Rekonstruktion von Ladungsdepositionen in einer GEM-Pixel-TPC mit einem Algorithmus aus der Astronomie

Alexander Deisting

Bachelorarbeit in Physik
angefertigt im Physikalischen Institut

vorgelegt der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität
Bonn

im Februar 2012

Referent: Prof. Dr. Klaus Desch
Koreferent: Dr. Thomas Erben

Ein Dankeschön an:

D. Klaes und Dr. T. Erben für Hilfreiches zu Source Extractor,

An die Mitglieder der Arbeitsgruppe Desch, die mit ihrem Rat und Antworten zum Gelingen dieser Arbeit beigetragen haben; insbesondere an Prof. Desch, an Dr. Kaminski und an die Besatzung des Highest Energy Office,

An alle Korrekturleser,

An den Betreuer dieser Arbeit, C. Brezina

und an Alle die dazu beigetragen haben das ich überhaupt soweit gekommen bin.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Frage nach dem, was die Welt im Innersten zusammenhält, treibt schon Generationen von Physikern zu immer neuen Entwicklungen an. So haben wir heute mit dem LHC und seinen Experimenten ein Werkzeug an der Hand, mit dem sich bald die gängigen Theorien bestätigen, oder verwerfen lassen werden. Danach wird es an zukünftigen Beschleunigern sein, die Entdeckungen des LHC weiter zu vermessen, oder bestimmte Energiebereiche von Interesse genauer zu untersuchen als dies mit dem LHC möglich ist. Beschleuniger, die dazu eingesetzt werden können, sind der International Linear Collider (ILC) oder der Compact Linear Collider (CLIC).

Von der Entwicklung neuer Detektoren im Rahmen der eben beschriebenen Beschleuniger verspricht man sich vor allem bessere Auflösungen der Teilchenspuren und somit höhere Impulsauflösungen. Um solche Spuren zu detektieren, nutzt man Spurdetektoren wie Zeitprojektionskammern oder Pixel-Detektoren. Letztere sind Halbleiterdetektoren, die eine hohe Energieauflösung und, auf Grund ihrer Strukturierung, eine hohe zweidimensionale Ortsauflösung besitzen. Große Halbleiterdetektoren wie der Pixel Detector und der Semi-Conductor Tracker am ATLAS-Experiment sind aus mehreren Lagen von Pixel-Detektoren oder Silizium-Streifen-Detektoren aufgebaut, um so die dreidimensionale Auflösung einer Teilchenspur durch die Kombination mehrerer Lagen aktiver Elemente zu erreichen. Durch die bereits erwähnte hohe zweidimensionale Auflösung, haben diese Halbleiterdetektoren eine sehr gute Auflösung der einzelnen Spurpunkte, allerdings ist die Anzahl der Spurpunkte pro Spurlänge begrenzt.
Um mehr Spurpunkte und somit eine bessere Auflösung einer Teilchenspur zu erreichen, bieten sich die zuerst erwähnten Zeitprojektionskammern an. Zeitprojektionskammern basieren auf dem Prinzip einer Ionisationskammer, sodass beim Auslesen des Detektorsignals im Idealfall jede Ionisation zu einem Spurpunkt führt. Eingeschränkt wird dies durch die Tatsache, dass sich einzelne Elektronen nicht detektieren lassen und es so erforderlich ist, die durch Ionisation entstandenen Elektronen mittels der Erzeugung von Elektronenlawinen im Gas zu verstärken. Dieser Prozess zieht eine zweidimensionale Verbreiterung des Signales nach sich und führt dazu, dass sich die Signale verschiedener Ionisationspunkte überlappen können. Um eine möglichst genaue Rekonstruktion der Spur zu ermöglichen ist es notwendig, die überlappenden Signale - als Cluster bezeichnete Ladungsdepositionen auf einem Auslesemedium - zu trennen und ihre Zentren zu ermitteln. Diese Zentren lassen sich mit der Position eines primären Elektrons identifizieren. Somit erlaubt die Trennung der Cluster in möglichst viele Hits die Ermittlung einer große Anzahl an Spurpunkten und man erhält eine exzellente Spurauflösung.

Im Rahmen dieser Bachelorarbeit wurde erstmals das in der Astronomie verwendete Programm Source Extractor für die Trennung solcher überlappenden Signale genutzt, und verglichen, ob sich mit diesem Programm mehr Spurpunkte rekonstruieren lassen als mit dem aktuell verwendeten Trennalgorithmus.
In Kapitel 2 wird zunächst auf die physikalischen Grundlagen eingegangen werden, denen die später zu analysierenden Daten zu Grunde liegen. Anschließend wird ein Überblick über Source Extractor (oder auch SExtractor), sowie die Funktionsweise des Programms mit den

verschiedenen Eingabeparameter gegeben (Kapitel 3) um im Folgenden die Anwendung von SExtractor auf Messdaten einer Zeitprojektionskammer beschreiben zu können. Bei dieser Beschreibung (Kapitel 4) wird das Hauptaugenmerk dann auf die Auswirkungen verschiedener Werte der Eingabeparameter auf die Separation von Clustern, und die Suche nach den optimalen Parametern gerichtet. Mit den gefundenen Parametern werden zum Abschluss größere Datensätze und simulierte Daten analysiert. Die mit Source Extractor erzielten Ergebnisse werden mit den Ergebnissen des bisher verwendeten Clustertrenners in Kapitel 5 verglichen um eine Aussage darüber treffen zu können, ob sich mit SExtractor bei der Analyse von Messergebnissen bessere Resultate als mit dem alten Trennalgorithmus erzielen lassen.

Kapitel 2

Zeitprojektionskammern

Im folgenden Kapitel wird zunächst das Konzept einer Zeitprojektionskammer (engl. **T**ime **P**rojection **C**hamber (TPC)) vorgestellt und danach der zur Datennahme genutzte TPC-Prototyp beschrieben. Zudem wird der Analyseprozess, den diese Daten durchlaufen, erläutert und insbesondere der bisherige Clustertrennalgorithmus erörtert.

2.1 Funktion und Detektionsprinzip einer Zeitprojektionskammer

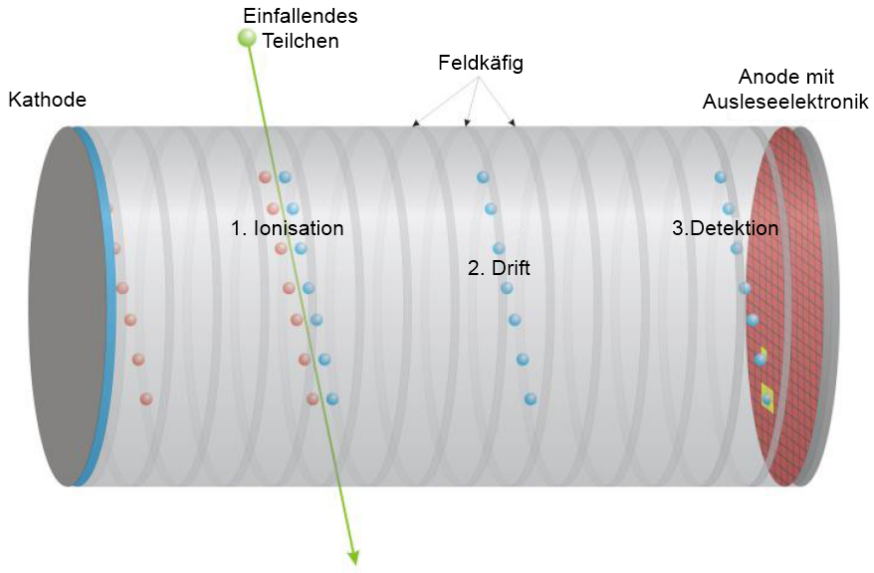
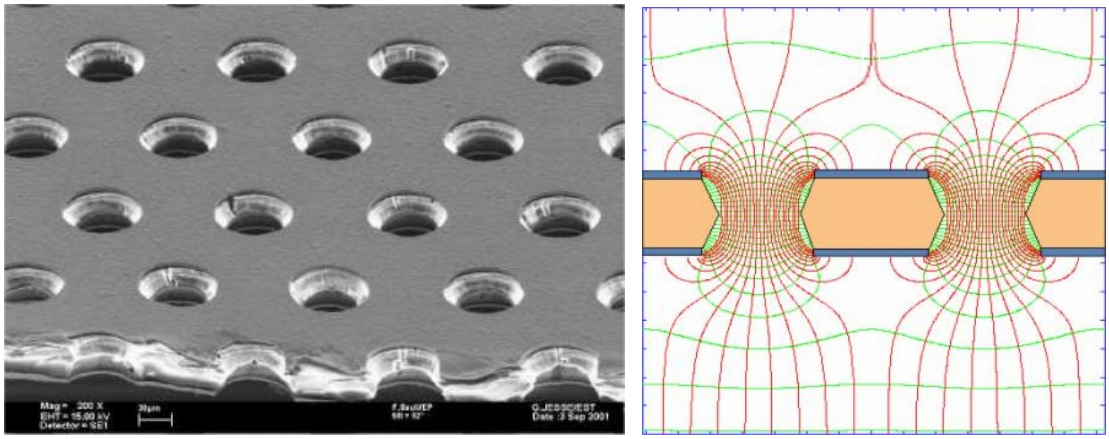


Abbildung 2.1: Schematische Darstellung einer Zeitprojektionskammer [1]

Bei einer TPC handelt es sich um einen Spurdetektor. TPCs bestehen meistens aus einem zylindrischen Volumen, an dessen Enden sich eine Kathode und eine Anode mit der Ausleseelektronik befindet. Das Volumen ist mit einem Gas gefüllt, das sich durch die Teilchen, die man detektierten möchte, ionisieren lässt.
In Abbildung 2.1 ist das ein Schema einer TPC dargestellt. Wenn ein geladenes Teilchen das aktive Detektorvolumen durchquert, ionisiert es entlang seines Weges Gasatome und hinterlässt so eine Spur von Ionen und Elektronen. Durch das angelegte elektrische Feld bewegen sich die Ionen zur



(a) Nahaufnahme einer GEM mit einem Lochabstand von 70, bzw. 140 µm

(b) Elektrisches Feld und daraus resultierende Äquipotentiallinien innerhalb der GEM - Löcher

Abbildung 2.2: Schema und Aufnahme der Löcher einer GEM.[3]

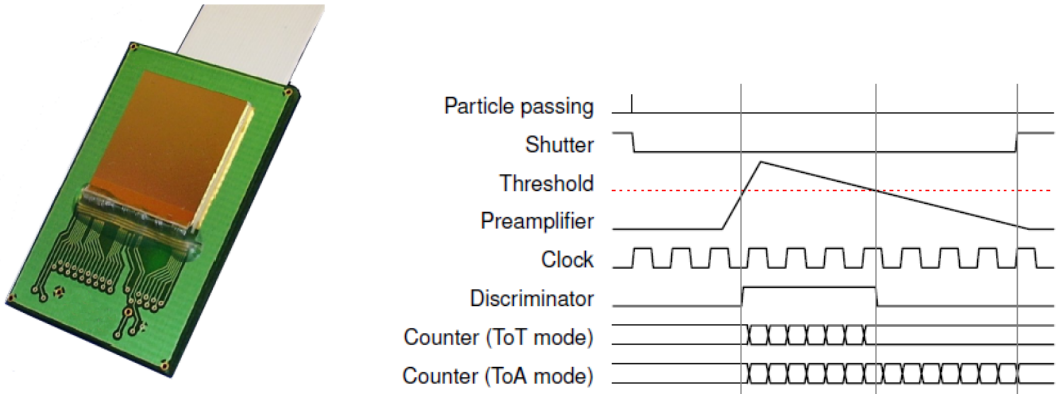
Kathode und Elektronen zu der Anode mit Ausleseelektronik am Ende der Kammer. Man spricht an dieser Stelle von der Drift der Ionen und Elektronen, sodass dieser Teil der Zeitprojektionskammer als Driftkammer bezeichnet wird. An der Endkappe müssen die Elektronen detektiert und dazu zunächst verstärkt werden. Hier gibt es verschiedene Möglichkeiten von denen im Folgenden auf die Gasverstärkung mittels „Gas Electronmultiplier"-Folien (GEMs) und auf die darauf folgende Detektion mittels Timepix Chip eingegangen wird.[1]
Bei dieser Art der Datennahme wird zunächst nur die Projektion der Teilchenspur in die Ebene der Ausleseelektronik ausgelesen. Die dreidimensionale Spur wird durch Hinzunehmen der Zeitinformation über das Eintreffen der Elektronen auf dem Auslesemedium berechnet. Weiterhin kann man durch (homogene) Magnetfelder innerhalb der Driftkammer den Impuls des detektierten Teilchens bestimmen: Durch die Lorentzkraft wird der senkrecht zum Magnetfeld stattfindende Teil der Bewegung des Teilchens auf eine radiale Bahn gezwungen. Die Krümmung dieser Bahn lässt auf den Impuls und - bei bekannter Masse - auf die kinetische Energie des Teilchens schließen.

2.1.1 GEM - Gas ElectronMultiplier Foil

Eine Möglichkeit, Primärelektronen zu verstärken und anschließend zu detektieren, ist die Kombination von einer oder mehrerer GEMs (**G**as **E**lectron**M**ultiplier Foil) mit einer segmentierten Ausleseebene. Eine solche Folie verstärkt Primärelektronen indem diese in einem starken elektrischen Feld beschleunigt werden und durch Stoßionisation eine Elektronenlawine auslösen.

Eine GEM besteht aus einer 50 µm dicken Kapton-Folie, die auf beiden Seiten mit Kupfer beschichtet ist und durch Ätzprozesse mit einem Lochabstand von 140 µm gelocht wurde (Abb. 2.2(a)).
Wird nun zwischen den beiden Kupferflächen eine hohe Spannung angelegt ($\approx 400\,\mathrm{V}$), entsteht ein starkes Feld innerhalb der Löcher (Abb. 2.2(b)) und Elektronen bewegen sich in diese hinein und werden dort beschleunigt. Hierbei ionisieren sie weitere Gasmoleküle, sodass man unterhalb der GEM für jedes eingegangenen Elektron eine Verstärkung von 10 - 100 erhält. Bei der Verwendung einer GEM innerhalb einer TPC wird diese in die Anode eingebaut. Der Bereich über der GEM heißt in diesem Fall Driftbereich und der zwischen dieser und der Ausleseelektronik wird als Induktionsbereich bezeichnet. Nutzt man bei der Verstärkung mehrere GEMs, wird der Bereich zwischen zwei GEMs als Transferbereich bezeichnet. Der Vorteil bei einem Aufbau mit mehreren GEMs ist, dass diese auf einer niedrigeren Spannung operieren können, und sich so, bei gleicher Gasverstärkung, das Risiko von Überschlägen sowie der Ionenrückfluss minimieren lässt.[4]

[1] Weiteres zu Zeitprojektionskammern findet sich in [2]



(a) (b)

Abbildung 2.3: (a): Der Timepix Chip(b): Schema der verschiedenen Messmodi des Timepix Chip[5]

2.1.2 Der Timepix Chip

Der Timepix Chip (Abb. 2.3(a)) wurde ursprünglich für Anwendungen in der medizinischen Bildgebung (Röntgendetektoren) entwickelt. Der Chip ist unterteilt in 256 × 256 Pixel, von denen jeder 55 × 55 µm^2 groß ist. Somit hat der Chip eine aktive Fläche von etwa $1,4 \times 1,4\,\mathrm{cm}^2$. Jeder Pixel ist mit einem ladungsempfindlichen Vorverstärker und einem Diskriminator ausgestattet und kann in vier verschiedenen Messmodi betrieben werden. Anhand von Abbildung 2.3(b) sollen diese verschiedenen Messmodi erläutert werden:

Single Hit: Gibt die Information aus, ob ein betreffender Pixel getroffen wurde oder nicht.

Medipix: Zählt die Ereignisse über einer gegebenen Schwelle.

ToA: Beginnt ab dem Zeitpunkt der Ladungsdeposition auf dem Pixel Taktsignale zu zählen und hört damit auf, wenn das „Shutter“-Signal (vgl. „Shutter“ und „Counter (ToA mode)“ in Abb. 2.3(b)) wieder hoch gesetzt wird. Mit Hilfe dieses Messmodus lässt sich die Ankunftszeit (engl. **T**ime **o**f **A**rival) bestimmen.

