrack“
`Bool_t` `fAnalyseSteps`	`true`, wenn die Speicherung der im ersten Schritt erzeugten Daten in der ROOT-Datei erfolgen soll
`PndGeoSttPar*` `fSttParameters`	zur Erzeugung der `PndSttStrawMap`- und `PndSttGeometryMap`-Objekte (siehe Abschnitt 5.2.2)
`TClonesArray*` `fTubeArray`	enthält an der Stelle i das `PndSttTube`-Objekt mit der Tube-ID i, zur Initialisierung der `PndSttStrawMap`- und `PndSttGeometryMap`-Objekte

Tabelle 5.1: Attribute der Klasse `PndSttCellTrackFinderTask`

Die Speicherung der Daten, die vom zellulären Automaten erzeugt werden, in der ROOT-Datei ist für das eigentliche Verfahren nicht notwendig und nicht als Default vorgesehen. Das heißt, `fAnalyseSteps` ist standardmäßig `false`. Die Werte von `fAnalyseSteps` und `fTubeArray` werden an das `PndSttCellTrackFinder`-Objekt weitergegeben.

5.3.2 Funktionen

`virtual void SetParContainers()`
: In dieser Methode wird `fSttParameters` initialisiert.

`virtual InitStatus Init()`
: Diese Methode erzeugt einen IO-Manager von FairRoot und registriert bei diesem die Arrays, die gespeichert werden sollen (`fFirstTrackCandArray`, `fFirstRiemannTrackArray`, `fCombiTrackCandArray`, `fCombiRiemannTrackArray`). `fHitBranch` wird initialisiert und für alle angegebenen Zweige wird `fHitArray` mit der nachfolgenden Methode gefüllt. Es wird der Status der Initialisierung zurückgegeben (Erfolg, Fehler).

`void InitHitArray(TString branchName)`
: Über einen IO-Manager werden die Objekte im Zweig `branchName` eingelesen und in `fHitArray` gespeichert.

`virtual void Exec(Option_t* opt)`
: Diese Funktion erzeugt ein `PndSttCellTrackFinder`-Objekt, setzt die benötigten Parameter und startet das Trackfinding. Die berechneten `PndTrackCand`- und `PndRiemannTrack`-Objekte werden in den dafür vorgesehenen Arrays gespeichert. `Option_t` ist ein `typedef` für `const char` und dient zur Angabe einer Option, welche zur Zeit noch keine Rolle spielt.

`virtual void FinishEvent()`
: In dieser Methode wird der Inhalt der zu speichernden Arrays gelöscht, sodass die Daten für die Events klar abgegrenzt werden.

5.4 Implementierung des Trackfinders

5.4.1 Attribute

Auf die Datenstrukturen in Tabelle 5.2 wird während des gesamten Verfahrens zugegriffen. Sie werden zu Beginn einmalig initialisiert. Über sie kann an alle zum Trackfinding benötigten Daten gelangt werden. In Tabelle 5.3 sind die Attribute aufgelistet, die zur Realisierung des zellulären Automaten dienen. Die letzte Tabelle 5.4 enthält Attribute, welche zum Speichern der Kombinationen genutzt werden.

Attribut	Beschreibung
`vector<FairHit*>` `fHits`	Hits des STTs, Cast zu `PndSttHit*` erforderlich
`PndSttStrawMap` `fStrawMap`	für Lage-Informationen über die Straw Tubes im STT

`PndSttGeometryMap* fGeometryMap`	zur Ermittlung der Nachbarschaftsbeziehungen der Straw Tubes
`map<int, FairLink> fMapHitToFairLink`	ordnet einem Index des Vektors `fHits` den entsprechenden FairLink zu, FairLinks werden zur Erzeugung der Track-Kandidaten (`PndTrackCand`) benötigt (Definition eines FairLinks in Kapitel 6.1)
`map<int, int> fMapTubeIdToHit`	Zuordnung der Hit-Indizes in `fHits` zu den Tube-IDs, erlaubt es mit den Tube-IDs anstelle der Hits zu arbeiten
`map<int, TVector3> fMapTubeIdToPos`	Verknüpfung einer Tube.ID mit der Position (drei-dimensionaler Vektor) der entsprechenden Straw Tube im STT, wird zur Abstandsberechnung benötigt

Tabelle 5.2: Allgemeine Attribute der Klasse `PndSttCellTrackFinder`

Attribut	**Beschreibung**
`map<int, int> fStates`	Zuordnung des Status/der Track-ID zur Tube-ID
`map<int ,vector<int>> fHitNeighbors`	speichert zu jeder Tube-ID aus `fHits` die Tube-IDs der benachbarten aktiven Zellen
`map<int, vector<int>> fSeparations`	zur Speicherung der Tube-IDs in den Gruppen 0 bis 7 entsprechend der Anzahl der aktiven Nachbarn, Gruppe 7 umfasst alle Tube-IDs, die mehr als sechs aktive Nachbarn besitzen
`map<int, TrackletInf_t> fStartTracklets`	Ergebnis des zellulären Automaten, Status eines Tracklets (Track-ID) wird mit der Tracklet-Information abgespeichert (siehe nächster Abschnitt)
`Bool_t fCalcFirstTrackletInf`	`true`, wenn die Track-Kandidaten und Riemann-Tracks des ersten Schrittes abgespeichert werden sollen
`vector<PndTrackCand> fFirstTrackCand`	Track-Kandidaten, die im ersten Schritt erzeugt wurden
`vector<PndRiemannTrack> fFirstRiemannTrack`	Riemann-Tracks (falls vorhanden) zu den im ersten Schritt erzeugten Tracklets

Tabelle 5.3: Attribute der Klasse `PndSttCellTrackFinder`, die vom zellulären Automaten genutzt werden

Attribut	Beschreibung
`vector<Combination_t> fCombinedData`	aus den Start-Tracklets gebildete Kombinationen
`vector<int> fTrackletsWithoutCombi`	Status der Tracklets, die in keiner Zweier-Kombination auftauchen
`vector<PndTrackCand> fCombiTrackCand`	endgültige Track-Kandidaten

Tabelle 5.4: Attribute der Klasse `PndSttCellTrackFinder`, die beim Kombinieren der Tracklets genutzt werden

5.4.2 Strukturen

TrackletInf_t

Für die im ersten Schritt erzeugten Tracklets müssen Informationen über den Verlauf und die Anzahl der gedrehten Straw Tubes vorhanden sein, welche zum Filtern der Kombinationen dienen. Dafür dient die Struktur `TrackletInf_t`. Da diese Daten ebenfalls für Kombinationen (für das Weiter-Kombinieren) ermittelt werden müssen, werden auch Hits gespeichert, die zu den Tracklets gehören. Diese Information liegt eigentlich schon in `fStartTracklets` vor, muss dann aber für die Kombinationen nicht erst durch die Betrachtung der einzelnen Tracklets ermittelt werden. Für die Tracklets können in dieser Struktur der approximierte Riemann-Track und zugehörige Fehlerparameter gespeichert werden. Diese werden beim Hinzufügen der fehlenden Hits im letzten Schritt des Verfahren benötigt. Die Datenstruktur setzt sich aus folgenden Komponenten zusammen:

Attribut	Beschreibung
`vector<int> hitIDs`	Indizes der Hits in `fHits` des (Ausgangs- oder kombinierten) Tracklets
`int numSkewed`	Anzahl der gedrehten Straw Tubes
`int endID`	Tube-ID der abschließenden Straw Tube
`int maxID`	maximale Tube-ID aller Straw Tubes, die das Tracklet enthält
`bool straight`	gibt an, ob das Tracklet strikt vom Inneren des STTs nach außen verläuft
`PndRiemannTrack riemannTrack`	Riemann-Track aus den Hits des Tracklets
`double error`	mittlerer quadratischer Abstand der Mittelpunkte der Straw Tubes des Tracklets vom Riemann-Track

`int numErrHits`	Anzahl der Straw Tubes des Tracklets, die einen Abstand zum Riemann-Track haben, der größer als der Radius einer Straw Tubes ist

Tabelle 5.5: Attribute der Struktur `TrackletInf_t`

Combination_t

Die Struktur verknüpft für eine Tracklet-Kombination folgende Werte:

Attribut	Beschreibung
`vector<int> tracklets`	Status der kombinierten Tracklets
`TrackletInf_t trackletInf`	Tracklet-Information des resultierenden (längeren) Tracklets

Tabelle 5.6: Attribute der Struktur `Combination_t`

5.4.3 Funktionen

Es folgt eine Beschreibung der Implementierung der Methoden. Als Erstes werden die Funktionen erläutert, die zur Initialisierung des Trackfinders aufgerufen werden müssen. Es ist wie in PandaRoot üblich nur ein Standard-Konstruktor vorhanden, weshalb vom Verfahren benötigte Daten explizit gesetzt werden müssen. Der Übersichtlichkeit halber werden dann die Methoden bezüglich der drei wesentlichen Schritte des Verfahrens gruppiert und in ihrer Abarbeitungsreihenfolge aufgelistet. Als Letztes wird auf Operationen eingegangen, die während des Trackfindings wiederholt ausgeführt werden und daher in Methoden ausgelagert worden sind.

Initialisierung

`void AddHits(TClonesArray* hits, Int_t branchId)`
: In dieser Methode wird `fHits` initialisiert. Zudem werden die FairLinks mit Hilfe der Branch-ID, die den Zweig der Speicherung in der ROOT-Datei angibt, erzeugt und in `fMapHitToFairLink` gespeichert.

`void SetSttTubeArray(TClonesArray* sttTubeArray)`
: Hier wird den Objekten `fStrawMap` und `fGeometryMap` das Tube-Array (siehe Abschnitt 5.2.2) übergeben, sodass die benötigten Informationen über die Straw Tubes berechnet werden können. Außerdem wird für jede Straw Tube die Position des Mittelpunktes ermittelt und in `fMapTubeIdToPos` vermerkt.

Generieren der Tracklets

Das Trackfinding wird durch den Aufruf der Methode `void FindTracks()` gestartet. In dieser wird der Aufruf der restlichen Funktionen koordiniert.

`void GenerateTracklets()`
: In dieser Methode wird der Status in `fStates` mit der jeweiligen Tube-ID der Straw Tubes initialisiert. Durch den Aufruf der anschließenden Methoden werden die Start-Tracklets generiert.

`void FindHitNeighbors()`
: Diese Methode initialisiert die Map `fHitNeighbors`. Dafür wird ein Set mit allen Tube-IDs der Hits aus `fHits` angelegt. Dann werden für jedes Element aus `fHits` mit Hilfe des Objektes `fGeometryMap` die Tube-IDs der Nachbarn ermitteln. Wenn die Tube-ID eines Nachbarn in dem Set enthalten ist, wird der Eintrag in `fHitsNeighbors` vermerkt.

`void SeparateNeighbors()`
: Die Funktion füllt `fSeparations`. Sie gruppiert alle Tube-IDs nach der Anzahl ihrer aktiven Nachbarn, die in `fHitNeighbors` gespeichert sind.

`void EvaluateState()`
: Diese Methode implementiert den zellulären Automaten. Es wird ein Set mit allen Tube-IDs erstellt, die ein oder zwei aktive Nachbarn haben. Diese Information wird aus `fSeparations` entnommen. Dann werden die Zustände der Zellen mit ein oder zwei aktiven Nachbarn geändert (also aus `fSeparations[1]` und `fSeparations[2]`). Zur Bestimmung des Minimums werden ausschließlich die Nachbarn aus `fHitNeighbors` herangezogen, die auch in dem zuvor erstellten Set vorhanden sind. Die Zustände werden so lange geändert, bis die vorherige Summe der Track-IDs der aktuell berechneten entspricht. Zur simultanen Änderung der Zustände wird eine zusätzliche Map angelegt, die die berechneten Zustände des aktuellen Schritts zwischenspeichert. Nachdem alle neuen Zustände ermittelt worden sind, wird `fStates` mit der temporären Map überschrieben.

`void InitStartTracklets()`
: Diese Funktion wird nach `EvaluateState()` ausgeführt, um die ermittelten Tracklets zu extrahieren. Sie speichert die Hits der Tube-IDs mit dem gleichen Status (aus `fStates`) in `fStartTracklets`. Dadurch werden die Hits in Bezug auf ihren Status gruppiert. Für die entstandenen Tracklets werden anschließend die Tracklet-Informationen berechnet. Ist `fCalcFirstTrackletInf` auf `true` gesetzt, werden die `PndRiemannTrack`s (falls möglich) mit Fehlerparametern berechnet.

Kombinieren der Tracklets

`void CombineTracklets()`
: In dieser Methode werden jeweils zwei Start-Tracklets aus `fStartTracklets` miteinander kombiniert. Dabei werden die Kombinationsmöglichkeiten wie in Abschnitt 4.3 beschrieben eingeschränkt. Für die Kombinationen werden neue Tracklet-Informationen berechnet, sodass sie als `fCombination_t` in `fCombinedData` abgelegt werden können.

`void FindTrackletsWithoutCombi()`
: Diese Methode sucht nach Start-Tracklets, die nicht in `fCombinedData` auftauchen und vermerkt diese in `fTrackletsWithoutCombi`.

`void CreateFurtherCombinations()`
: Die Funktion sucht in den Zweier-Kombinationen nach möglichen Dreier-Kombinationen und speichert diese zusätzlich in `fCombinedData`, wenn die Kombination als gut befunden wird. Eine Dreier-Kombination ist möglich, wenn dass zweite Tracklet einer Kombination selbst kombiniert worden ist, das heißt, es taucht in einer anderen Kombination als erstes Tracklet in `tracklets` der `Combination_t` auf.

Hinzufügen der fehlenden Hits

`void AddMissingHits()`
: Diese Methode versucht die restlichen Hits zu geeigneten Kombinationen oder nicht kombinierten Start-Tracklets hinzuzufügen. Da dafür die Riemann-Tracks benötigt werden, werden diese für die nicht kombinierten Start-Tracklets berechnet, falls dies noch nicht geschehen ist. Als Erstes wird für alle Hits mit drei oder vier aktiven Nachbarn (also aus `fSeparations[3]` und `fSeparations[4]`) nach einem passenden Tracklet gesucht. Anschließend werden auch die Hits der Start-Tracklets aus `fTrackletsWithoutCombi` betrachtet, die weniger als drei Hits von nicht gedrehten Straw Tubes enthalten. Zur Auswahl einer geeigneten Kombination wird die nächste Methode benutzt.