ToT: Beginnt ab dem Zeitpunkt der Ladungsdeposition auf dem Pixel Taktsignale zu zählen, bis das Signal des Vorverstärkers wieder unter (vgl. „Threshold“, „Preamplifier“ und „Counter (ToT mode)“ in Abb. 2.3(b)) die Diskriminatorschwelle gefallen ist. So wird die Zeit in der sich das Signals über der Schwelle befindet gemessen (engl. **T**ime **o**ver **T**hreshold), aus der sich die auf dem Chip deponierte Ladung bestimmen lässt.

Wenn man abwechselnd einen Pixel im ToT-Modus und einen Pixel im ToA-Modus betreibt, sodass der Chip ein Schachbrettmuster der beiden Modi aufweist, ist es möglich, eine Zeit und eine Ladungsinformation von Ereignissen zu erhalten, die mehr als einen Pixel bedecken. In Abbildung 2.4 findet sich ein Ausschnitt aus einem Ereignis, das diese Art der Konfiguration verdeutlicht. So weisen die in der Abbildung dargestellten Ladungsdepositionen (Cluster), das eben beschriebene Schachbrettmuster auf und es ist nun möglich, für den betreffenden Cluster sowohl Zeit-, als auch Ladungsinformation zu bestimmen.

2.2 Der TPC Prototyp in Bonn

In Bonn befindet sich eine zylindrische TPC in Betrieb (Abb. 2.5), deren Auslese mit Hilfe von GEMs und einem Timepix Chips realisiert ist.

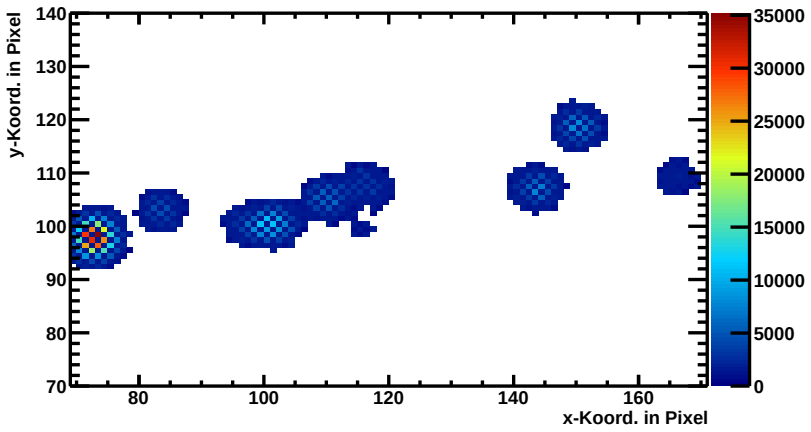
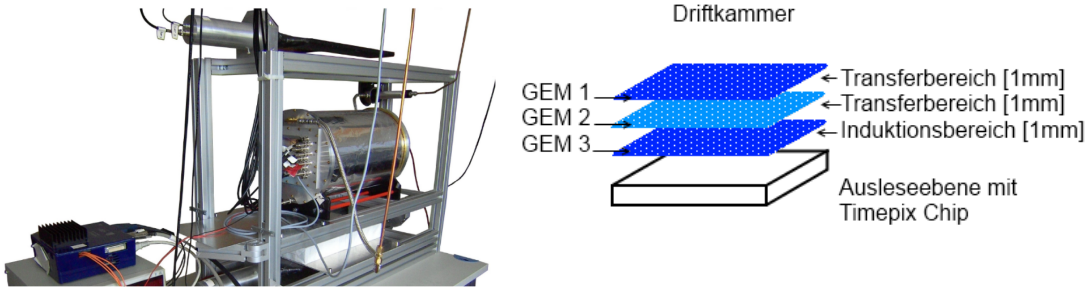


Abbildung 2.4: Ausschnitt der Ausgabe des Timepix Chips - (Ereignis 0, Datensatz 090515002) - Man erkennt bei den messenden Pixeln das durch abwechselnd geschaltete ToT (bunt)- und ToA-Pixel (gleichfarbig) hervorgerufene Schachbrettmuster.



(a) (b)

Abbildung 2.5: (a) Der TPC-Prototyp Bonn[5] sowie (b) ein Schema des Aufbaus der Ausleseelektronik

Die TPC hat eine Driftkammer mit einer Länge von 26 cm. Ihr innerer Durchmesser beträgt 23 cm und sie hat ein Driftfeld von bis zu einem 1 kV cm^{-1}, das durch einen Feldkäfig mit 187 Kupferringen geformt wird. In ihrer Endkappe befindet sich die Verstärkungs- und Ausleseelektronik. Diese bestehen aus einem Stapel von 3 GEMs, unter denen sich ein Timepix Chip in der im vorigen Kapitel beschriebenen ToA/ToT-Konfiguration befindet. Beide Transferbereiche zwischen den GEMs und der Induktionsbereich haben eine Höhe von 1 mm (vgl. Abb. 2.5(b)). Die GEM's selbst sind standard CERN GEM's mit einer Fläche von 10 cm × 10 cm und einer Dicke von 56 µm.
Bei den in späteren Kapiteln (Kapitel 4 und 5) diskutierten Daten handelt es sich um Teilchenspuren von kosmischer Strahlung. Diese wurden mit dem eben beschriebenen TPC-Prototypen aufgenommen, dessen Ausleseelektronik dazu mit zwei in Koinzidenz geschalteten Szintillationsdetektoren über und unter der Kammer getriggert wurde. Als Gas in der Driftkammer wurde eine Mischung aus Helium und Kohlenstoffdioxid mit einem Mischverhältnis von 70:30 genutzt. Das Driftfeld betrug 495 V cm^{-1}. Damit ergibt sich nach Simulationen, bei einer Temperatur von 30 °C und einem Druck von 1013 hPa, eine erwartete Driftgeschwindigkeit der Elektronen von $(9{,}74 \pm 0{,}02)$ µm ns^{-1}.[2] Die anliegende Spannung zwischen Ober- und Unterseite der GEMs der

[2]Simulationen mit Magboltz [6]

Ausleseelektronik beträgt 415 V, während das Feld in den Transferbereichen eine Feldstärke von 2200 V cm^{-1} hat und das Feld im Induktionsbereich 3000 V cm^{-1} groß ist.[3]

2.3 Analyse der Daten des Timepix Chips

Die Ausgabe des Timepix Chip erfolgt zunächst als 256×256 Matrix in eine Datei im ASCII-Format, wobei jeder Eintrag einem Pixel entspricht. Ein solcher Eintrag enthält einen ADC-Wert, der die entsprechende - von einem analogen Signal in ein digitales gewandelte - Information über die deponierte Ladung oder den „Zeitwert“ trägt. Eine Datei mit der oben beschriebenen Matrix wird für jedes Ereignis angelegt. Ein Ereignis entspricht einer durch die Szintillationsdetektoren ausgelöst Messung, somit trägt die Matrix in der Ausgabedatei die Daten einer Teilchenspur.

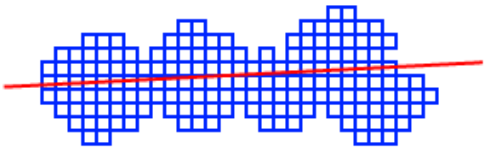
Danach werden die Ausgabedateien des Timepix Chip in eine Datei des LCIO-Format eingelesen, welches zum LCIO-Datenpaket (**L**inear **C**ollider **I**nput/**O**utput) gehört. Das Paket wurde im Rahmen von ILC Soft entwickelt und an die Verarbeitung von Daten angepasst, die bei Experimenten mit Linearbeschleunigeren anfallen. Neben dem Datenformat gehören zum LCIO-Datenpaket noch verschiedene C++ (sowie Java) Routinen, mit denen man LCIO-Daten in verschiedene Daten-Typen lesen und schreiben kann. Eine solche LCIO-Datei bietet die Möglichkeit die „Rohdaten“ vieler Ereignisse abzuspeichern. Darüber hinaus gibt es innerhalb einer LCIO-Datei weitere Daten-Typen, um die Ergebnisse verschiedener Analyseschritte, die die Rohdaten durchlaufen, zu speichern.[4]
Diese Analyseschritte werden mit Hilfe des Programmes MarlinTPC (**M**odular **A**nalyis & **R**econstruction for the **LIN**ear **C**ollider) realisiert.[5] Es basiert auf dem LCIO-Datenpaket und C++ und wird dazu genutzt, jeden einzelnen Analyseschritt in einem sogenannten Prozessor - einem C++ Algorithmus - ablaufen zu lassen. So kann die Analyseketten modular zusammengesetzt, und auch problemlos einen Analyseschritt mit verschieden Parametern mehrfach durchlaufen, werden. Die einzelnen Prozessoren für eine solche Analyse und deren Parameter werden MarlinTPC in einer XML-„Steuerungs“-Datei vorgegeben, welche dann Ereignis für Ereignis abgearbeitet werden.

Bevor der Analysefluss beginnt, werden die ASCII-Dateien mit den Matrizen zuerst in eine LCIO-Datei eingelesenen und dort als Rohdaten gespeichert. Dann beginnt die eigentliche Analyse innerhalb von Marlin. Dabei werden die Rohdaten zuerst auf alle „Pixelwerte“ mit einem ADC-Wert gleich Null, oder kleiner einer gegebenen Schwelle, bereinigt. Danach sucht ein erster Separationsalgorithmus nach dem, was zuvor schon als Cluster bezeichnet wurde: Zusammenhängenden „Inseln“ von Pixeln, die durch die Deposition von Ladung auf diesen entweder Ladung oder die Zeit des Eintreffens der Ladung gemessen haben. Diese Cluster werden dann durch den eigentliche Trennalgorithmus in sogenannte Hits separiert. Als Hits werden im Kontext dieser Bachelorarbeit die Resultate der Clusterseparation bezeichnet - bei einem idealen Trennalgorithmus würde man für jede Ionisation in der TPC einen einzelnen Hit finden. Ein schlechter Algorithmus könnte einen Cluster, der durch das Überlappen der Signale mehrerer Ionisationen entstanden ist, nicht weiter trennen und würde diesen dann als Hit und Resultat einer einzigen Ionisation sehen. Der aktuell verwendete Clustertrenner ermittelt zunächst für jede Insel durch Lineare Regression die Schwerpunktsachse (vgl. zur Veranschaulichung Abb. 2.6(b)), um danach alle Pixel dieses Clusters nach der Ladungsinformation gewichtet auf die Schwerpunktsachse zu projizieren (Abb. 2.6(c)). Es folgt die eigentliche Trennung der Inseln bei einem lokalen Minimum der Projektion, die für das Beispiel in Abbildung 2.6(a) zu dem in Abbildung 2.6(d) dargestellten Ergebnis führt. Diese Art der Separation funktioniert sehr gut für Cluster deren Hits auf einer Achse liegen und durch die sich so eine genaue Schwerpunktsachse legen lässt (z.B die linke, grüne Insel in Abb. 2.6(a)), während bei der Trennung von Clustern, für die dies nicht der Fall ist, Probleme auftreten. Ein solcher ist z.B. die rechte, blaue Insel (Abb. 2.6(a)), die auch als Beispiel für den Trennalgorithmus diente. Um dieses Problem zu beheben wurde dieser Separationsalgorithmus dahingehend angepasst, dass

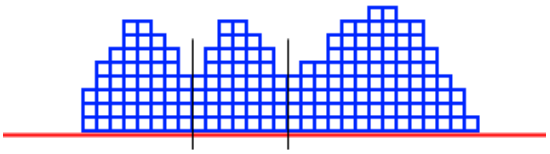
[3] Für weitere Details zum Aufbau der TPC und den Simulationen siehe [5] und [10]
[4] Weiteres zu LCIO siehe: [7] und [8]
[5] Näheres zum MarlinTPC Datenpaket unter [9]

Clustertrennung zweimal durchgeführt wurde und die Cluster so zweimal auf verschiedene Achsen projiziert werden um diese besser trennen zu können.

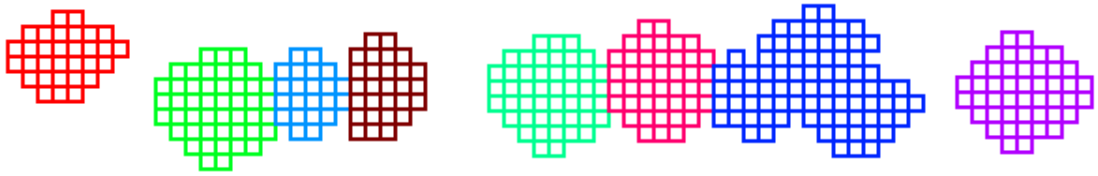
(a) Schema eine durch den Timepix Chip aufgenommen Ereignisses nach der Trennung in verschiedene Cluster.



(b) Trennalgorithmus: Legen einer Achse durch einen Cluster



(c) Trennalgorithmus: Projektion der Ladungsinformation auf die Achse des Clusters



(d) Schema das Endresultat des Trennalgorithmus

Abbildung 2.6: Die verschiedenen Schritte des Trennalgorithmus: Ein Ereignis wird zunächst in Cluster (Zusammenhängende Pixel, die etwas gemessen haben) geteilt (a). Für jeden dieser Cluster wird die Schwerpunktsachse gesucht (b) und danach die Pixel des betreffenden Clusters auf diese Achse projiziert (c). Die Trennung des Clusters erfolgt bei den lokalen Minima der Projektion und führt für das gewählte Ereignis (a) zum Resultat in der Abbildung (d).

Nach der Clustertrennung wird zunächst der Ladungsschwerpunkt jedes Hits berechnet, wobei die z-Koordinate aus den Zeit-Werten der Pixel im ToA-Modus bestimmt wird. Es folgt die Berechnung der Gesamtladung jedes Hits. Hierbei wird berücksichtigt, dass nur jeder zweite Pixel Ladung misst, indem die gesamte Ladung des Hits, die durch Pixel im ToT-Modus gemessen wurde, mit dem Quotienten „Anzahl aller Pixel“ / „Anzahl aller Pixel im ToT-Modus“ multipliziert wird. Wenn nun alle Hits, deren Zentren und Ladung bekannt sind, wird schlussendlich auf Grund dieser Daten die Spur des Teilchens durch die TPC rekonstruiert.

Kapitel 3

Source Extractor

Beim Skizzieren des Clustertrennalgorithmus im vorigen Kapitel (Abschnitt 2.3) wurde bereits auf das Problem aufmerksam gemacht, dass der aktuell verwendete Algorithmus Probleme hat Cluster korrekt zu trennen, deren Hits nicht auf einer Achse liegen. Um die Spur des Teilchens möglichst genau rekonstruieren zu können, ist es aber erstrebenswert bei der Clustertrennung alle Hits zu separieren. Um dies zu erreichen, wurde in dieser Arbeit der bestehende Trennalgorithmus durch Source Extractor ersetzt.
Source Extractor, oder auch SExtractor ist ein Programm, das in der Astronomie benutzt wird, um auf CCD-, beziehungsweise Photoaufnahmen Objekte (Sterne bzw. Galaxien) zu lokalisieren und weitere Information über diese zu gewinnen. SExtractor ist insbesondere in der Lage die gewünschten astronomischen (Ort eines Objekts), geometrischen (Form des Objektes: rund, elliptisch) und photometrischen Daten aller gefundenen Objektes eines Bildes auszugeben. Im folgenden soll nun die Funktionsweise des Programms erläutert und die wesentlichen Konfigurationsparameter vorgestellt werden.

Beim Aufruf von SExtractor wird als Argument die Datei mit der astronomischen Aufnahme (*Eingabedatei*) und bei Bedarf eine *Konfigurationsdatei* bzw. Konfigurations*parameter* an das Programm übergeben. Das Programm führt dann seine Operationen gemäß der Konfigurationsdatei durch und legt als Ausgabe einen „Katalog“ an, in den es für jedes Objekt die gewünschten Daten (angegeben in einer „*Parameterdatei*“) hineinschreibt. Wird keine Konfigurationsdatei angegeben, sucht das Programm nach der Standarddatei „*default.sex*“.

sex [*Eingabedatei*].fits -c [*Konfigurationsdatei*].sex -[*Parametername*] [*Parameterwert*]

3.1 FITS - Flexible Image Transport System

Die Eingabe-Datei ist eine Art Bild (**.fits*) im FITS-Format (FITS - **F**lexible **I**mage **T**ransport **S**ystem) mit den möglichen Clustern, die SExtractor trennen soll. Das FITS-Format wurde für astronomische Aufnahmen entwickelt und bietet die Möglichkeit Bilder in Form von null- bis zu „999“-dimensionalen Arrays von Pixeln oder auch Tabellen abzuspeichern. Weiterhin gibt es innerhalb einer FITS-Datei Platz weitere Daten zu jedem Bild oder jeder Tabelle in einem Header zu speichern.
Der Aufbau einer FITS-Datei soll an Abbildung 3.1 erläutert werden: Jede FITS-Datei besteht zunächst aus einer sogenannten „Primary Header/Data Unit“ (Primary HDU oder auch Primary Extension), auf die weitere, optionale HDUs folgen können. (Angedeutet durch den Spalt in der Mitte der Abb. 3.1.) In diesem Fall spricht man von einem „Multi-Extension-FITS-File“. Jede HDU selbst besteht wiederum aus einer „Data Unit“ und einer „Header Unit“, wobei die Data Unit der entsprechenden HDU die eigentliche Information enthält, und die Header Unit zusätzliche Informationen zum Inhalt der Data Unit trägt. Hierzu ist anzumerken, dass - unabhängig von der Anzahl der HDU's - die Data Unit der Primary HDU immer aus einem Bild (Array) besteht,

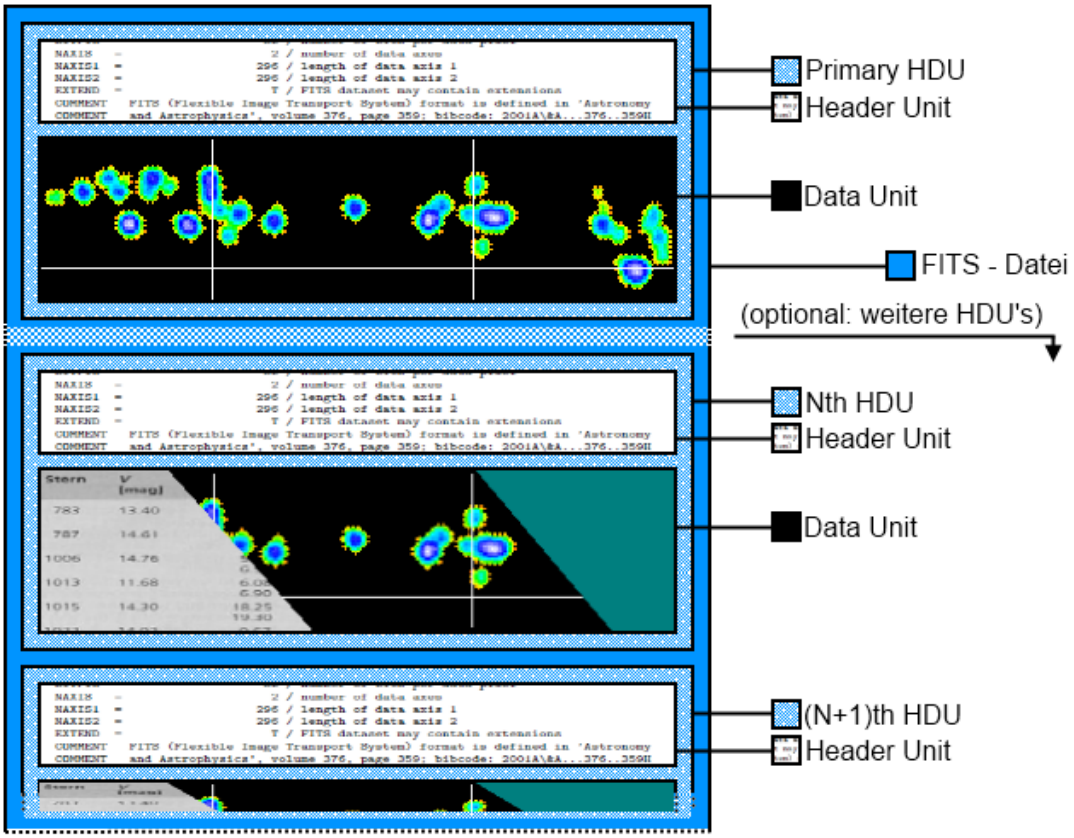


Abbildung 3.1: Aufbau einer Datei im *.fits-Format.

während die Data Units der folgenden HDUs entweder ein null- bis „999“-dimensionale Array oder eine Tabelle enthalten oder auch einfach leer sein können. Die Header Unit muss hingegen für jede Extension vorhanden sein. Sie ist im ASCII-Format abgefasst, grundsätzlich ein vielfaches von 2880 Byte lang (nicht benötigter Platz wird mit ASCII Nullen oder Leerzeichen gefüllt) und enthält eine Serie benötigter Daten zu Größe und Format der folgenden Data Unit. Jeder Wert in einem solchen Header ist in der Art *WERTNAME = WERT / KOMMENTAR* formatiert.
Wenn im Zusammenhang von FITS-Dateien von einem Bild die Rede ist, ist ein zweidimensionales Array innerhalb der HDU gemeint.

3.2 Konfigurationsdateien und Katalogdatei

Neben der FITS-Datei wurde zu Beginn dieses Kapitels eine weitere Eingabedatei (*Konfigurationsdatei.sex*) erwähnt, die Konfigurationsdatei. Neben dieser gibt es zwei weitere: Eine Datei (Parameterdatei) in der angegeben wird welche Daten SExtractor aus der vorliegenden FITS-Datei ermitteln soll (*Parameterdatei.param*), und eine Filterdatei in der sich ein Filter (in matrixform) befindet, der auf das Bild innerhalb der FITS-Datei angewendet werden kann. (Ein Beispiel einer Filterdatei befindet sich in auf Seite 15 als Beispiel 1, während eine Beschreibung der Funktionsweise eines Filters im Abschnitt 3.3.2 dieses Kapitels zu finden ist.) Die Bennenungskonvention für eine Filter-Datei (*mexhat_2.0_7×7.conv*) ist: Art des Profils (z.B. „Mexicanhat“), gefolgt von der Halbwertsbreite des selben (z.B. 2) und von der Größe der Matrix in der der Filter verfasst ist (z.B. 7×7).
In der Konfigurationsdatei werden der Pfad zur Parameterdatei, der zur Filterdatei und der

Pfad der Ausgabedatei neben vielen weiteren Parametern angegeben, die vom Nutzer an seine Bedürfnisse angepasst werden müssen, damit mit SExtractor optimale Ergebnisse erzielt werden. Die Parameter werden im folgenden Abschnitt 3.3 vorgestellt, zudem wird auf ihre Anwendung näher eingegangen. In Kapitel 4 werden dann die Auswirkungen der Wahl verschiedener Parameterwerte am Beispiel von mit einer TPC aufgenommen Ereignissen illustriert.
Die Ausgabe der von SExtractor getrennten Cluster und der zugehörigen Daten (z.B. Koordinaten des Zentrums, Form, FWHM) erfolgt in eine „Katalogdatei“ (Katalog). Der Katalog besteht aus einer Tabelle mit einem Header: Im Header finden sich Angaben zu den von SExtractor ermittelten Daten in der Reihenfolge, in der diese in den Spalten der im Katalog auf den Header folgenden Tabelle zu finden sind. In jeder Zeile einer solchen Tabelle findet sich die Daten eines Hits.

3.3 Arbeitsschritte innerhalb SExtractor

Source Extractor geht in 10 Schritten beim Finden, Trennen und Berechnen von Clustern vor, die im folgenden genannten und dann erklärt werden sollen:

1. Feststellen des Rauschens und dessen Effektivwert
2. Abziehen des Rauschens
3. Anwenden des Filters
4. Cluster suchen und finden
5. Teilen der Cluster
6. Feststellen von Form und Position der gefundenen Hits
7. Bereinigen der detektierten Hits (Einfluss von Nachbarn, etc)
8. Photometrie
9. Klassifizieren der Hits
10. Erstellen des „Ausgabe-Katalog“

3.3.1 Feststellen des Rauschens und dessen Effektivwert

Source Extractor bestimmt standardmäßig ein lokales Rauschen für gegebene Gebiete des Bildes. So wird das Bild in Bereiche der Größe *BACK_SIZE* aufgeteilt und in diesen der Effektivwert (RMS) und das σ (Standardabweichung) der Pixelwerteverteilung berechnet. Danach werden die am weitesten abweichenden Pixelwerte verworfen und die Berechnung findet erneut statt. Dies wird so lange vorgenommen bis alle übrigen Pixelwerte in einer $\pm 3\sigma$ - Umgebung um den Mittelwert liegen, der dann als Wert des Rauschens angenommen wird. Nach der Bestimmung des Rauschens wird dann dieses vom eigentlichen Pixelwert abgezogen.
Es besteht weiterhin die Möglichkeit, sich das Rauschen nicht von SExtractor errechnen zu lassen, sondern ein globales, konstantes Rauschen anzugeben. Dazu stellt man den *BACK_TYPE* Wert von *AUTO* auf *MANUAL* und gibt den gewünschten Wert in *BACK_VALUE* vor. An dieser Stelle sei bereits gesagt, dass die im Rahmen dieser Bachelorarbeit als FITS dargestellten Daten kein Rauschen tragen, so dass bei der Clustertrennung mit SExtractor auf die Berechnung verzichtet und als Wert des manuell eingestellten Rauschens 0 gewählt wurde.

BACK_SIZE Größe eines Gebietes in der das lokale Rauschen für jedes Bild des betreffenden FITS berechnet wird. Wird nur ein Wert angegeben, arbeitet SExtractor mit einem quadratischen Gebiet der Seitenlänge *BACK_SIZE*, werden zwei Werte angegeben wird mit einem entsprechenden Rechteck gearbeitet.

BACK_TYPE Einstellmöglichkeit ob SExtractor das ermittelte Rauschen von den Pixelwerten (Wert: *AUTO*) oder einen konstanten Wert (angegeben in *BACK_VALUE*) von diesen abziehen soll (Wert: *MANUAL*).

BACK_VALUE Wert der im Falle des Abziehens eines konstanten Rauschens als Wert des Rauschens angenommen wird.

3.3.2 Glättung durch Filter

Bevor die Cluster in ihre einzelnen Hits getrennt werden, kann das Bild mit SExtractor geglättet werden. Wenn *FILTER* auf *Y* gesetzt ist wird der in *FILTER_NAME* angegebene Filter geladen und auf alle Pixel mit Pixelwerten größer dem in *FILTER_TRESH* angegebenen Wert angewendet. Dies führt zu einer Glättung des Bildes und kann, bei Nutzung des richtigen Filters, von großem Nutzen sein. Es gibt verschiedene Arten von Filtern: gaußförmige, blockfunktionsförmige, in der Art eines „mexicanhat"-Profils, und in Form eines abgeschnittenen Gauß.
Bei einem solchen Mexicanhatprofil handelt es sich um die zweite, normalisierte Ableitung einer Gaußfunktion. In Gleichung 3.1 findet sich eine Mexicanhatfunktion $M(x)$ in zwei Dimensionen, die auch in Abbildung 3.2 für ein σ von 1.5 abgebildet ist.

$$M(x) = \frac{2}{\sqrt{3\sigma}\pi^{\frac{1}{4}}} \left(1 - \frac{x^2}{\sigma^2}\right) e^{-\frac{x^2}{2\sigma^2}} \tag{3.1}$$

Der eigentliche Filter ist eine Datei mit der Endung *.conv*, in der sich eine quadratische, zwei dimensionale Matrix mit dem Profil der dem Filter zu Grunde liegenden Funktion befindet (vgl. Beispiel 1). Für die eben genannte Mexicanhatfunktion würde sich ein entsprechender Filter aus den Werten einer dreidimensionalen, normierten Mexicanhatfunktion an bestimmten Koordinaten zusammen setzten. Damit enthält die Matrix des Beispiel Filters die Werte eines solchen Profils (mit einer Halbwertsbreite von 2) an den Koordinaten im Intervall von $x \in [-3:3]$ und $y \in [-3:3]$.

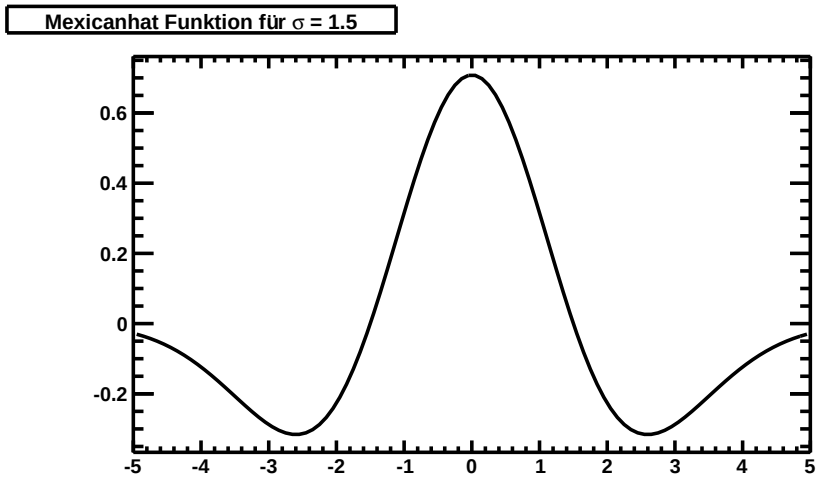


Abbildung 3.2: Mexicanhatprofil in zwei Dimensionen für ein σ von 1.5

Wie die Anwendung des Filters in Source Extractor realisiert ist, soll an Abbildung 3.3 erklärt werden. Die Pixelwertevertilung (linker Kasten in der Abbildung) wird Pixel für Pixel durchlaufen und der Filter (mittlerer Kasten) auf den Pixel sowie auf eine durch den Filter gegebene Anzahl von Nachbarpixeln angewendet. So setzt sich zum Beispiel der rot umrandete Pixelwert im geglätteten Bild (rechter Kasten) aus dem Pixelwert im ursprünglichen Bild und den acht benachbarten Pixelwerten der Art zusammen, dass die ursprünglichen Pixelwerte mit den Werten des Filters multipli-

```
CONV NORM
# 7x7 convolution mask of a mexican-hat for images with FWHM~2.0 pixels.
-0.000006 -0.000132 -0.000849 -0.001569 -0.000849 -0.000132 -0.000006
-0.000132 -0.002989 -0.017229 -0.028788 -0.017229 -0.002989 -0.000132
-0.000849 -0.017229 -0.042689  0.023455 -0.042689 -0.017229 -0.000849
-0.001569 -0.028788  0.023455  0.356183  0.023455 -0.028788 -0.001569
-0.000849 -0.017229 -0.042689  0.023455 -0.042689 -0.017229 -0.000849
-0.000132 -0.002989 -0.017229 -0.028788 -0.017229 -0.002989 -0.000132
-0.000006 -0.000132 -0.000849 -0.001569 -0.000849 -0.000132 -0.000006
```

Beispiel 1: Beispiel für einen SExtractor-Filter - Mexicanhat, FWHM = 2.0 Pixel, Größe: 7×7