`bool AddHitToBestCombi(int hitID)`
: Die Funktion liefert `true` zurück, wenn der Hit aus `fHits` an der Stelle `hitID` zu einem (evtl. kombinierten) Tracklet hinzugefügt werden konnte. Wurde ein passendes Tracklet gefunden, wird der Hit-Index (`hitID`) in der Tracklet-Information ergänzt.

Erzeugen der Track-Kandidaten

`void CreatePndTrackCands()`
: In dieser Methode werden `PndTackCand`-Objekte basierend auf den Ein-

trägen von `fCombinedData` und `fTrackletsWithoutCombi` (mit mehr als zwei Hits) erzeugt und in `fCombiTrackCand` gespeichert. Damit ist das Trackfinding abgeschlossen.

Weitere Funktionen

`TrackletInf_t GetTrackletInf(vector<int> tracklets)`
: Diese Methode ermittelt zu Tracklets, die durch den Status in dem Vektor `tracklets` eindeutig bestimmt sind, die Tracklet-Informationen. Ist die Länge von `tracklets` 1, so erfolgt nur die Berechnung des Riemann-Tracks und der Fehlerparameter, da die restlichen Informationen schon in `InitStartTracklets()` ermittelt wurden.

`bool IsEndTubeOfTracklet(int tubeID)`
: Hierbei handelt es sich um eine Hilfsmethode, die beim Generieren der Tracklet-Informationen genutzt wird. Sie gibt `true` zurück, wenn `tubeID` der Tube-ID der abschließenden Straw Tube eines Tracklets entspricht. Dies ist der Fall, wenn die Straw Tube nur einen aktiven Nachbarn hat oder wenn sie einen Nachbarn mit mehr als zwei aktiven Nachbarn besitzt.

`PndRiemannTrack CreateRiemannTrack(vector<int> hitIDs)`
: Die Funktion erzeugt aus einem Vektor von Hit-Indizes ein `PndRiemannTrack`-Objekt.

`double CalcDeviationOfRiemannTrack(PndRiemannTrack& track)`
: Diese Methode ermittelt über Vektorrechnung den mittleren quadratischen Abstand aller Hits (Mittelpunkte der Straw Tubes) von der Kreisbahn des übergebenen Riemann-Tracks.

`double CalcDistance(PndRiemannTrack& track, int hitID)`
: Diese Methode berechnet den Abstand eines Hits mit dem Index `hitID` in `fHits` von der Kreisbahn des Tracks `track`.

`int GetOffHitCount(PndRiemannTrack& track)`
: Die Funktion liefert die Anzahl aller Hits des übergebenen Tracks, die einen größeren Abstand zu seiner Kreisbahn als den Radius einer Straw Tube haben.

6 Bewertung des Verfahrens

Dieses Kapitel analysiert die Qualität der rekonstruierten Tracks des entwickelten Trackfinders. Zur automatischen Analyse wurde die Klasse `PndSttCellTrackFinderAnalyseTask` implementiert, die vor allem Histogramme für diesen Zweck erstellt. Auf die Beschreibung der Implementierung dieser Task wird verzichtet. Stattdessen wird erläutert, welche Rolle die FairLinks bei der Analyse spielen und was für Kriterien überprüft wurden. Anschließend werden die Ergebnisse der Analyse ausgewertet.

6.1 FairLinks

Um die Qualität der erzeugten Track-Kandidaten zu überprüfen, muss eine Verbindung zwischen dem rekonstruierten Track und dem ursprünglichen (physikalischen) Tracks zur Verfügung stehen. Da alle Messwerte mit PandaRoot simuliert worden sind, kann auf die originalen Tracks über die Monte-Carlo-Daten zugegriffen werden, die in der ROOT-Datei vermerkt sind (siehe Abschnitt 3.3.2). Die vollständige Simulation in PandaRoot erfolgt in den aufeinander abfolgenden Schritten: Event-Generation, Propagation, Digitalisierung und Analyse. Jedes dabei generierte Datenobjekt enthält einen oder mehrere FairLinks, die eindeutig auf die Daten zeigen, die zur Erzeugung des Datenobjekts verwendet worden sind.

Ein FairLink enthält Informationen über einen *Zweig* und die *Position* in diesem Zweig. Über die Angabe des Zweiges in Form einer Branch-ID kann die Datenstruktur ermittelt werden, die einem Objekt (in der Simulation/Rekonstruktion) zu Grunde liegt. Die verschiedenen Objekte werden in Form von `TClonesArray`s abgespeichert. Mit der Position wird der genaue Eintrag in dieser Datenstruktur identifiziert. In einem `FairLink`-Objekt sind noch weitere Informationen gespeichert. Diese wurden von der Analyse-Task allerdings nicht benutzt und werden deshalb nicht weiter erläutert.

Zum Beispiel werden aus einem vom Monte-Carlo-Programm berechneten MCTrack, der den physikalischen Track repräsentiert, MCPoints erzeugt. Dies sind die Punkte, an denen die MCTracks die verschiedenen Detektorkomponenten passieren. Basierend auf einem MCPoint werden Detektorsignale simuliert und digitalisiert, aus welchen wiederum Hits erzeugt werden. Während der Track-Rekonstruktion werden mehrere Hits zu Track-Kandidaten (`PndTrack-`

`Cand`) gruppiert. Wird solch ein Track-Kandidat betrachtet, müssen alle zugeordneten Hits auf denselben MCTrack verweisen.

In Abbildung 6.1 sind die Beziehungen zwischen den Daten-Objekten für den Trackfinder des STTs dargestellt. Die FairLinks werden durch Pfeile ausgehend von den Track-Kandidaten, die im Zweig CombiTrackCand gespeichert sind, repräsentiert. Die Zahlen über den verschiedenen Zweigen stehen für die Ebene bzw. die Branch-ID. Die Kästchen unter den Zweigen stellen die einzelnen Einträge in einem `TClonesArray` dar. Die enthaltene Zahl definiert die Position innerhalb des Feldes. Der STTHit-Eintrag (STTHit-Objekt) an der Stelle 0 im `TClonesArray` zeigt z. B. auf den Eintrag der Ebene 1 an der Position 0 (STTPoint-Objekt). Ein Track-Kandidat setzt sich aus mehreren Hits zusammen. Daher gibt es pro Eintrag im `TClonesArray` in CombiTrackKand mehrere Verweise auf die Hits. In dem Beispiel ist der erste Track-Kandidat (Ebene 44, Index 0) ein reiner Track, da alle enthaltenen Hits auf den MCTrack an der Position 0 zurückzuführen sind. Der zweite Track-Kandidat ist fehlerhaft. Ein Hit verweist auf den MCTrack an der Position 0, die anderen Hits auf denjenigen an der Position 1. Um eine Aussage über die Vollständigkeit der Track-Kandidaten zu machen, müssen zusätzlich die FairLinks in die andere Richtung ausgehend von den MCTracks überprüft werden.

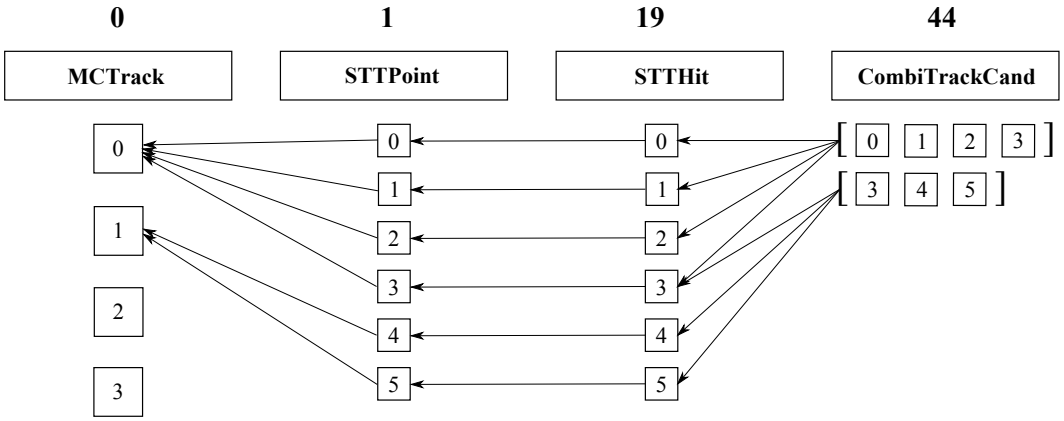

Abbildung 6.1: FairLinks

Zur Auswertung der FairLinks (z. B. von STTHit zu MCTrack) ist es nicht nötig, die FairLinks von Stufe zu Stufe zu betrachten. Es steht eine Klasse namens `PndMCMatch` zur Verfügung. Diese bietet Funktionen an, mit denen alle FairLinks eines Zweiges zu einem anderen ermittelt werden können. Dies kann in eine beliebige Richtung erfolgen - von den simulierten Daten zu den rekonstruierten oder umgekehrt.

6.2 Die Analyse-Task

Die Analyse-Task untersucht sowohl die nach Anwendung des zellulären Automaten entstandenen Tracklets (`FirstTrackCand`s) als auch die endgültigen `CombiTrackCand`s. Die Ausführung kann in dem Makro nach der Abarbeitung der Trackfinder-Task initiiert werden.

Im Folgenden werden die Tracklets auch als Track-Kandidaten angesehen. Es wurden Funktionen implementiert, die Daten ermitteln, welche eine Bewertung der Qualität der erzeugten Track-Kandidaten erlauben. Die Track-Kandidaten wurde u. a. auf ihre *Reinheit* untersucht. Ein reiner Track-Kandidaten enthält nur Hits, die zu ein und demselben physikalischen Track gehören. Dementsprechend verweisen alle FairLinks der zugeordneten Hits auf den gleichen MCTrack.

Zur Analyse wurden folgende Daten erhoben:

- die Anzahl der Verweise auf verschiedene MCTracks pro Track-Kandidat - Hiermit kann die Reinheit der Track-Kandidaten geprüft werden.
- die Anzahl der Hits pro Tracklet - Sie sollte sich durch das Kombinieren erhöhen.
- die Anzahl der vom CA aufgrund von Ambiguitäten ausgeblendeten Hits
- die Anzahl der Tracklets in die ein MCTrack zerfällt - Durch das Kombinieren sollte sich die Anzahl verringern.
- die Anzahl der aufgetretenen Fehler pro Event - Sie gibt Aufschluss darüber, wie sich die unreinen Track-Kandidaten auf die Ereignisse verteilen.
- die Anzahl der Hits des größten Tracklets, das für einen MCTrack gefunden wurde - Sie sollte sich (für unreine Tracklets) durch das Kombinieren erhöhen.
- die Anzahl von gefundenen (in reinen und unreinen Tracklets) und nicht gefundenen MCTracks
- die Anzahl von unvollständigen, vollständigen und fehlerhaften Track-Kandidaten - Ein Track-Kandidat gilt als vollständig, wenn er alle Hits von nicht gedrehten Straw Tubes des MCTracks enthält. Hits von gedrehten Straw Tubes werden nicht mitgezählt, da diese in das Verfahren noch nicht vollständig integriert wurden.

Bei der Auswertung der Daten muss beachtet werden, dass die Anzahl der erzeugten Track-Kandidaten wesentlich höher ist als die der physikalischen Tracks. Es fehlen noch belastbare Kriterien, die die Auswahl einer Kombination

aus mehreren Kombinationsmöglichkeiten für ein Tracklet regelt. Diese können sich erst aus der Hinzunahme von Informationen über Flugbahnen von anderen Detektoren ergeben.

6.3 Testergebnisse

In diesem Abschnitt wird auf die Ergebnisse der Analyse eingegangen. Dabei werden die vom zellulären Automaten gebildeten Track-Kandidaten bewertet und diejenigen, die sich durch das anschließende Kombinieren und Hinzufügen fehlender Hits ergeben. Die angegebenen Werte resultieren aus der Analyse von 5 000 Events in denen 22 243 MCTracks erzeugt wurden, die vom STT wahrgenommen worden sind und dementsprechend rekonstruiert werden müssen.

Es ist schwierig, das Verfahren zu bewerten, da es nur lokale Tracks im STT rekonstruiert. Für einen späteren Einsatz müssen globale Tracks aus den Teilinformationen aller Detektoren ermittelt werden. Erst dann ist eine vollständige Qualitätsanalyse möglich. Folglich können nur Aussagen über die Eigenschaften der Teilstücke von Tracks im STT gemacht werden. Im Folgenden wird vor allem auf die Probleme des CA eingegangen und anschließend wird untersucht, wie sich die weiteren Schritte des Trackfindings auf die Ergebnisse des CA auswirken.

6.3.1 Generieren der Tracklets

Unreine Tracklets

Das Verfahren zur Erzeugung der ersten Tracklets mit Hilfe eines zellulären Automaten betrachtet ausschließlich Hits, die eindeutig einem Track zugewiesen werden können. Ambiguitäten wie Kreuzungen und Verzweigungen werden ausgeblendet, da nicht eindeutig definiert werden kann, wie die Hits zu einem Track gruppiert werden müssen. Dies bedingt die Erwartung, dass als Ergebnis nur reine Tracklets vorliegen können. Die Überprüfung ergab, dass von 46 468 erzeugten Track-Kandidaten 224 Links auf auf mehr als einen MCTrack beinhalten und somit fehlerhaft sind. Dies entspricht ca. 0.48 %. Diese unreinen Tracklets verteilen sich auf 220 Events, das heißt, für 95.6 % der Events arbeitet der zelluläre Automat ohne Fehler. Im ersten Schritt begangene Fehler können nicht von den sich anschließenden Schritten behoben werden. Sie werden durch den gesamten Trackfinding-Algorithmus mitgeführt. Daher ist ein solch geringer Anteil an unreinen Tracklets durchaus erforderlich.

In den nachfolgenden Abbildungen sind unreine Tracklets verschiedener Ereignisse zu sehen. Diese zeigen, dass besondere Formationen der Hits bei der Entwicklung des Algorithmus nicht bedacht worden sind.

In Abbildung 6.2 sind zwei Tracks zu sehen, die nebeneinander in den STT eintreffen, ein Signal bei zwei benachbarten Straw Tubes in der ersten Reihe

auslösen und sich dann auseinander bewegen. Jede Straw Tube des Tracklets hat nur zwei aktive Nachbarn, was zur Erzeugung eines unreinen Tracklets führt. Dieser Fall kann auch eintreten, wenn zwei Tracks den STT über zwei benachbarte Straw Tubes verlassen und ihre Krümmung bis dahin entgegengesetzt verlief. Es wurde ein Event unter 5 000 gefunden, in dem ein Tracklet generiert wird, das Verweise auf drei MCTracks besitzt (siehe Abbildung 6.3). Auch hier treffen die Tracks nebeneinander in den STT ein. Zwei von ihnen verlaufen fast parallel zur Strahl-Achse durch den STT, sodass ein Track nur ein Signal und der andere zwei auslöst.