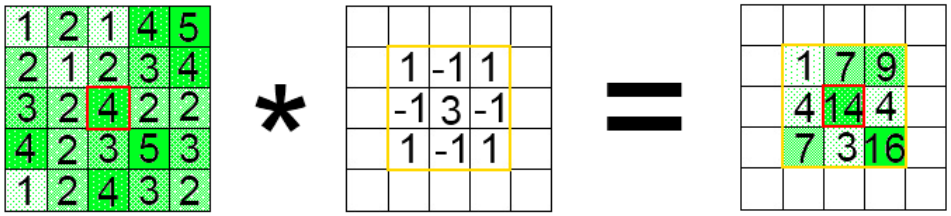
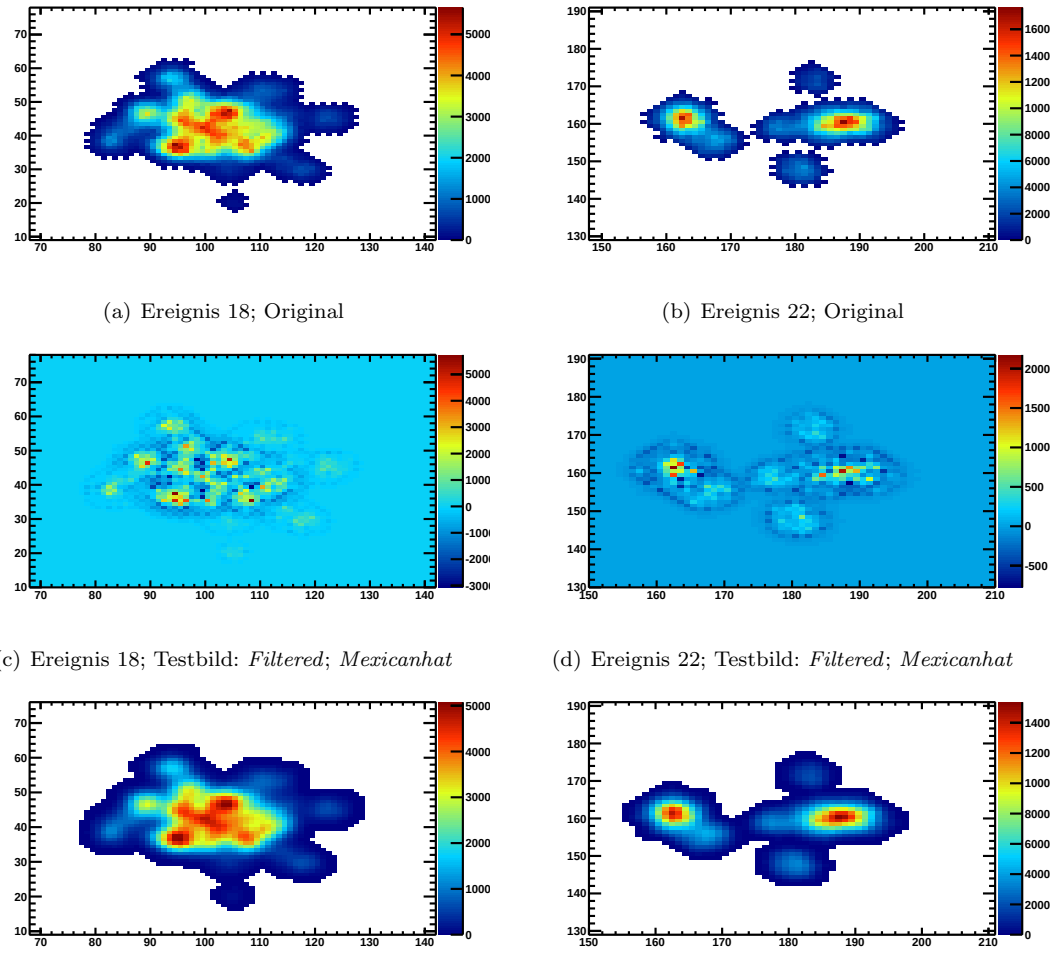


Abbildung 3.3: Schema zur Arbeit von Filtern: Der linke Kasten stellt eine Pixelwerteverteilung dar, die mit dem Filter (mittlerer Kasten) „gefaltet“ das Bild im rechten Kasten ergibt. Die Pixelwerte in der Ergebnis Verteilung kommen folgendermaßen zu Stande: (Betrachtet wird der Pixelwert im roten Kasten.) Die Pixelwerte um den zu bearbeitenden Pixel (und auch dieser) werden mit den Werten des Filters multipliziert und ergeben addiert den neuen Wert. $\rightarrow 1 \cdot 1 - 2 \cdot 1 + 3 \cdot 1 \; - 2 \cdot 1 + 4 \cdot 3 - 2 \cdot 1 \; + 2 \cdot 1 - 3 \cdot 1 + 5 \cdot 1 \; = \; 14$

ziert und dann addiert werden. Die 4 wird so über $1 \cdot 1 - 2 \cdot 1 + 3 \cdot 1 \; - 2 \cdot 1 + 4 \cdot 3 - 2 \cdot 1 \; + 2 \cdot 1 - 3 \cdot 1 + 5 \cdot 1$ zur 14.
Nach der Umsetzung des Filterns in SExtractor, soll nun die Wirkung eines solchen Filters an einer etwas größeren Pixelwerteverteilung betrachtet werden (vgl. Abb. 3.4). Das in der FITS-Datei enthaltene Bild wird durch Anwendung des Filter an das diesem zu Grunde liegende Profil angepasst. So glätten zum Beispiel gaußförmige Filter die Ränder der Cluster und passen deren Pixelwerte-Verteilung an einen zweidimensionale Gauß-Verteilung an. Dieser Effekt lässt sich gut im Vergleich der Bilder 3.4(a) (und 3.4(b)) zu den geglätteten Bildern 3.4(e) (und 3.4(f)) beobachten.
Im Vergleich dazu verstärken Filter in Form von „mexicanhat“ Profilen die Unterschiede zwischen den Pixeln verschiedener Pixelwerte. Dies führt dazu, dass Maxima im Verhältnis zu Minima angehoben, bzw. Minima im Verhältnis zu Maxima abgesenkt werden (vgl. die Originale 3.4(a) und 3.4(b) im Vergleich zu den gefilterten Bildern 3.4(c) und 3.4(d)). Die Wirkung der „mexicanhat“ Filter macht diese zu einem nützlichen Instrument bei der Trennung von Clustern, die viele Hits auf kleinem Raum enthalten. Neben den beiden schon genannten Filtern gibt es noch Filter in Form einer „Plateau“- und „Pyramiden“-Funktion, wobei letztere zu den „Standardeinstelllungen“ von SExtractor gehört, wenn man *FILTER* auf *Y* setzt und keinen eigenen Filter angibt.

Bei den folgenden Arbeitsschritten arbeitet SExtractor nur bei der Trennung der Cluster mit dem gefilterten Bild. Nach der Bestimmung der geometrischen Parameter nutzt das Programm wieder die ursprünglichen Daten für weitere Berechnungen.

FILTER Wenn *FILTER* auf *Y* gesetzt ist, wird bei der Detektion der in *FILTER_NAME* ange-



(a) Ereignis 18; Original

(b) Ereignis 22; Original

(c) Ereignis 18; Testbild: *Filtered*; *Mexicanhat*

(d) Ereignis 22; Testbild: *Filtered*; *Mexicanhat*

(e) Ereignis 18; Testbild: *Filtered*; *Gauß*

(f) Ereignis 22; Testbild: *Filtered*; *Gauß*

Abbildung 3.4: Auschnitte zweier in FITS umgewandelter Events bei verschiedenen Filtern (Gauß & „Mexicanhat“) - Auf den Achsen ist jeweils die x- und y-Koordinate des Timepix Chip aufgetragen.

gebene Filter genutzt.

FILTER_NAME Name des verwendeten Filters.

FILTER_THRESH Schwelle für das Anwenden des Filters in σ des Hintergrundrauschens - auf Pixel mit Pixelwerten über der Schwelle wird der Filter angewendet.

3.3.3 Finden und Trennen

Bevor SOURCE EXTRACTOR mit der Hitsuche beginnt muss noch mit *DETECT_MINAREA* (*DETECT_MAXAREA*) festgelegt werden wie viele Pixel ein solcher Hit mindestens enthalten muss beziehungsweise höchstens enthalten darf. Der SExtractor-Tennalgorithmus sucht dann Gebiete zusammenhängender Pixel mit Pixelwerten größer einer gegebenen Schwelle (*DETECT_THRESH*), die die an sie gesetzten Anforderungen hinsichtlich ihrer Größe erfüllen. Jedes dieser Gebiet durchläuft dann die folgende Prozedur um alle enthaltenen Hits zu finden.

Zuerst wird nun nach dem Pixel mit dem höchsten Pixelwert eines solchen Clusters gesucht um mit dieser Information ein sogenanntes „multithresholding“ vornehmen zu können. Im Rahmen

dessen werden zwischen der in *DETECT_THRESH* angegebenen Schwelle und dem eben gefundenen maximalen Pixelwert weitere Schwellen eingefügt. Deren Anzahl richtet sich nach dem in *DEBLEND_NTHRESH* angegebenen Wert und sie werden entweder exponentiell oder linear verteilt, je nachdem ob der *DETECT_TYPE* Wert auf *PHOTO* oder *CCD* gesetzt ist. (Vergleiche hierzu Abbildung 3.5 - in dieser stellt der untere Rand die durch *DETECT_THRESH* angegebene Schwelle dar und SExtractor hat, nach dem *DEBLEND_NTHRESH* Wert, 24 neue Schwellen linear (*DETECT_TYPE CCD*) zwischen Spitze und ursprünglicher Schwelle verteilt.) Nun läuft der Algorithmus von der höchsten bis zur niedrigsten Schwelle und testet an jeder Schwelle ob plötzlich eine neue Spitze (oder mehrere) des Clusters auftaucht.

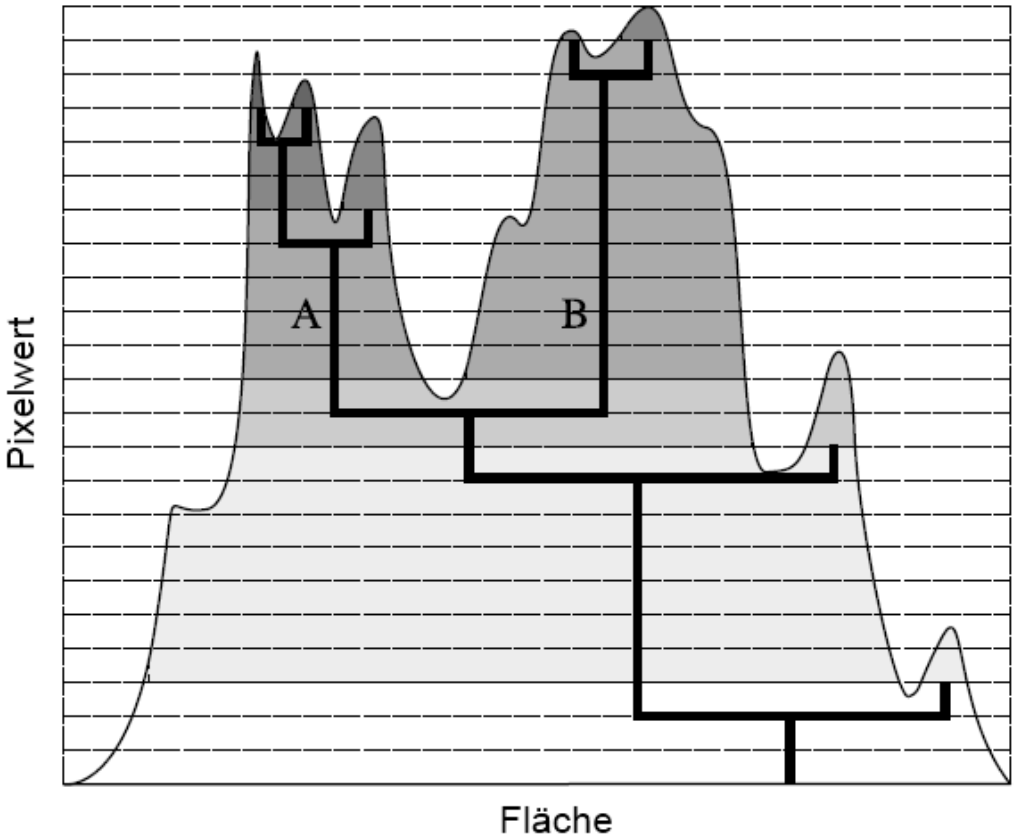


Abbildung 3.5: Zwei dimensionales Beispielbild für die Clustertrennung per „Multithresholding“ [11] - Zuerst werden zwischen dem maximalen Pixelwert und der in *DETECT_THRESH* angegebenen Schwelle 24 (*DEBLEND_NTHRESH*) weitere Schwellen eingefügt. Sie werden linear verteilt (*DETECT_TYPE* Wert *CCD*). Nun läuft der Algorithmus von der höchsten Schwelle abwärts bis zu *DETECT_THRESH* und testet an jeder Schwellen ob eine neue Spitze/mehrere neue Spitzen auftaucht/en. Vereinigen sich zwei oder mehr vorher gefundene Spitzen, überprüft SExtractor ob die integrierte Pixel-Intensität jeder dieser Spitzen für sich größer ist, als der in *DEBLEND_MINCONT* angegebene Teil des gesamten Clusters. Treffen alle diese Bedingungen für eine Spitze zu, wird diese als Hits gesehen. Der Algorithmus stoppt bei Erreichen von *DETECT_THRESH*.

Für die erste Schwelle in der Abbildung würde der Algorithmus so die beiden Spitzen des *B* Teils des Clusters sehen. Dabei betrachtet SExtractor zunächst jeden gefunden Pixel als mögliche Spitze und potentiellen Hit - unabhängig von Abstand zu einer schon gefunden Spitze oder Grundfläche der Spitze. Bei der Überprüfung der zweiten Schwelle würde eine weitere Spitze (aus dem

A Teil des Clusters) gefunden. Außerdem haben sich die beiden Spitzen aus dem *B* Teil zu einer vereinigt. Bei einer solchen Vereinigung, überprüft SExtractor ob das Volumen jeder dieser Spitzen für sich größer ist, als ein bestimmter Teil (angegeben in *DEBLEND_MINCONT*) des gesamten Clusters. Außerdem wird überprüft ob beide Spitzen eine Fläche größer/kleiner als die in *DETECT_MINAREA*/*DETECT_MAXAREA* angegebene mindest/maximal Pixelanzahl umfassen. Alle Spitzen für die dies zutrifft werden dann als einzelne Hits gesehen. Der Algorithmus stoppt, wenn er die durch *DETECT_THRESH* gegebene Schwelle erreicht hat.

Zu diesem Algorithmus ist anzumerken, dass er für alle Clusterformen gleich gut funktioniert. Wichtig für die Trennung ist nur nur das es einen Sattel zwischen zwei Hits gibt, der mindestens eine Schwelle überschreitet. Dieser Umstand ist auch ein Teil der Erklärung, warum man durch Anwendung von Filtern mit Mexicanhatprofilen Cluster besser trennen kann: Da ein solcher Filter das Verhältnis von Maxima und Minima zueinander verstärkt, werden auch Sättel vertieft und unterschreiten so eher die neu gesetzten Schwellen.

Nachdem alle Hits eines Clusters getrennt wurden, kann SExtractor die Hits um jene bereinigen (*CLEAN* auf *Y* oder *N*), die ohne ihre Nachbarn nicht gefunden worden wären. Dazu wird ein Moffat-Profil (Gleichung 3.2) an den/die jeweiligen Nachbarn eines Hits angepasst, so ihr Beitrag zum aktuell betrachteten Hit berechnet und dieser dann abgezogen. Sollte ein Hit so unter die durch *DETECT_THRESH* gegebene Schwelle fallen, wird er nicht mehr als separater Hit aufgeführt.
Für das Moffat-Profil kann durch Reihenentwicklung gezeigt werden, dass das Profil für bestimmte Werte von k und β analog zu einer Normalverteilung ist. Im Fall einer eindimensionalen Normalverteilung und eines eindimensionalen normalisierten Moffat-Profils beträgt so $k = \pi/\beta$ und β kann entweder -2.0865 oder 1.5666 annehmen.

$$I(r) = \frac{I(0)}{(1 + k \times r^2)^\beta} \tag{3.2}$$

DETECT_MINAREA Minimale Anzahl von Pixeln, die ein Hit haben muss um detektiert zu werden.

DETECT_MAXAREA Maximale Anzahl von Pixeln, die ein Hit haben darf um detektiert zu werden.

DETECT_THRESH Schwelle, ab der SExtractor annehmen soll, dass ein Pixel nicht mehr zum Rauschen sondern zu einem Objekt gehört. Hierbei handelt es sich nicht um eine absolute Schwelle, sondern der *DETECT_THRESH* Wert wird zum „lokalen“ Rauschen hinzu addiert. Außerdem muss beachtet werden, dass ein *DETECT_THRESH* von z.B. $1,5$ nicht $1,5$ in Pixelwerten, sondern $1,5\sigma$ des berechneten Rauschens entspricht.

DETECT_TYPE Angabe zur Art des zu bearbeiteten Bildes - beeinflusst die Verteilung der Schwellen beim „Multithresholding“: Ist *DETECT_TYPE* auf *CCD* gesetzt werden die Schwellen linear verteilt. Für *DETECT_TYPE PHOTO* findet eine exponentielle Verteilung der Schwellen statt.

DEBLEND_NTHRESH *DEBLEND_NTHRESH* gibt die Anzahl der Stufen an, die für den Trennalgorithmus verwendet werden.

DEBLEND_MINCONT Verhältnis zwischen dem gesamten Volumen eines Clusters und einem Kandidaten für einen Hit, dass mindestens erreicht werden muss um diesen als Hit zu zählen. Für Werte in der Nähe von 0 werden selbst kleinste Schwankungen des „Pixelwertes“ als einzelne Hits aufgelöst - für den Wert 1 werden zusammenhängende Cluster nicht getrennt.

CLEAN Wenn auf *Y*, werden die gefundenen Cluster um die Beiträge ihrer Nachbarn bereinigt.

CLEAN_PARAM Der Parameter β in der für das Bereinigen an die Nachbarn angepasste Moffat-Profil (vgl. Gleichung 3.2)

3.3.4 Analyse und Ausgabe der gefundenen Hits

Nachdem die einzelnen Hits gefunden wurden, werden diese weiter untersucht. So werden während dieser Analyse die in der Parameterdatei angegeben Parameter für jeden Hit bestimmt und in die Katalog-Datei geschrieben. Damit dies geschehen kann, müssen in der Konfigurationsdatei die Namen beider Dateien angeben werden - der, der Parameterdatei unter *PARAMETERS_NAME* und der Name der Katalog-Datei unter *CATALOG_NAME*.
Mögliche Parameter die man SExtractor in der Parameterdatei zur Berechnung vorgeben kann sind zum Beispiel die Position des Zentrums eines Hits, die Position des Pixels mit dem größten Pixelwert und auch Parameter die die Ausdehnung des Hits betreffen.

X_IMAGE & Y_IMAGE: x- bzw. y- Koordinate eines Hitzentrums in (Bruchteilen von) Pixeln, die über eine Schwerpunktsverteilung berechnet werden. Im Kontext einer astronomischen Aufnahme wäre diese eine „Lichtverteilung“, bei einer Analyse von Daten einer TPC könnte es sich um eine „Ladungsverteilung“ handeln.

$$X\_IMAGE = \bar{x} = \frac{\sum_j i_j \cdot x_j}{\sum_j i_j} \qquad Y\_IMAGE = \bar{y} = \frac{\sum_j i_j \cdot y_j}{\sum_j i_j}$$

Hierbei steht i für die Intensität (Ladung) des jeweiligen Pixels.

XMIN_IMAGE & XMAX_IMAGE: Der kleinste, bzw. der größte Wert der x-Koordinate eines Hits. Dieser Wert wird nicht berechnet, sondern einfach nur gesucht. Analog dazu gibt es die Parameter *YMIN_IMAGE* und *YMAX_IMAGE*, so dass man durch die Abfrage aller vier Werte die Koordinaten eines Rechtecks erhält, dass den Hit einschließt.

XPEAK_IMAGE & YPEAK_IMAGE: x- bzw. y-Koordinate des Pixels mit der größten Intensität innerhalb eines Hits.

Neben Informationen zur Position eines Hits kann SExtractor auch Daten zu solchen mit elliptischer Form ausgeben. So gibt es einige weitere Parameter wie z.B. die Achsen (*A_IMAGE* / *B_IMAGE*) einer solchen Ellipse und den Winkel (*THETA_IMAGE*), den diese mit der x- (y-) Achse des Bildes einschließen. Darüber hinaus gibt es noch zahlreiche weitere Parameter wie zum Beispiel die Fehler der schon beschrieben Werte, weitere Ellipsenparameter zur besseren Verarbeitung, oder auch Parameter um sich Positionen in astronomischen Koordinaten angeben zu lassen.

Neben diesem Satz von Positionsparametern gibt es noch photometrische Parameter. Zu diesen gehören zum Beispiel verschiedene Parameter zur Intensität der getrennten Objekte, die SExtractor ebenfalls berechnen und ausgeben kann. Weiterhin ist das Programm in der Lage zwischen Sternen und Galaxien (verschiedene *CLASS_STAR*-Parameter) zu unterscheiden und sich die Halbwertsbreite eines Hits (*FWHM_IMAGE*) oder auch den größten Pixelwert (*FLUX_MAX*) des selben ausgeben zu lassen.

PARAMETERS_NAME Unter *PARAMETERS_NAME* wird eine Datei (Endung: **.param*) angeben, die die Parameter enthält, die SExtractor aus dem angegeben Bild ermitteln soll.

CATALOG_NAME *CATALOG_NAME* enthält als Wert den Namen des Kataloges in den SExtractor die jeweiligen ermittelten Parameter jedes Zentrums schreiben soll.

ANALYSIS_THRESH Schwelle zur Berechnung der *CLASS_STAR*- und *FWHM*-Parameter.

THRESH_TYPE Gibt die Einheit der Schwellen *ANALYSIS_THRESH* und *DETECT_THRESH* an. Wird *THRESH_TYPE* auf *ABSOLUTE* gesetzt werden die eben genannten Schwellenwerte als „Pixelwerte“ angenommen während *RELATIVE* dafür sorgt, dass entsprechende Werte als Skalierungsfaktor des Hintergrundrauschens angenommen werden.

Kapitel 4

Implementierung des SExtractor in den Marlin Analyseprozess

In diesem Kapitel wird die Anwendung von SOURCE EXTRACTOR als Clustertrenner auf mit der TPC aufgenommene Ereignisse beschrieben und dabei genauer auf die Auswirkungen der Wahl von verschiedenen Werten für bestimmte Parameter eingegangen.

4.1 Umwandlung von LCIO-Daten in FITS

Wie zu Beginn von Kapitel 3 erwähnt, müssen die Daten, auf die SEXTRACTOR zur Clustertrennung abgewendet werden soll, im FITS-Format vorliegen. Somit muss vor dem Trennprozess eine Konvertierung vom LCIO-Format in das FITS-Format erfolgen.
Zu diesem Zweck wurde ein C++-Programm geschrieben, das sowohl den Export aus einer LCIO-Datei als auch das Schreiben eines FITS ausführt. In diesem Programm wird der Export mit Hilfe der C++ Routinen vorgenommen, die das LCIO-Datenpaket (Release Version v01-51) zur Verfügung stellt. Da es für das Arbeiten mit FITS ebenfalls einige Datenpakete mit C/C++ Routinen gibt, konnte das Schreiben der FITS-Datei über diese realisiert werden. Hierfür wurden die C/C++ Routinen genutzt, die das CFITSIO-Paket (Version 3.28) zur Verfügung stellt.[6]

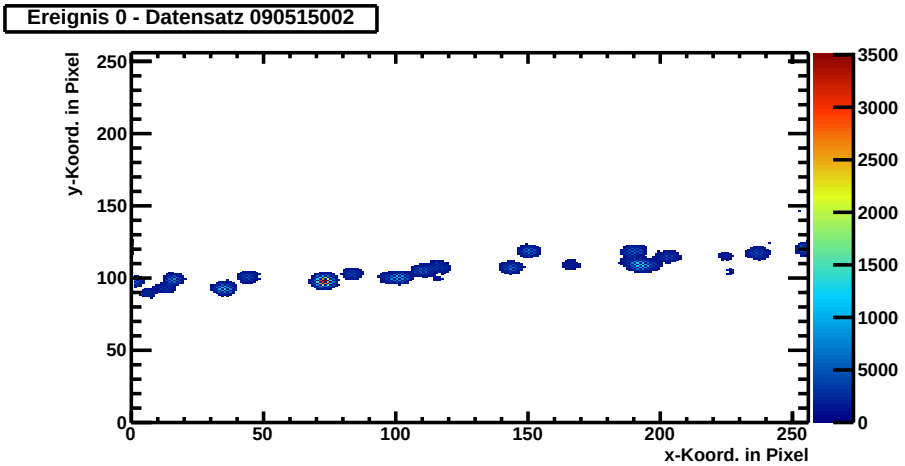
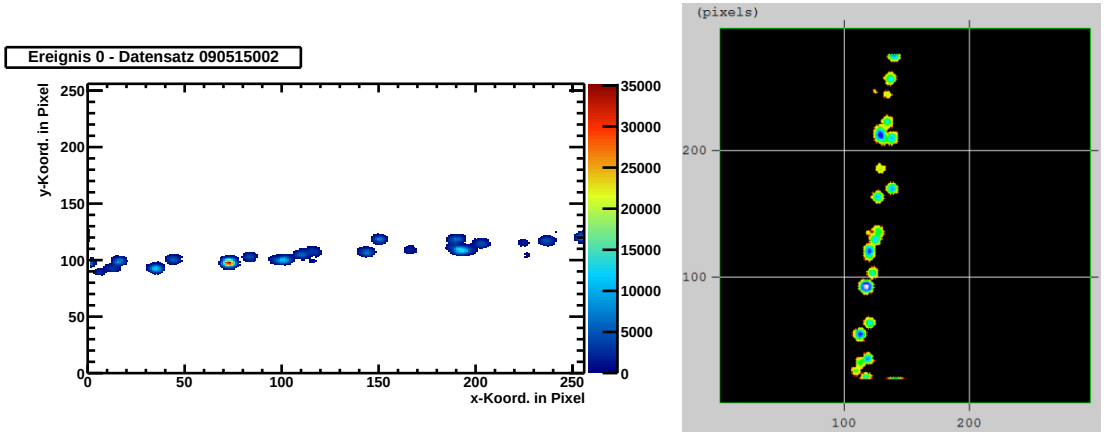


Abbildung 4.1: Beispiel für ein durch den Timepix Chip aufgenommenen Ereignis (Ereignis 0, Datensatz 090515002)

[6]Die Internetseite des CFITSIO-Projekts findet sich unter [13]

In Abbildung 4.1 findet sich ein Beispiel für ein Ereignis aus einer LCIO-Datei. In diesem ist das Schachbrettmuster, hervorgerufen durch sich abwechselnde Ladungs- und Zeitwerte, gut zu erkennen. Da die Zeitwerte allerdings für die Clustertrennung nicht von Bedeutung sind, wurden die Zeitwerte, vor der Konvertierung nach FITS durch Mittlung über die Ladungswerte der angrenzenden Pixel, zu Ladungswerten interpoliert. In Gleichung 4.1 findet sich eine Beispiel-Formel für die Ersetzung eines zentralen Pixels P an den Koordinaten x/y, der einen Zeitwert trägt. Für Pixel am Rand des Chips wurde die Formel angepasst.

$$P_{[x][y]} = \frac{1}{4}\left(P_{[x+1][y]} + P_{[x-1][y]} + P_{[x][y+1]} + P_{[x][y-1]}\right) \tag{4.1}$$



(a) (b)

Abbildung 4.2: Transformation von Daten aus dem LCIO-Format in eine FITS-Datei. Abbildung (a) zeigt das Ereignis aus Abbildung 4.1 nach der Interpolation der Zeitwerte - In Abbildung (b) findet sich das Ereignis als Darstellung einer FITS-Datei.

Das Resultat der Interpolation des Ereignisses aus Abbildung 4.1 findet sich in Abbildung 4.2(a) während in Abbildung 4.2(b) das Ereignis als FITS zu sehen ist. In der Abbildung aus der FITS-Datei fällt auf, dass die Pixelwerteverteilung an der Diagonalen zwischen x/y Achse gespiegelt ist. Dieser Effekt trat beim Schreiben der FITS auf und wurde beim späteren Einlesen der getrennten Cluster wieder rückgängig gemacht.

4.2 Parameterwahl für SExtractor als Clustertrenner

Bei ersten Tests von SOURCE EXTRACTOR (Version 2.5.0) mit den *default* Konfigurationsdateien (*default.sex, default.param, defalut.nnw, default.conv*) stellte sich heraus, dass SEXTRACTOR Probleme hat Cluster am Rand eines Bildes als solche zu erkennen.[7] Um dem Abhilfe zu schaffen wurde bei der Transformation vom LCIO-Format nach FITS ein Rand mit der Breite von 20 Pixeln um die eigentlichen Daten gelegt.
Nachdem die Ereignisse als FITS vorliegen kann nach den optimalen Einstellungen für SEXTRACTOR gesucht werden, um nach Möglichkeit alle Hits dieser Ereignisse zu finden. In diesem Kapitel wird dargestellt, wie sich die Wahl verschiedener Werte für die Parameter in der Konfigurationsdatei auf die Clusterseparation auswirkt. Wenn im Folgenden der Wert eines Parameters variiert wird oder die Werte einer Parametergruppe variiert werden, bleiben die übrigen Parameter auf den im Anhang A.1 dargestellten Werten. Bei diesem Anhang handelt es sich um die resultierende, optimierte Konfigurationsdatei mit der sich für Ereignisse mit großer, als auch mit niedriger Hit-Dichte gute Ergebnisse bei der Clustertrennung erzielen lassen.

[7] Die Quelle der *default* Konfigurationsdateien und des SOURCE EXTRACTOR-Paketes findet sich unter [14]

4.2.1 Rauschen

Wie in Kapitel 3.3.1 bereits erwähnt wurde, haben die aus der LCIO-Datei entnommen Daten kein Untergrundrauschen. Um SExtractor ohne Schwellen (*BACK_TYPE MANUAL*) zu verwenden wird eine konstante Schwelle (*BACK_VALUE*) von 0 eingestellt, mit der das Programm im Folgenden arbeitet. Weiterhin müssen trotzdem noch die Parameter für das automatische Ermitteln der Schwellen angegeben werden (*BACK_SIZE*, *BACK_FILTERSIZE*) - andernfalls liefert das Programm eine Fehlermeldung. Die entsprechenden Parameter lauten:

```
BACK_SIZE           1
BACK_FILTERSIZE     10
BACK_TYPE           MANUAL
BACK_VALUE          0.0
```

4.2.2 Trennen und die Anwendung von Filtern

Der Trennprozess innerhalb von Source Extractor kann durch die Angabe von verschiedenen Parametern beeinflusst werden. Diese lassen sich in drei verschiedene Gruppen einordnen: Parameter, die die Größe der erwarteten Hits betreffen, Parameter, die den Einsatz eines Filters auslösen und schließlich Parameter, die die Separation rund um das Setzen und Verteilen der neuen Schwellen (Kapitel 3.3.3) beeinflussen.

Die Wahl der erstgenannten Parameter ergibt sich aus den Gegebenheiten des Chips. Um eine Aussage über den Hit treffen zu können muss sich dieser aus mindestens einem Pixel mit Zeitwert und aus einem Pixeln mit Ladungsinformation zusammensetzten. Um beide Voraussetzungen mit großer Sicherheit zu erfüllen, sollte ein Hit mindestens 5 Pixel groß sein (Parameter: *DETECT_MINAREA*). Um noch größere Sicherheit zu haben wurde allerdings oft mit einem Minimum von 9 Pixeln gearbeitet. Hinsichtlich der maximalen Größe eines Hits gibt es keine Beschränkungen die bei der Trennung beachtet werden müssen. Vergleicht man Abbildung 4.3(a) mit 4.4(a) erkennt man den Effekt der Wahl des *DETECT_MINAREA* Parameters: Die im Original (Abb. 4.3(a)) zu sehenden kleinen Cluster unten rechts und die etwas größeren Cluster in der Mitte werden bei der Clustertrennung nicht berücksichtigt, wie sich am Ergebnis der Trennung erkennen lässt (Abb. 4.4(a)).[8]
Ein Pixelwert muss um den in *DETECT_THRESH* angegebenen Wert größer als das Rauschen sein, damit Source Extractor den Pixel bei der Clustertrennung betrachtet. Auch für diese Schwelle ist ein Wert von Null wünschenswert, doch erwartet SExtractor einen Wert größer als Null. So wurde 0.0001 gewählt, was bei Pixelwerten, die selten unter 100 liegen, den gleichen Effekt hat.(Gleiches gilt für die Schwelle *FILTER_THRESH*, ab der auf Pixel ein Filter angewendet wird.) *THRESH_TYPE ABSOLUTE* bewirkt hierbei, dass der für *DETECT_THRESH* angegebene Wert als Wert in der Einheit Pixelwerten interpretiert wird.