Demzufolge kann das Generieren von Tracklets Probleme bereiten, wenn Tracks in den STT in benachbarten Straw Tubes eintreffen oder den STT durch diese verlassen.

Abbildung 6.4 stellt einen anderen Fehlerfall dar. Alle Hits, die im Ausschnitt der rechten Detektor-Hälfte zu sehen sind, werden zu einem Tracklet gruppiert. Sie gehören aber zu zwei verschiedenen MCTracks. Die vertikale Lücke des STTs sorgt dafür, dass keine Ambiguitäten auftauchen.

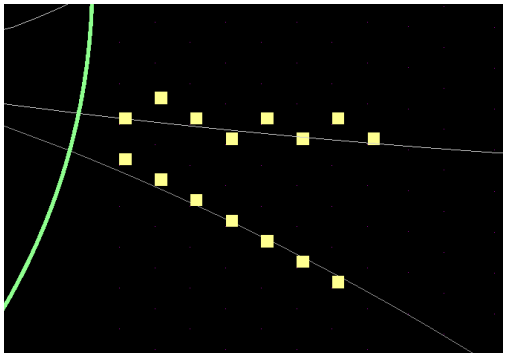

Abbildung 6.2: Vom CA erzeugtes Tracklet mit Links auf 2 MCTracks

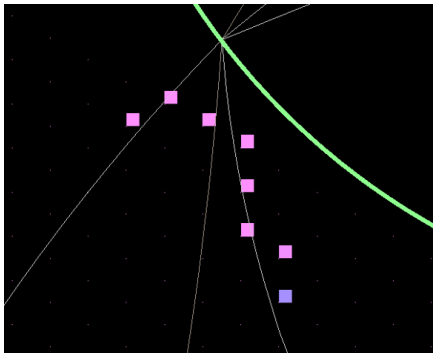

Abbildung 6.3: Tracklet das Hits von 3 MCTracks enthält

In die 224 unreinen Tracklets fließen allerdings auch Tracklets mit ein, die eigentlich als „richtig generiert“ bewertet werden müssten. Es kann passieren, dass ein Teilchen zerfällt und eines der resultierenden Teilchen scheinbar die vorherige Flugbahn „fortsetzt“, während das zweite Teilchen nicht nachgewiesen wird. Aus den STT-Hits geht dieser Vorgang nicht hervor, wenn sich der Impuls nicht stark ändert. Die Hits gehören offensichtlich zu einem Track, verweisen aber auf verschiedene MCTracks, da es sich um die Flugbahnen verschiedener Teilchen handelt. Ein solches Beispiel ist in Abbildung 6.5 zu sehen. Ein geladenes Pion (grau) zerfällt in ein Myon (blau) und Neutrino. Die Flugbahn des Neutrinos ist nicht zu sehen. Der graue und der blaue Track haben unterschiedliche MCTrack-Indizes, wodurch das Tracklet als unrein bewertet wird. Anhand der STT-Signale kann aber nicht erkannt werden, dass ein Zerfall stattgefunden hat, sodass das Tracklet, wie es gebildet wurde, erwünscht ist.

Aufgrund derartiger Ereignisse ist die Anzahl unreiner Tracklets eigentlich geringer als 224.

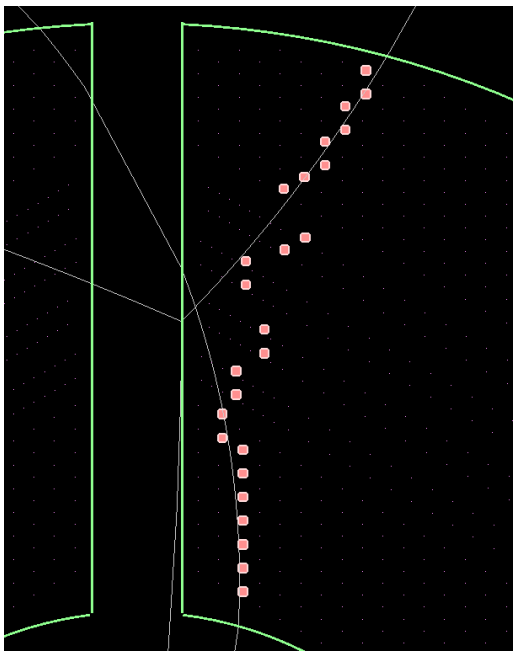

Abbildung 6.4: Beispiel für ein weiteres unreines Tracklet

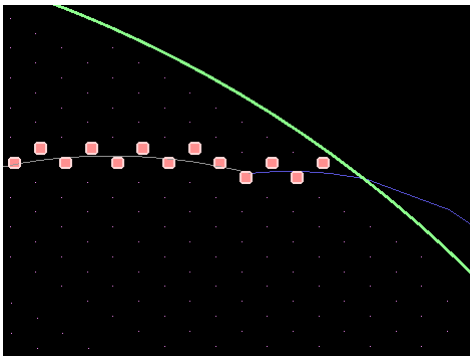

Abbildung 6.5: Als falsch bewertetes Tracklet, das richtig generiert wurde

Anzahl der Hits der Tracklets

Die Anzahl der Hits, die einem Tracklet zugewiesen werden, kann sehr stark variieren. Ein MCTrack besitzt im Durchschnitt 21 Hits, ein Tracklet dagegen nur 7-8 Hits. Über ein Fünftel der erzeugten Tracklets besitzen allerdings nur einen Hit. Dies ist vor allem auf stark gekrümmte Tracks zurückzuführen und Ereignisse bei denen es zu vielen Überschneidungen in der Projektion kommt. Ein Beispiel dafür ist in Abbildung 6.6 zu sehen. Viele der gebildeten Tracklets enthalten nur ein bis zwei Hits. Ein großer Teil der Hits wird gar nicht betrachtet. Die Rekonstruktion solcher Tracks ist eine Schwachstelle des Verfahrens. Die durchschnittliche Anzahl von 7-8 Hits pro Tracklet ist relativ hoch, da 20 % der Tracklets nur ein Hit zugewiesen wurde. Es gibt viele Tracklets, die wesentlich mehr als 8 Hits besitzen. So werden vom CA trotz dieser Schwachstelle ein Drittel der MCTracks vollständig rekonstruiert.