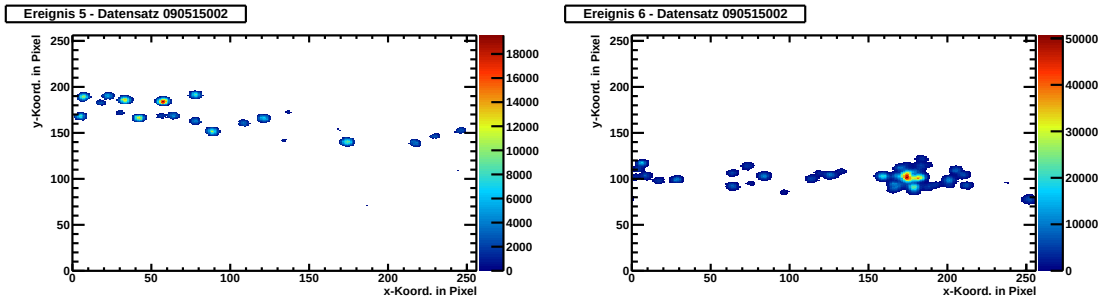
```
DETECT_MINAREA      9
DETECT_THRESH       0.0001

THRESH_TYPE         ABSOLUTE
```

Filter

Die Wahl des richtigen Filters und der zugehörigen Parameter hat großen Einfluss auf die Qualität der Clustertrennung. Dies lässt sich am Beispiel der Abbildungen 4.3 bis 4.5 erläutern. Hierzu wurde ein Ereignis (Ereignis 5, vgl. Abb. 4.3(a)) mit auseinander liegenden, sowie ein Ereignis (Ereignis 6, vgl. Abb. (b)) mit eng zusammen liegenden Hits gewählt. Mit dem bloßen Auge

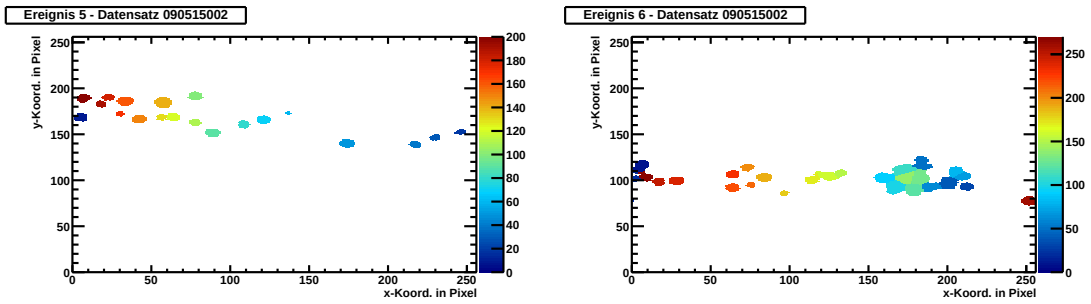
[8] Das Verschwinden des zentral liegenden kleinen Clusters in Abbildung 4.3(a), ist allerdings nicht durch die Wahl des *DETECT_MINAREA* bedingt. Dieser wird in der Analysekette für TPC-Daten schon vor der Trennung durch SExtractor verworfen.

würde man für Ereignis 5 bei der Trennung der Cluster ca. 20 Hits erwarten. Für Ereignis 6 ist diese Zahl schwerer zu bennenen und liegt bei etwa 35 zu erwartenden Hits.



(a) (b)

Abbildung 4.3: Beispiel für ein Ereignis mit (a) weit auseinander liegenden Hits, bzw. mit (b) dicht beieinander liegenden Hits (bereits interpolierte Zeitwerte, Ereignis 5 und Ereignis 6, Datensatz 090515002).



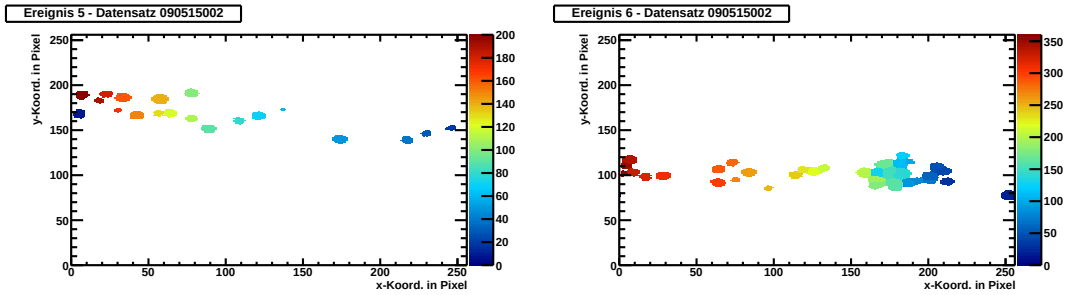
(a) (b)

Abbildung 4.4: Clustertrennung ohne Filter für ein Ereignis (a) weit auseinander liegenden Hits, bzw. mit (b) mit dicht beieinander liegenden Hits (Ereignis 5 und Ereignis 6, Datensatz 090515002).

Bei der Clustertrennung ohne Filter erhält man für Ereignis 5 die erwarteten 20 Hits (vgl. Abb. 4.4(a)). Für Ereignis 6 werden nur 27 Hits separiert, wobei die Diskrepanz zwischen diesem und dem erwarteten Wert durch überlappende, nicht richtig getrennte Hits entsteht. Lässt man SExtractor das Bild vor der Separation durch die Anwendung eines Filters glätten, verbessert sich das Ergebnis signifikant. Für das problematische Bild (Abb. 4.5(b)) werden nun - passend zu den Erwartungen - 37 Hits gefunden. Anderseits werden durch SExtractor auch die Cluster im Beispiel in Ereignis 5 in mehr Hits aufgetrennt (Abb. 4.5(a), erster und dritter Hit von rechts), sodass für dieses Ereignis mit 22 Hits ein Wert größer als der erwartete vorliegt. Dies macht bei Benutzung eines Filters (*FILTER Y*) das Anpassen weiterer SExtractor-Parameter erforderlich.

```
FILTER              Y
FILTER_THRESH       0.0001
```

Bei diesen Parametern handelt es sich an erster Stelle um die Wahl des eigentlichen Filters. Wie bei der Beschreibung der internen Arbeitsschritte von SExtractor (Kapitel 3.3.2) erwähnt, hat man bei der Nutzung von SExtractor die Wahl zwischen vier Filter-Arten: Gauß, Blockfunktion, Mexicanhat und abgeschnittener Gauß (Tophat). Da Filter mit Mexicanhatprofilen das Verhältnis zwischen hohen und niedrigen Pixelwerten verstärken, werden für diese bei der Clustertrennung



(a) (b)

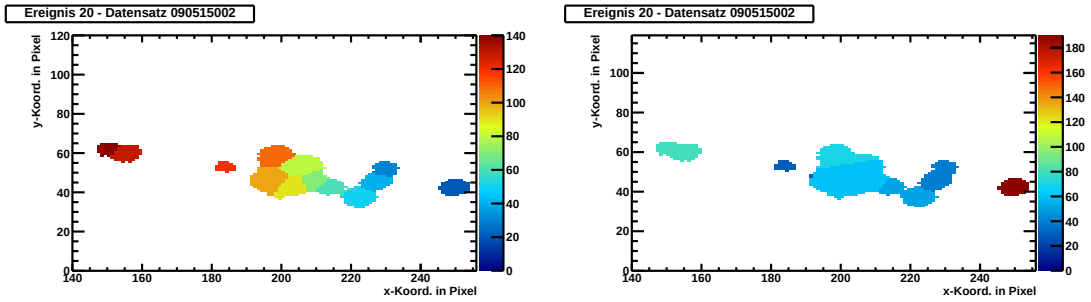
Abbildung 4.5: Clustertrennung mit Filter für ein Ereignis mit (a) weit auseinander liegenden Hits, bzw. mit (b) vielen überlappenden Hits (Ereignis 5 und Ereignis 6, Datensatz 090515002)

mehr Hits erwartet, als mit anderen Filtern.
Die Auswirkungen auf die Qualität der Clusterseparation aller vier Filter Arten werden in Abbildung 4.6 dargestellt. Man erkennt an der Auflösung des roten zwei-Hit-Clusters und an der des großen Clusters, dass für den Filter mit dem Mexicanhatprofil die beste Clustertrennung für viele und auch für wenige überlappende Hits erreicht wird. Anderseits neigen Filter mit solchen Profilen dazu auch Cluster, die nur aus einem Hit bestehen weiter zu trennen, wie bei der Diskussion von Abbildung 4.5(a) angemerkt wurde.
Dieser Umstand muss bei der Wahl des eigentlichen Filters bedacht werden. Da es verschiedenen Filter mit dem gewünschten Profil gibt, wurde nach dem Filter mit Mexicanhatprofil gesucht, der keine Hits weiter aufspaltet und gleichzeitig Cluster mit vielen Hits gut auflöst. Die einzelnen Filter unterscheiden sich hinsichtlich der Halbwertsbreite der zu Grunde liegenden Funktion und hinsichtlich der Größe der Matrix in der der Filter gespeichert wird. Da der in Abbildung 4.6(a) genutzte Filter (*mexhat_2.0_7*×*7*) alle abgebildeten Cluster gut trennt, allerdings noch ein-Hit-Cluster weiter aufgespalten werden, muss ein anderer gewählt werden. Es stellte sich heraus, dass mit wachsender Halbwertsbreite des dem Filter zu Grunde liegenden Profils immer weniger Cluster zu weit separiert werden. Somit wurde der Filter *mexhat_3.0_9*×*9* die beste Wahl, da mit diesem keine aus einem Hit bestehende Cluster noch weiter aufgespalten werden.

```
FILTER_NAME          mexhat_3.0_9x9.conv
```

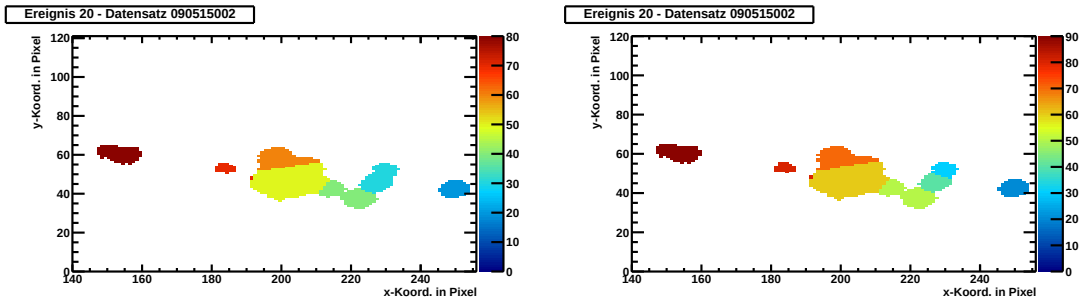
Trennen

Der eigentliche Trennenalgorithmus wird von drei Parametern kontrolliert: Der Anzahl der Schwellen die im Rahmen des „Multithresholding“ verteilt werden, der Art der Verteilung dieser Schwellen und dem Parameter, der festlegt welchen Prozentsatz des gesamten Clustervolumens ein Hit mindestens haben muss um als solcher gezählt zu werden.
Für die Anzahl der Schwellen des „Multithresholding“ (*DEBLEND_NTHRESH*) lässt sich die einfache Regel formulieren: Je größer der *DEBLEND_NTHRESH*-Wert, desto besser wird die Qualität der Trennung und desto langsamer wird der Trennalgorithmus. Da die der Trennung zu Grunde liegenden FITS-Dateien nur Bilder mit 256×256 Pixel (bzw. 296×296 Pixel für ein Bild mit einem Rand von 20 Pixel) enthalten, kann für *DEBLEND_NTHRESH* der größtmögliche Wert von 64 gewählt werden, da das Zeit-Argument für kleine Bilder vernachlässigbar ist.
Es stellte sich heraus, dass - mit der Festlegung eines hohen Wertes für *DEBLEND_NTHRESH* - die Wahl des *DETECT_TYPE*-Wertes keinen großen Einfluss mehr auf die Qualität der Trennung hat. Für *DETECT_TYPE PHOTO* wird in einem kleinen Teil der Testereignisse ein Hit mehr separiert. Dieses leicht bessere Ergebnis bei exponentiell verteilten Schwellen kann dadurch begründet werden, dass in verschiedenen Ansätzen für die Form der Ladungsverteilung eines Hits von einem dreidimensionalen Gauß, also von einer exponentiellen Abhängigkeit, ausgegangen wird.



(a) Filter: *mexhat_2.0_7×7*

(b) Filter: *gauss_2.0_3×3*



(c) Filter: *default*

(d) Filter: *tophat_2.0_3×3*

Abbildung 4.6: Beispiel für die Clustertrennung eines Ereignisses mit verschiedenen Filtern: (b) Gauß, (c) Blockfunktion, (a) Mexicanhat und (d) Tophat(Ereignis 20, Datensatz 090515002).

So würden die Schwellen für *DETECT_TYPE PHOTO* besser zur Pixelwerteverteilung nach der Interpolation der Zeitwerte passen.
Ein Parameter, der einen größeren Einfluss auf die Clustertrennung hat, ist der Anteil, den ein Hit am gesamten Clustervolumen mindestens haben muss (*DEBLEND_MINCONT*), um als einzelner Hit detektiert zu werden. Für den gewählten Filter (*mexhat_3.0_9x9.conv*) reicht hier ein *DEBLEND_MINCONT*-Wert von 0.0001 aus um alle Cluster in Hits zu separieren. Um allerdings den Parameter beim Auswechseln des Filters nicht ebenfalls ändern zu müssen wurde 10^{-8} für den *DEBLEND_MINCONT* gewählt, da sich bei Tests ergab, dass eine Unterschreitung des *DEBLEND_MINCONT*-Wertes bei Verwendung von Mexicanhat-Filtern keine negativen Auswirkungen hat. Wird der Wert jedoch überschritten, werden weniger Hits getrennt bis jeder Cluster (*DEBLEND_MINCONT* 1) nur noch als ein Hit gesehen wird.

```
DETECT_TYPE         PHOTO

DEBLEND_NTHRESH     64
DEBLEND_MINCONT     0.00000001
```

4.2.3 Bereinigen

Beim Bereinigen wird für jeden Hit überprüft ob man diesen mit SOURCE EXTRACTOR ohne den Einfluss der benachbarten Hits gefunden hätte, indem eine Moffat-Kurve an jeden Hit angefittet und der so berechnete Beitrag der Nachbarn abgezogen wird (vgl. Kapitel 3.3.3, Gleichung 3.2). Der Grund warum ein Hit ohne seine Nachbarn nicht mehr als Hit ansehen werden sollte ist, dass dieser für sich genommen unter eine bestimmte Schwelle im Programm fällt. Da SEXTRACTOR allerdings ohne Schwelle betrieben wird, werden per se nur solche Hits bereinigt, die ohne ihre Nachbarn zwischen eine der 64 durch das „Multithresholding“ gesetzten Schwellen fallen. Diese

sollten einen verschwindenden Prozentsatz ausmachen.
Vergleicht man so das Ergebnis der Clustertrennung mit und ohne Bereinigung stellt man fest, dass sich diese Erwartung erfüllt und sich nahezu keine Unterschiede in den Katalogen mit und ohne Bereinigen finden. Aus diesem Grund wurde auf die Bereinigung verzichtet (*CLEAN N*). Auch hier muss der zusätzliche Parameter gesetzt werden *CLEAN_PARAM*) um keinen Fehler zu erhalten.

```
CLEAN               N
CLEAN_PARAM         1.0
```

4.3 Einlesen der getrennten Cluster

Nachdem SExtractor genutzt wurde um die Cluster in den von LCIO nach FITS formatierten Ereignissen zu trennen, muss die resultierende Katalogdatei zurück nach LCIO konvertiert und aus dieser die eigentlichen Hits rekonstruiert werden. Im LCIO-Format besteht ein Hit aus einer Liste aller Pixel, die zu diesem gehören. Um diese Rekonstruktion vornehmen zu können wird das entsprechende Ereignis in der LCIO-Datei aufgegriffen, und gleichzeitig der Katalog, mit den aus der Clustertrennung resultierenden Hits, aufgerufen. Da im Katalog die Hit-Zentren aufgelistet sind, aber keine Auflistung aller Pixel eines Hits, ist es erforderlich die Zuordnung jedes Pixels zum jeweiligen Hit beim Speichern der Hits anders vorzunehmen.
Es wurden zwei Ansätze verfolgt: Zunächst durchläuft jeder Pixel eines Ereignisses die Liste der Hitzentren im Katalog. Im ersten Ansatz wird ein dreidimensionaler Gauß für jeden Hit erstellt und jeder Pixel dem Hit zugeordnet, dessen Amplitude der Gaußfunktion an der Position des Hits am größten ist. Der alternativen Ansatz ordnet Pixel dem Hit zu, dessen Zentrum zum gerade betrachtetem Pixel das nächste ist.

4.3.1 Ansatz: Zuordnung nach größter Amplitude

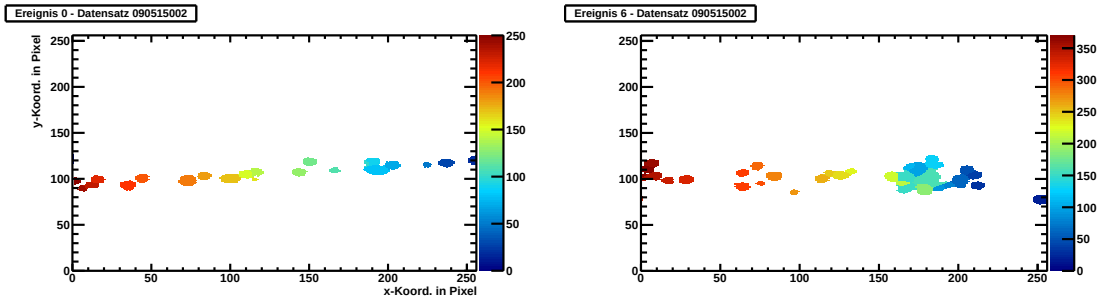
Bei der Zuordnung der Pixel zu ihren Hits über eine dreidimensionale Gaußverteilung (Gleichung 4.2), wurde für jeden Hit aus den Katalog-Parametern zur Position des Hit-Zentrums (*X_IMAGE* und *Y_IMAGE*), zur Halbwertsbreite (*FWHM*) und zum höchsten Pixelwert des Hits (*FLUX_MAX*) eine Gaußfunktion erstellt. Dieser Ansatz ist möglich da die Form der Hits einer Gaußverteilung ähnlich ist.

$$A = FLUX\_MAX \cdot e^{-\frac{\left((x_{\mathrm{Pixel}} - X\_IMAGE)^2 + (y_{\mathrm{Pixel}} - Y\_IMAGE)^2\right)}{\frac{FWHM^2}{4\ln(2)}}} \tag{4.2}$$

Danach durchläuft jeder Pixel die Liste der Gaußverteilung und wird dem Hit zugeschlagen, dessen Gaußfunktion die größte Amplitude A an Stelle des Pixels hat. Bei diesem Ansatz kommt es allerdings vor, dass die Verteilungen einer wenigen Hits die der anderen überlagern und so Pixel Hits zugeschlagen werden, zu denen diese eigentlich nicht gehören. Dieser Effekt tritt bei Ereignissen mit wenigen Hits kaum auf (Abb. 4.7(a)) und äußerst sich immer stärker, je mehr Hits auf einem kleinen Gebiet liegen (vgl. Abb. 4.7(b), der große Cluster sowie die Cluster links).

4.3.2 Ansatz: Zuordnung nach kleinstem Abstand

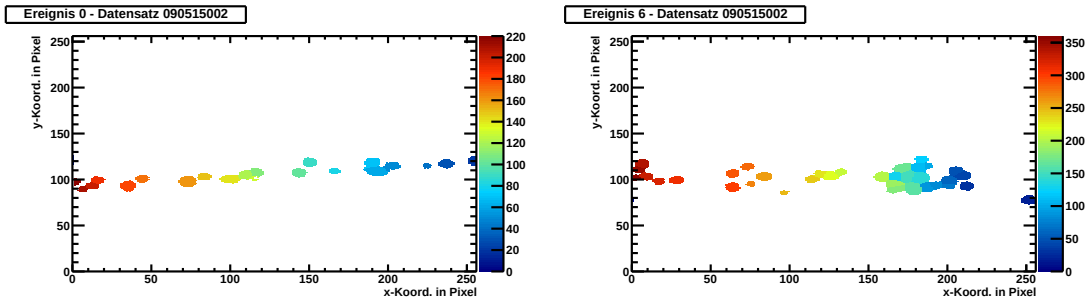
Da auch eng zusammen liegende Hits korrekt getrennt werden sollen, wurde zur Lösung des Problems des Ansatz „Zuordnung nach größter Amplitude“ ein weiterer, einfacherer Ansatz erprobt: Anstatt einen Pixel dem Hit mit der größten Amplitude am Ort des Pixels zuzuordnen, wurde die Zuordnung der Pixel über den Abstand zum Zentrum des Hits versucht. Da beim vorherigen Ansatz von gaußförmigen Hits ausgegangen wurde, ist die Annahme von runden Hit-Grundflächen naheliegend. Die Entfernung Hit-Zentrum - Pixel für diesen Ansatz wird über die Gleichung 4.3 berechnet. Die hierzu nötigen Koordinaten wurden dem *X_IMAGE* und *Y_IMAGE* Werten des Kataloges entnommen.



(a) (b)

Abbildung 4.7: Mit dem „größte Amplitude“ Ansatz getrennte Hits für zwei verschiedene Ereignisse - (a) Hits für das Ereignis aus Abb. 4.2(a) - (b) Hits für das Ereignis aus Abb. 4.3(b)

$$D = \sqrt{(x_{\text{Pixel}} - X\_IMAGE)^2 + (y_{\text{Pixel}} - Y\_IMAGE)^2} \tag{4.3}$$



(a) (b)

Abbildung 4.8: Mit dem „kleinster Abstand“ Ansatz getrennte Hits für Ereignis 0 und Ereignis 6 (Datensatz 090515002)

Auch bei diesem Ansatz kann es vorkommen, dass Pixel dem falschen Hit zugeordnet werden. Hier sind allerdings nur Pixel betroffen, die näher am Zentrum eines anderen Hits liegen, als dem zu dem Zentrum ihres eigentlichen Hits. Dieser Effekt tritt selten, und für Ereignisse mit wenigen, getrennt liegenden Hits in dem Maße wie auch die falsch Zuordnung beim anderen Ansatz auf (vgl. Abb. 4.7(a)) mit Abb. 4.8). Im Gegensatz zum Ansatz „Zuordnung nach größter Amplitude“ wird dieser Effekt allerdings mit steigender Hit-Dichte nicht stärker (vgl. die Abbildungen 4.7(b) und 4.8(b)).

4.4 Die MarlinTPC Analysekette mit SExtractor

Um sowohl bei Ereignissen mit einer großen als auch einer geringen Hitdichte eine optimale Clustertrennung zu erzielen wird der Ansatz „Trennung nach kürzestem Abstand“ verwendet. Hierbei muss bedacht werden, dass auch bei diesem Algorithmus eine Zuordnung von Pixel zum falschen Hit geschieht. Am Beispiel von Abbildung 4.6 lässt sich dies gleich drei mal für einige Pixel im großen Cluster beobachten, die einem Hit im Cluster außerhalb zugeschlagen werden. Da dieses Problem allerdings nur in wenigen Fällen auftritt, wurde dieses vernachlässigt und der Ansatz trotzdem in den Separationsprozess eingebaut.

Weiterhin musste der Separationsprozess in die MarlinTPC-basierte Analysekette für TPC-Daten eingebaut werden. So wurde das Schreiben nach FITS, das Trennen der Cluster mit Source Extractor, das Einlesen des Kataloges und die darauf basierende Zuordnung der Pixel zu den Hit-Zentren in jeweils einem MarlinTPC-Prozessor realisiert. Diese sollen im folgenden kurz vorgestellt werden:

SEWriteSlcioToFitsProcessor Ein Prozessor, der dazu dient das aktuelle Ereignis nach FITS zu konvertieren und dabei die Zeitwerte zu interpolieren. Als Eingangsparameter erhält er die Pixelwerte des betreffenden Ereignisses und er gibt eine FITS-Datei aus.

SExtractClusterProcessor Dieser Prozessor wendet Source Extractor zur Clustertrennung auf das durch den *SEWriteSlcioToFitsProcessor*-Prozessor erstellte FITS an. Als Eingangsparameter erhält dieser Prozessor die FITS-Datei und die Konfigurationsdateien, die für SExtractor benötigt werden. Alternativ können nur die von den Konfigurationsdateien benötigte Parameter an den Prozessor übergeben werden und dieser erstellt die Konfigurationsdateien selbst. Die Ausgabe dieses Prozessors ist eine Katalogdatei mit Daten zu den getrennten Clustern.

SEWriteCatToSlcioProcessor Dieser Prozessor erhält als Eingabe den durch den *SExtractClusterProcessor*-Prozessor geschriebenen Katalog und trägt die dort angegebenen Daten in die LCIO-Datei ein.

SETimePixClusterSeparatorProcessor Der *SETimePixClusterSeparatorProcessor* nimmt die durch den *SEWriteCatToSlcioProcessor* geschriebenen Daten zu den Hits und schreibt über die Zuordnung „Hit-Zentren - Pixel“ Hits in die LCIO-Datei.

Kapitel 5

Ergebnisse

Im folgenden Kapitel wird auf die mit SOURCE EXTRACTOR erzielten Resultate eingegangen um deren Qualität zu bewerten. Dazu wurden zum einen die mit SEXTRACTOR erzielten Ergebnisse mit den Ergebnissen des bisher genutzten Trennalgorithmus verglichen. Zu dem wurde SEXTRACTOR zur Clustertrennung von simulierten Daten mit einer bekannten Anzahl von Hits genutzt, und auch hier wurden die erzielten Ergebnisse mit den Ergebnissen des bisher verwendeten Clustertrenners verglichen.

5.1 Vergleich mit dem aktuell genutzten Trennalgorithmus

In Kapitel 2.3 wurde beschrieben, dass der aktuell verwendete Trennalgorithmus Probleme hat, Cluster richtig zu separieren, die über keine eindeutige Schwerpunktsachse verfügen (vgl. Abb. 2.6). Dieses Problem wird in den Abbildung 5.1 zusammengefassten Ereignissen deutlich und es sollte das Ziel sein dass dieses Problem mit dem neuen, SEXTRACTOR-basierten Clustertrenner nicht mehr auftritt.

5.1.1 Vergleich auf Basis einzelner Ereignisse

Um festzustellen ob dieses Problem nicht mehr auftritt, werden beide Trennalgorithmen am Beispiel des in Abbildung 5.1(a) abgebildeten Ereignisses (Ereignis 0) betrachtet. So erwartet man mit Kenntnis dieses Ereignisses bei der Clustertrennung etwa 22 Hits, die auch mit SEXTRACTOR erreicht werden (Abb. 4.8). Der aktuelle Separationsalgorithmus liefert entgegen der Erwartung für dieses Ereignis 21 Hits, die im Bereich überlappender Hits ungenau bis falsch separiert wurden (Abb. 5.1(c)). Allein liegende Hits werden erkannt und korrekt getrennt. Dies ist auch im Beispiel von Abbildung 5.1(c) für verschiedene kleinere Cluster zu sehen (z.B. der blaue Cluster links sowie der orange rechts), die in zu wenige Hits getrennt werden.
Für Cluster, die sich aus einer großen Zahl überlappenden Hits zusammen setzten, liefert der aktuelle Clustertrenner kein sinnvolles Ergebnis. So wird zum Beispiel der große blaue Cluster in Abbildung 5.1(d) nur in 6 Hits mit zum Teil unrealistischer Form zerlegt, obwohl man für diesen (Abb. 5.1(b)) eine Hit-Anzahl in der Größenordnung von 15 oder mehr Hits erwartet hätte. Insgesamt lässt sich für diesen Algorithmus bei der Betrachtung von einzelnen Ereignissen sagen, dass die Resultate immer schlechter werden, je mehr Hits sich in einem Cluster überlagern. Im Vergleich dazu erzielt die Clustertrennung mit SOURCE EXTRACTOR für den großen Cluster aus Ereignis 6 (Abb. 4.5(b)) eine Separation in 17 Hits.
Zusammengefasst führt der Vergleich der beiden Trennalgorithmen an einzelnen Ereignissen zu der Aussage, dass der SEXTRACTOR-basierte Algorithmus wesentlich mehr Cluster in Hits trennt als der bisher verwendete. Dabei sehen die mit SEXTRACTOR erhaltenen Resultate realistischer aus als die, die man mit dem alten Algorithmus erhält. Das Eingangs erwähnte Problem bei der Clustertrennung von solchen mit keiner klaren Symmetrieachse besteht nicht mehr. Dabei gilt

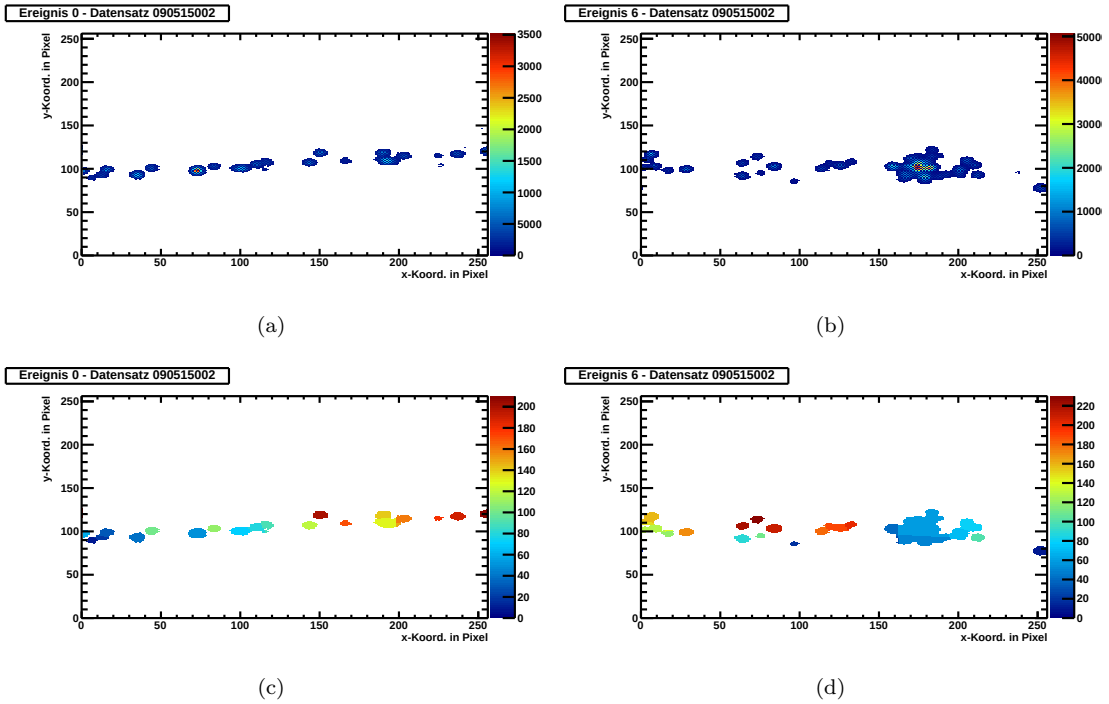


Abbildung 5.1: Beispiele für die Clustertrennung mittels des aktuellen Separation-Algorithmus und die zugehörigen original Teilchenspuren: (a) und (c) Ereignis mit Clustern bestehend aus wenigen Hits (Ereignis 0, Datensatz 090515002); (b) und (d) Ereignis mit dicht beieinander liegenden Hits (Ereignis 6, Datensatz 090515002).

allerdings die Einschränkung, dass SOURCE EXTRACTOR durch den Ansatz: „Zuordnung der Pixel zu ihren Hits nach dem kleinsten Abstand“ manchen Hits Pixel zuschlägt die nicht zu diesen gehören.