Bei den vollständig rekonstruierten Tracks handelt es sich um „normale“ Tracks, die vom Inneren des STTs nach außen ohne Überschneidungen mit anderen Tracks verlaufen. In Abbildung 6.7 ist ein Track zu sehen, der diese Eigenschaften besitzt und trotzdem in zwei Tracklets zerfällt, weil vier Hits der gedrehten Straw Tubes ausgeblendet werden. Das liegt daran, dass der

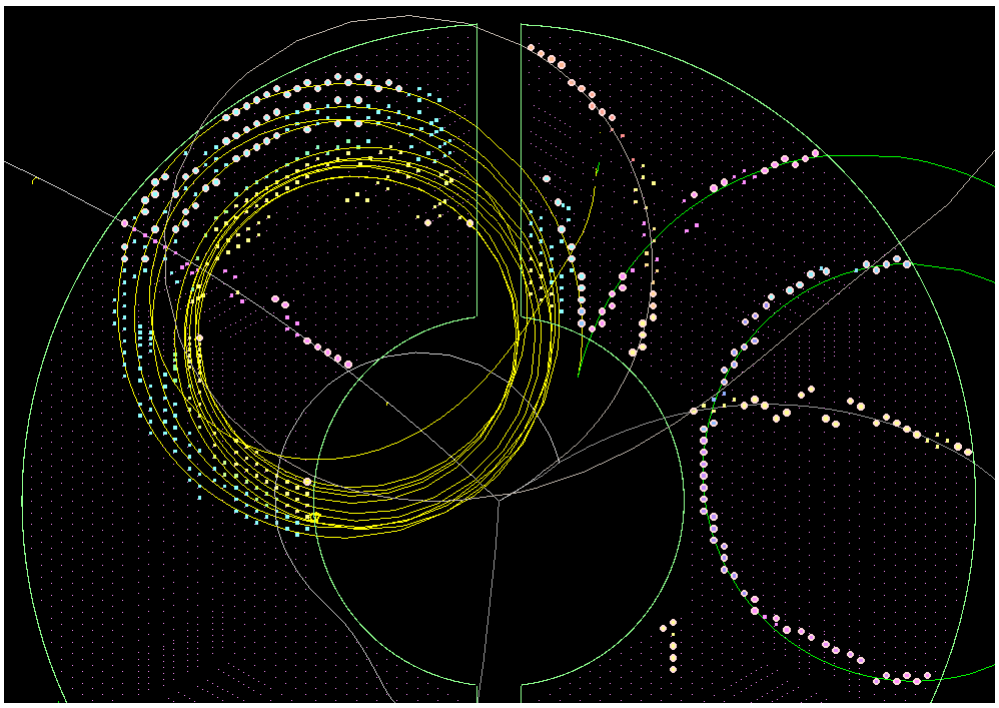

Abbildung 6.6: Event für das der CA 47 Track-Kandidaten für 5 MCTracks generiert

Track zwei gedrehte (hintereinanderliegende) Straw Tubes in einer Reihe passiert und mehr als zwei aktive Nachbarn entstehen. Das zeigt, dass der CA nicht die vollständige Rekonstruktion „normaler“ Tracks garantiert. Im Regelfall ergeben sich aber wenige und große Tracklets, die die Mehrheit der Hits derartiger Tracks umfassen.

Die Analyse ergab, dass in den 5 000 Events ca. 530 000 STT-Hits erzeugt wurden. Von diesen waren ungefähr 65 % von der Evaluierung der Zustände im CA betroffen. Das heißt, dass mehr als ein Drittel der Hits aufgrund von Ambiguitäten von dem Verfahren nicht in Betracht gezogen wurde. Dies zeigt, dass sich zwingend weitere Schritte anschließen müssen, die die unangetasteten Hits in die Tracklets integrieren.

Nicht gefundene MCTracks

Die vom CA generierten Tracklets werden alle unabhängig von der Anzahl ihrer Hits in Track-Kandidaten überführt. MCTracks, auf die keine Tracklets verweisen, müssen sich ausschließlich aus Hits mit mehr als zwei aktiven Nachbarn zusammensetzen. In Abbildung 6.8 ist ein Event zu sehen, für das acht MCTracks nicht gefunden werden. Dies ist auf den Sekundärvertex zurückzuführen, der eine „Wolke“ von STT-Hits nach sich zieht. Es kommt hinzu, dass viele der Tracks den STT früh verlassen, sodass keine Wegstücke

gefunden werden können, deren Hits eindeutig zugeordnet werden können. Insgesamt werden von den 22 243 MCTracks 1 418 nicht gefunden, was einem Prozentsatz von 6.38 % entspricht.

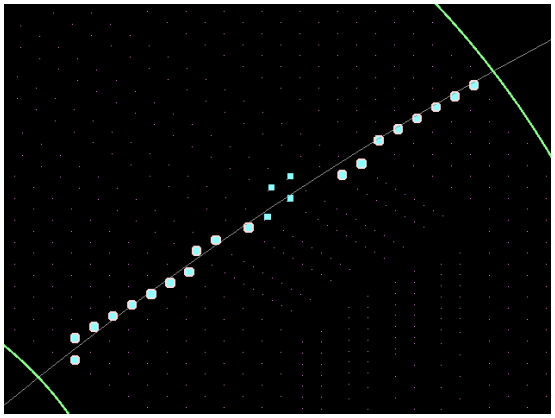

Abbildung 6.7: Normaler Track, der in 2 Tracklets zerfällt

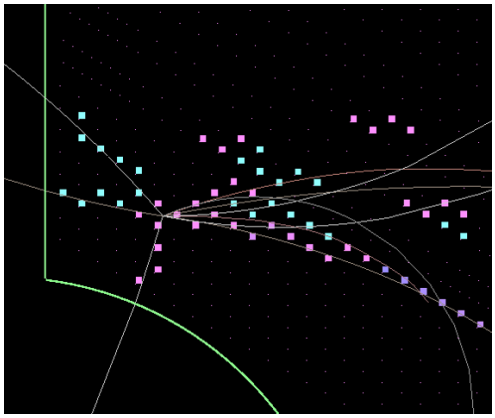

Abbildung 6.8: Event bei dem 8 MCTracks nicht gefunden werden.

6.3.2 Zusammenfassung

Qualität der Rekonstruktion

Die Tabelle 6.1 zeigt, wie viele der insgesamt 22 243 MCTracks durch die erzeugten Track-Kandidaten gefunden bzw. nicht gefunden worden sind. Unter „FirstTrackCand“ sind alle vom CA generierten Track-Kandidaten/Tracklets zu verstehen, während sich „CombiTrackCand“ auf die vom Trackfinder endgültig erzeugten Track-Kandidaten bezieht. Die Eigenschaft „In reinen TK gefunden“ bedeutet, dass ein Teil der Spurpunkte eines einzigen MCTracks in einem Track-Kandidaten(TK) gefunden worden ist. „Vollständig rekonstruiert“ steht für MCTracks, von denen alle Hits in einem Track-Kandidaten gefunden worden sind. Die Anzahl vollständig rekonstruierter MCTracks ist eine Teilmenge der „In reinen TK gefundenen“ MCTracks. „In unreinen TK gefunden“ besagt, dass ein Teil der Spurpunkte mehrerer MCTracks in einem Kandidaten gefunden wurde. Dabei handelt es sich demzufolge um einen falsch generierten Kandidaten. „Nicht gefunden“ bedeutet, dass kein Hit des MCTracks einem Track-Kandidaten zugeordnet wurde.

Die Anzahl der MCTracks, die in unreinen Track-Kandidaten gefunden werden ist nach dem Kombinieren höher. Das bedeutet, dass Tracklets falsch miteinander kombiniert werden. Nach der Anwendung des CA werden alle Tracklets in Track-Kandidaten überführt. Es werden auch Kandidaten aus Tracklets gebildet, die sich aus weniger als drei Hits zusammensetzen. Dies ist nach dem Kombinieren nicht mehr der Fall. Kandidaten mit nur einem oder zwei Hit/s

Eigenschaft	Anzahl der MCTracks in FirstTrackCand	Anzahl der MCTracks in CombiTrackCand
In reinen TK gefunden	20 640 ≙ 92.79 %	18 542 ≙ 83.36 %
Davon vollständig rekonstruiert	7 551 ≙ 33.95 %	9 056 ≙ 40.71 %
In unreinen TK gefunden	185 ≙ 0.83 %	1 142 ≙ 5.13 %
Nicht gefunden	1 418 ≙ 6.38 %	2 559 ≙ 11.51 %

Tabelle 6.1: Ergebnisse der Suche von MCTracks in den Track-Kandidaten (TK)

werden verworfen. Dies führt dazu, dass weniger MCTracks gefunden werden. Die Anzahl vollständig rekonstruierter Tracks steigt - allerdings nur um ca. 8 %. Daher sollten die Methoden zum Kombinieren und Hinzufügen der nicht betrachteten Hits noch verbessert werden. Außerdem muss beachtet werden, dass es sich nur bei den vom CA vollständig rekonstruierten Track-Kandidaten um diejenigen handelt, die auch wirklich alle Hits des physikalischen Tracks besitzen. Nur in diesem Schritt werden Hits gedrehter Straw Tubes einbezogen. Ein Track-Kandidat wird von der Analyse-Task als „vollständig “ bewertet, wenn er alle Hits von nicht gedrehten Straw Tubes des MCTracks umfasst.

Wie in Tabelle 6.2 zu sehen, ist die Anzahl der generierten Track-Kandidaten wesentlich höher als die der MCTracks. Vom CA werden ca. doppelt so viele Kandidaten erzeugt als vorhanden sein müssten. Durch das Kombinieren verringert sich die Anzahl deutlich. Da bei mehreren guten Kombinationsmöglichkeiten noch keine verworfen wird, werden aber auch hier mehr Kandidaten gebildet. Nach dem Kombinieren ergeben sich prozentual weniger unvollständige Tracklets. Falsche Kombinationen führen allerdings zu mehr unreinen Tracklets. Der Anteil vollständig rekonstruierter Tracks steigt.

Die Zahl der Events, in denen Fehler beim Rekonstruieren gemacht werden, steigt von 220 auf 965. Das heißt aber nicht, dass letztendlich für 965 Ereignisse die Rekonstruktion falsch ist. Sobald für ein Event ein Tracklet falsch kombiniert worden ist, wird die Anzahl der Fehler erhöht. Allerdings kann das Tracklet auch noch in anderen Kombinationen auftauchen, die dem physikalischen Track entsprechen.

Laufzeitverhalten

Zur Untersuchung der Rechenzeit, die das Trackfinding-Verfahren benötigt, wurde ein Rekonstruktions-Makro mit der `PndSttCellTrackFinderTask` aus-

	Anzahl der TK in FirstTrackCand	Anzahl der TK in CombiTrackCand
Gesamt	46 468 ≙ 208.91 % der Anzahl an MCTracks	25 174 ≙ 113.18 % der Anzahl an MCTracks
Rein, aber fehlende Hits	38 693 ≙ 83.27 %	14 528 ≙ 57.71 %
Rein und vollständig	7551 ≙ 16.25 %	9 056 ≙ 35.97 %
Unrein	224 ≙ 0.48 %	1 590 ≙ 6.32 %

Tabelle 6.2: Ergebnisse der Untersuchung der Track-Kandidaten (TK) des CA und der endgültig generierten TK

geführt. Um die Zeit der verschiedenen Stufen des Verfahrens zu analysieren, wurde ihre Abarbeitung bzw. Ausführung schrittweise ergänzt. Der Trackfinder benötigt für 5 000 Events 51.61 s. Dies entspricht einer Rechenzeit von ca. 0.01 s pro Event. Die Daten beziehen sich auf einen Rechner mit Intel Core i7-3770 Prozessor. In Tabelle 6.3 ist zu sehen, wie sich die CPU-Zeit auf die drei Stufen des Verfahrens aufteilt. Der zelluläre Automat beansprucht über 93 % der gesamten Ausführungszeit des Trackfinders. Durch die starke Einschränkung der Kombinationsmöglichkeiten erfordert das Kombinieren nur einen Bruchteil der Gesamtzeit. Das Hinzufügen der verbleibenden Hits fällt etwas stärker ins Gewicht, aber benötigt lange nicht so viel Zeit wie die Ausführung des CA.

Schritt	CPU-Zeit
Zellulärer Automat (CA)	48.06 s
Bilden der Kombinationen	0.95 s
Hinzufügen fehlender Hits	2.6 s

Tabelle 6.3: CPU-Zeiten der drei wesentlichen Schritte des Trackfinders

Die Tabelle 6.4 zeigt, wie sich die vom CA benötigte Zeit aufgliedert. Dabei nimmt die Initialisierung der Nachbarschaftsbeziehungen einen hohen Stellenwert ein. Zu jeder Straw Tube, die ein Signal gemeldet hat, werden die Tube-IDs der angrenzenden Straw Tubes berechnet. Diese Berechnung erfolgt für jedes Event und ist daher sehr zeitintensiv. Da es sich hier um Initialisierungsvorgänge handelt, können die Berechnungen aus dem eigentlichen Verfahren ausgelagert werden. Zudem sind die Nachbarschaften der Straw Tubes für jedes Event gleich und müssten nur einmalig berechnet und abgespeichert werden. Dies würde die Laufzeit des gesamten Verfahrens auf 15.74 s (0.003 s/Event)

verkürzen. Zur Berechnung der Nachbarschaftsbeziehungen wurde eine bereits existierende Klasse von PandaRoot genutzt, weshalb auf diese Aspekte kein Einfluss genommen werden konnte.

Die Anpassung der Zustände benötigt ca. 20 % der Rechenzeit des CA. Sie ist allerdings hochgradig parallelisierbar.

Da das Verfahren noch parallelisiert und auf einer GPU eingesetzt werden soll, lassen sich keine Aussagen über die endgültige Rechenzeit des Programms machen. Der Trackfinding-Algorithmus bietet eine gute Grundlage für eine schnelle Software, da viele Operationen nebenläufig ausgeführt werden können.

Schritt des CA	CPU-Zeit
Initialisieren der Nachbarschafts-beziehungen	35.87 s
Gruppierung der Zellen nach Anzahl ihrer aktiven Nachbarn	0.54 s
Anpassung der Status	9.75 s
Initialisieren der Start-Tracklets mit Tracklet-Informationen	1.9 s

Tabelle 6.4: Aufteilung der 48.06 s Rechenzeit des CA auf seine Teil-Schritte

7 Fazit und Ausblick

Es wurde ein Trackfinding-Verfahren entwickelt, das Signale (Hits) des Straw Tube Trackers (STT) des $\overline{\text{P}}$ANDA-Detektors zu möglichen Track-Kandidaten gruppiert und das gut parallelisierbar ist. Es wurde in C++implementiert und in das Simulations-Framework PandaRoot eingebunden. Im Fokus des Verfahrens steht das Finden von „normalen“ Teilchen-Spuren, die sich vom Inneren des STTs nach außen bewegen. Die Rekonstruktion von stark gekrümmten Flugbahnen wird vernachlässigt. Die Signale von gedrehten Straw Tubes wurden nicht vollständig in das Verfahren integriert. Das Verfahren betrachtet den Querschnitt durch den STT und somit nur die x-y-Projektion der Flugbahnen in den zwei-dimensionalen Raum. Die Hits von gedrehten Straw Tubes liefern Informationen über die z-Komponente der Flugbahn. Eine Berechnung der x- und y-Werte anhand der z-Komponente steht aus.

Der Trackfinding-Algorithmus lässt sich in drei Stufen unterteilen. In der ersten Stufe wird das Prinzip eines zellulären Automaten (CA) genutzt, um erste Wegstücke der Flugbahnen zu erzeugen. Der CA kann nur für normale Tracks ohne Kreuzungen zur vollständigen Rekonstruktion führen. Aus diesem Grund werden die Wegstücke in der zweiten Stufe miteinander kombiniert. Die Kombinationen werden so gefiltert, dass sich nur physikalisch sinnvolle Tracks ergeben können. Da in der x-y-Projektion Flugbahnen in Form von Kreisen zu erwarten sind, wird für die Hits der Kombinationen eine Kreisapproximation durch die Mittelpunkte der entsprechenden Straw Tubes durchgeführt. Dazu wird der Riemann-Fit eingesetzt, der das quadratische Problem in ein lineares überführt. Zur Bewertung der Kombination werden die Abstände der Mittelpunkte der Straw Tubes zur angenäherten Kreisbahn betrachtet. Im letzten Schritt werden Signale, die in der ersten (und damit auch zweiten) Stufe nicht beachtet worden sind, zu geeigneten Track-Kandidaten hinzugefügt, falls solche vorhanden sind. Hits gedrehter Straw Tubes werden nur vom CA betrachtet, da für die anschließenden Schritte u. a. Abstandsberechnungen vorgenommen werden und die x- und y-Komponenten noch geeignet ermittelt werden müssen.

Das entwickelte Verfahren liefert für über 30 % der physikalischen Tracks eine vollständige Rekonstruktion. Bei Vernachlässigung der Hits von gedrehten Straw Tubes werden 40 % der ursprünglichen Tracks komplett erkannt. Der Algorithmus kann mit Kreuzungen von Tracks umgehen. Es ist aber nicht garantiert, dass alle Wegstücke (Tracklets) des CA korrekt zusammengefügt werden. Einige unvollständige Start-Tracklets bleiben als solche bestehen oder werden sogar falsch kombiniert. Gibt es mehrere gute Kombinationsmöglichkeiten,

werden mehrere Track-Kandidaten erzeugt. Aus diesem Grund ist die Anzahl der ermittelten Track-Kandidaten in der Regel wesentlich größer als die Anzahl der physikalischen Tracks.

Der CA führt bereits im ersten Schritt zu einer großen Anzahl an vollständig rekonstruierten Tracks. Es gibt nur wenige Sonderfälle, für die der zelluläre Automat unreine Track-Kandidaten erzeugt. Das Kombinieren bereitet noch Probleme, da der Riemann-Fit für viele Tracklets nicht die gewünschten großen Kreisbahnen berechnet und daher keine vergleichbaren Kreisparamter anbietet. Hauptsächlich die Berechnung der Nachbarschaftsbeziehungen, die Voraussetzung für die Anwendung des CA ist, wirkt sich negativ auf das Laufzeitverhalten des Verfahrens aus. Die einzelnen Stufe des Verfahrens bieten jedoch gute Voraussetzungen für die Parallelisierung des Trackfinders.

Damit der Trackfinder in Zukunft zur Auswertung der Signale des STTs genutzt werden kann, sollte sich eine Verbesserung und Weiterentwicklung des Verfahrens anschließen. Folgende Aspekte werden dabei eine wichtige Rolle spielen:

- Um die Parallelisierbarkeit auszunutzen, sollte das entwickelte Programm auf eine GPU übertragen werden.
- Bisher kann das Programm nicht mit einem kontinuierlichen Datenstrom an Hits als Eingabe arbeiten. Dies wird allerdings durch den späteren Einsatz in $\overline{\text{P}}$ANDA bedingt und muss noch umgesetzt werden.
- Die Signale gedrehter Straw Tubes müssen durch Einbeziehung der z-Komponente in das Verfahren integriert werden. Dies könnte sich positiv auf die Approximation der Kreisbahnen auswirken, da mehr Punkte in den Fit einfließen würden.
- Der Riemann-Fit könnte durch das Definieren von Randbedingungen angepasst werden, sodass für die vom CA generierten Wegstücke vergleichbare Kreisparameter berechnet werden können. Dies würde ein unnötiges Kombinieren ersparen und die Kreisapproximationen müssten nur einmalig für die Tracklets ermittelt werden, was die Rechenzeit des Programms verringern würde.
- Zum Kombinieren der Tracklets können alternative Verfahren zur Berechnung der Kreisapproximation erprobt werden. Diese sollten aber auch einen linearen und nicht iterativen Lösungsansatz bieten.

Literaturverzeichnis

[1] *Physics Performance Report for: PANDA - Strong Interaction Studies with Antiprotons*. PANDA Collaboration.

[2] *Straw Tube Tracker Technical Design Report*. PANDA Collaboration, 2012.

[3] PANDA Collaboration. `http://www-panda.gsi.de/framework/content/detector/img/panda_full_label2_1.jpg`. [aufgerufen am 23.06.2013].

[4] GSI Helmholtzzentrum für Schwerionenforschung GmbH. `https://www.gsi.de/forschungbeschleuniger.htm`. [aufgerufen am 22.06.2013].

[5] R. Frühwirth, A. Strandlie, and W. Waltenberger. Helix fitting by an extended riemann fit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 490(1–2):366 – 378, 2002.

[6] Martin J. Galuska. `http://panda-wiki.gsi.de/cgi-bin/view/Computing/PandaRootCodingRules`. [aufgerufen am 31.07.2013].

[7] M. Gardner. The fantastic combinations of john conway's new solitaire game "life". *Scientific American*, 223:120–123, 1970.

[8] FAIR GmbH. `https://www.gsi.de/uploads/pics/fair-topologie.jpg`. [aufgerufen am 23.06.2013].

[9] FAIR GmbH. `http://www.fair-center.de/de/oeffentlichkeit.html`. [aufgerufen am 22.06.2013].

[10] A.G. Hoekstra, J. Kroc, and P.M.A Sloot. *Simulating Complex Systems by Cellular Automata*. 2010.

[11] Forschungszentrum Juelich GmbH IKP. `http://www.fz-juelich.de/ikp/DE/Forschung/ExperimentelleEntwicklungen/DriftrohrKammern/Bilder/straws_001.jpg;jsessionid=DD8B4C8F8BBC0F5F8C9B74DB4F809CC8?__blob=poster`. [aufgerufen am 23.06.2013].

[12] Stefano Spataro. Simulation and event reconstruction inside the pandaroot framework. *Journal of Physics: Conference Series*, 119, 2008.

[13] A. Strandlie, J. Wroldsen, and R. Frühwirth. Treatment of multiple scattering with the generalized riemann sphere track fit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 488(1–2):332 – 341, 2002.

[14] A. Strandlie, J. Wroldsen, R. Frühwirth, and B. Lillekjendlie. Particle tracks fitted on the riemann sphere. *Computer Physics Communications*, 131(1–2):95 – 108, 2000.

[15] The ROOT Team. `http://root.cern.ch/drupal/content/discovering-root`. [aufgerufen am 03.07.2013].

[16] The ROOT Team. `http://root.cern.ch/drupal/content/virtual-monte-carlo`. [aufgerufen am 03.07.2013].

[17] The ROOT Team. `http://root.cern.ch/drupal/content/eve`. [aufgerufen am 06.07.2013].

[18] The ROOT Team. `http://root.cern.ch/drupal/content/users-guide`. [aufgerufen am 31.07.2013].

[19] Foto von Artur Cebulla (Forschungszentrum Juelich GmbH IKP).

[20] Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55, 1983.