Um diese Aussage zu einer quantitativen Aussage zum Vergleich beider Algorithmen zu machen wird eine größere Statistik betrachtet. Durch Analyse von ca. 100.000 Ereignissen mit zwei MarlinTPC Analyseketten, einer mit dem aktuellen Clustertrenner und einer mit einem auf SEXTRACTOR basierendem Trennalgorithmus, wird eine solche erstellt.

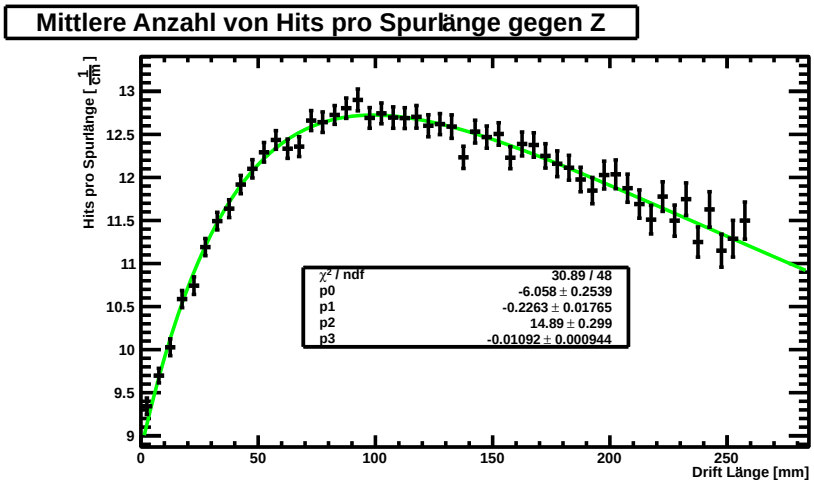
5.1.2 Vergleich der durchschnittlich pro Spurlänge gefunden Hits

Als erste Vergleichsgröße wird die Anzahl der rekonstruierten Hits pro Spurlänge in Abhängigkeit der Driftstrecke z betrachtet. In Abbildung 5.2 ist die eben genannte Vergleichsgröße für beide Trennalgorithmen zu sehen. Hierbei fällt für beide Graphen auf, dass die Anzahl der separierten Hits zunächst steil ansteigt um danach wieder abzufallen. Der Anstieg liegt darin begründet, dass eng zusammen liegende Elektronen auf einer kurzen Driftstrecke nicht die Möglichkeit haben auseinander zu driften und so nur schwer separiert werden können. Somit wird die Wahrscheinlichkeit Hits zu separieren mit steigender Driftstrecke des zugehörigen Primärelektrons immer größer. Andererseits steigt mit größerer Driftstrecke auch die Wahrscheinlichkeit dafür, dass ein Elektron von einem Gasmolekül wieder gebunden wird. Dies erklärt den abfallenden Teil der Kurve. Ein Ansatz für eine Funktion die beide Effekte beschreibt ist:[9]

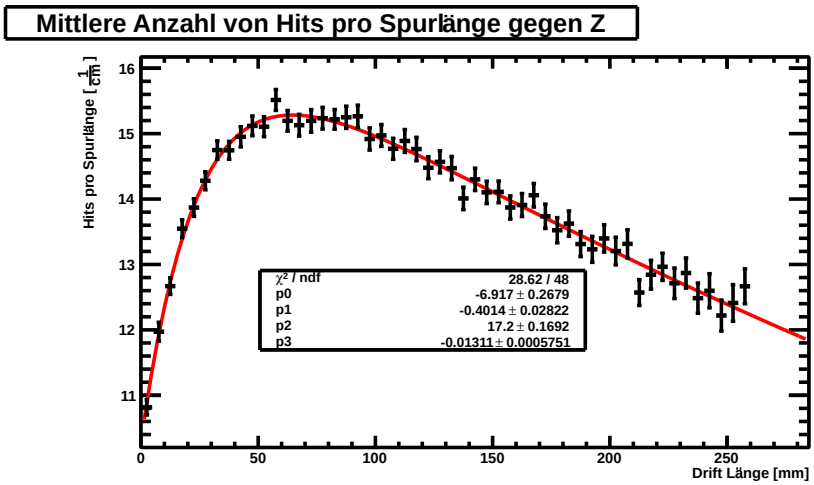
$$H = p_0 \cdot e^{p_1 \cdot z} + p_2 \cdot e^{p_3 \cdot z} \tag{5.1}$$

Da die beiden Effekte für den Anstieg und das Abfallen der Funktion auf exponentielle Prozesse

[9] Für eine ausführlichere Diskussion des Ansatz vgl. [5]



(a)



(b)

Abbildung 5.2: Hits pro Spurlänge bei einer minimalen Hitgröße von 9 Pixeln, für: (a) den aktuellen Trennalgorithmus und (b) den SExtractor basierten Trennalgorithmus. Die Parameter stehen für: p_0*:* Die Differenz zwischen der gemessenen Elektronenanzahl für Driftstrecken von 0 cm und der Anzahl der maximal gemessenen Elektronen, die man für keinen Primärerlektronen Einfang erwarten würde; p_1*:* Separationsrate in cm^{-1}; p_2*:* Maximale Anzahl von Hits pro cm; p_3*:* Anlagerungsrate (temporärer Elektroneneinfang) in cm^{-1}

zurück zu führen sind, kann eine mögliche Fit-Funktion aus zwei Exponentialfunktionen - einer für der Anstieg der Hit-Zahl im Bereich kleiner und einer für den Abfall der Hit-Zahl hin zu größeren Driftstrecken - beschrieben werden. Für die Clustertrennung bei einer minimalen Hitgröße von 9 Hits (Abb. 5.2) sollen nun die durch den Fit an die Daten angepassten Parameter diskutiert werden.

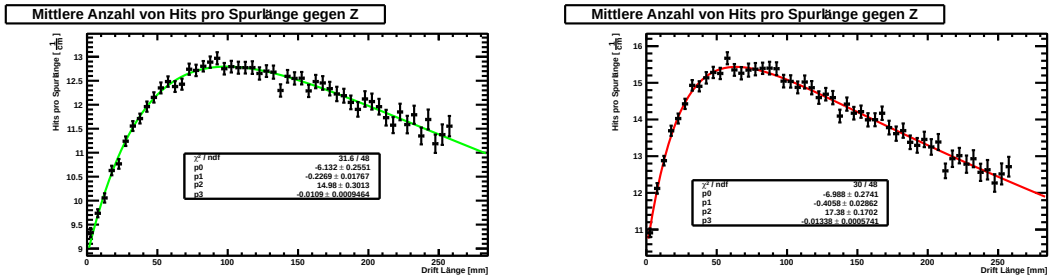
Aus den angepassten Fitwerten können Informationen über den Detektor und über die Qualität des Clustertrenners entnommen werden. Der y-Achsenabschnitt liefert die Anzahl der Hits, die für eine Driftstrecke von 0 cm detektiert werden können. Dieser setzt sich aus der Summe der Fit-Parameter p_0 und p_2 zusammen. Hier ist p_2 die Anzahl der Hits, die bei der Clustertrennung - wenn es keinen Elektronen-Einfang geben würde und alle Hits separiert werden könnten - gefunden würden. p_0 ist somit der Unterschied zwischen der maximal möglichen Anzahl an Hits und der tatsächlichen Zahl getrennter Hits bei einer Drift von 0 cm.
Der Parameter p_2 liefert für den SExtractor-basierten Trennalgorithmus $(17{,}2 \pm 0{,}2)$ Hits als obere Grenze für die Clusterseparation bei gut 10 separierten Hits für eine Driftstrecke von 0 cm. Der Wert für den Parameter p_2 des aktuell verwendeten Clustertrenners liegt bei $(14{,}9 \pm 0{,}3)$ Hits mit knapp 9 separierten Hits bei minimaler Drift. Somit trennt Source Extractor einzelne Hits schon bei kleinen Driftstrecken besser als der bisher verwendete Separationsalgorithmus und hat auch eine höhere Anzahl von maximal trennbaren Hits pro Spurlänge.
Der Literaturwert für die Anzahl der ausgelösten Elektronen in der TPC beträgt für das verwendete Gas 29,5 Elektronen cm^{-1} bei 15,13 Wechselwirkungen cm^{-1}.[10] Somit würde ein Wert von 29,5 Hits für den Parameter p_2 einem idealen Aufbau mit idealem Clustertrenner entsprechen. Die Differenz des „idealen“ und des aktuell erhaltenen Wertes lässt an dieser Stelle darauf schließen, dass sich die Anzahl der gefunden Hits noch weiter steigern lassen sollte.

Der bisher noch nicht betrachtete Exponent der ansteigenden Exponentialfunktion ist die Separationsrate. Sie ergibt sich aus der Separationkraft der Trennung und dem Diffusionskoeffizienten während der Drift. Da mit beiden Analyseketten auf den gleichen Daten gearbeitet wurde, ist letzterer Wert in beiden Graphen gleich und es lässt sich aus Parameter p_1 eine Aussage über das Verhältnis der Separationskraft beider Algorithmen machen. So ist die Separationsrate von Source Extractor für die Analyse der vorliegenden Teilchenspuren mit $(0{,}40 \pm 0{,}03)\,\mathrm{cm}^{-1}$ größer als die Separationsrate des bisher verwendeten Algorithmus von $(0{,}23 \pm 0{,}02)\,\mathrm{cm}^{-1}$. Dies stützt die bisher gewonnenen Eindrücke hinsichtlich der Separationskraft der beiden Trennalgorithmen.

Als letztes soll der Parameter p_3 der abfallenden Exponentialfunktion betrachtet werden. Bei diesem handelt es sich um die Anlagerungsrate der Primärelektronen, die vom Gas im Detektor abhängt und so für beide Trennalgorithmen etwa gleich groß und mit dem simulierten Wert für die Einfangsrate im Detektor kompatibel sein sollte. Dies ist für beider Werte der Fall. Die angepassten Werte - der an die mit dem bisher verwendetem Clustertrenner erzeugten Daten $((0{,}011 \pm 0{,}001)\,\mathrm{cm}^{-1})$ und der entsprechende Wert auf Grundlage der Anwendung von SExtractor $((0{,}013 \pm 0{,}001)\,\mathrm{cm}^{-1})$ - liegen in der gleichen Größenordnung wie der erwartete Wert für 10ppm Sauerstoff im Detektor von $0{,}01\,\mathrm{cm}^{-1}$.[11]

Für die Fitparameter, die an die Daten, für die eine minimale Hitgröße von 5 Pixeln voraus gesetzt wurde, angepasst wurden, ähnelt das Resultat dem der gerade diskutierten Analyse (Abb. 5.3). Hier liegen die Separationsraten und die Zahl der maximal trennbaren Hits beider Algorithmen leicht über den Werten der Analyse für eine minimale Hitgröße von 9 Pixeln. Dies verwundert nicht, da bei einer kleineren Hitgröße potentiell mehr Cluster gefunden und als Hit interpretiert, beziehungsweise getrennt werden können.
Abschließend lässt sich zum Vergleich der beiden Trennalgorithmen auf Basis der separierten Hits pro Spurlänge in Abhängigkeit der Drift-Strecke z sagen, dass die Clustertrennung mit Source Extractor besser funktioniert als mit dem bisher verwendeten Algorithmus. Damit festigt sich der Eindruck, der beim Vergleich der beiden Algorithmen, anhand einzelner Ereignisse gewonnen wurde.

[10] Simulation mit HEED, Details siehe [10]
[11] Simulation mit Magboltz, weitere Details siehe [6] und [5]

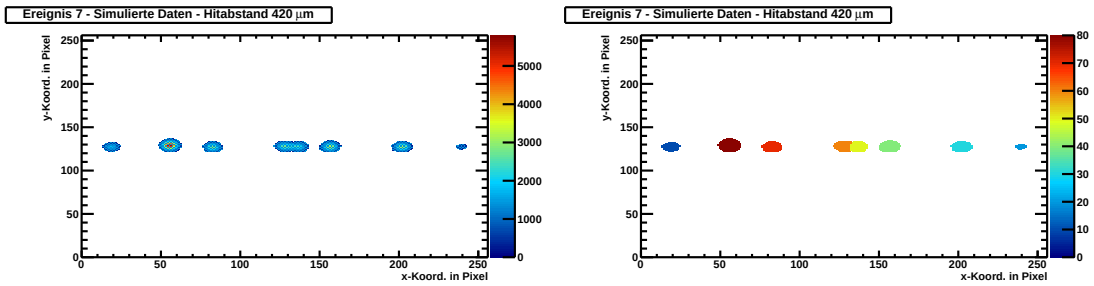


(a) (b)

Abbildung 5.3: Hits pro Spurlänge bei einer minimalen Hitgröße von 5 Pixeln, für: (a) den aktuellen Trennalgorithmus und (b) den SExtractor basierten Trennalgorithmus. Die Parameter stehen für: p_0*:* Die Differenz zwischen der gemessenen Elektronenanzahl für Driftstrecken von 0 cm und der Anzahl der maximal gemessenen Elektronen, die man für keinen Primärerlektronen Einfang erwarten würde; p_1*:* Separationsrate in cm^{-1}; p_2*:* Maximale Anzahl von Hits pro cm; p_3*:* Anlagerungsrate (temporärer Elektroneneinfang) in cm^{-1}.

5.2 Vergleich auf Grundlage simulierter Daten

Neben dem Vergleich der beiden Clustertrenner mit Messdaten wurde auch ein Vergleich der beiden Algorithmen auf Basis eines simulierten Datensatzes mit bekannter Hit-Zahl untersucht.[12] So kann beurteilt werden ob zu wenige oder vielleicht sogar zu viele Hits am Ende des Separationsprozesses stehen.

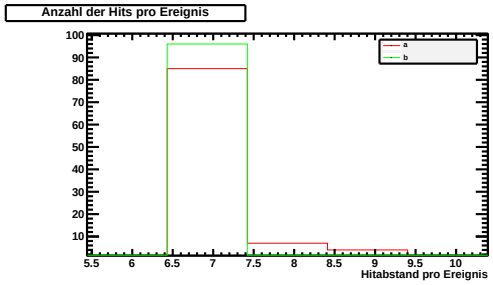


(a) (b)

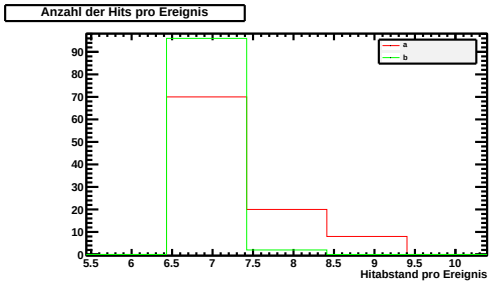
Abbildung 5.4: Ereignis 6 aus simulierten Daten, einmal der original Datensatz des Ereignisses (a) und einmal nach der Cluster-Separation mit SExtractor 5.4(b). Die überlappenden Hits wurden mit einem Abstand von 420 µm zueinander simuliert.

Es wurden verschiedene Datensätze mit jeweils 100 Ereignissen simuliert. Jedes dieser Ereignisse enthält eine normierte, für alle Ereignisse identische, Spur mit 6 Hits. Zu diesen wurden noch zwei weitere Primärelektronen im Abstand von 140 µm bis 560 µm in 70 µm Schritten hinzugefügt. Dabei wurden die Spuren so simuliert, dass der Effekt der Bindung von Elektronen in der GEM stark unterdrückt wurde. Die resultierenden Teilchenspuren bestehen so größtenteils aus 8 Hits, Spuren mit nur 7 oder gar 6 Hits treten äußert selten auf. Die simulierte Gasverstärkung entspricht hier allerdings nur in der Größenordnung den gemessenen Daten. Ein Beispiel für ein solches Ereignis und das Ergebnis der Clustertrennung mit dem SExtractor-basierten Algorithmus ist in Abbildung 5.4 zu sehen.

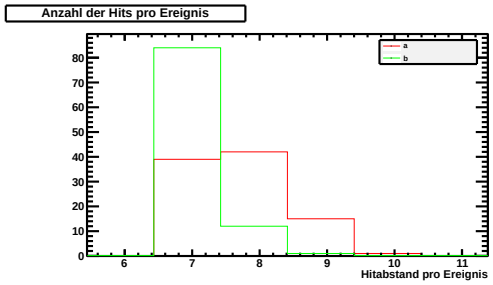
[12] Die Simulation wurde mit Garfield++ durchgeführt, weitere Details siehe [10]



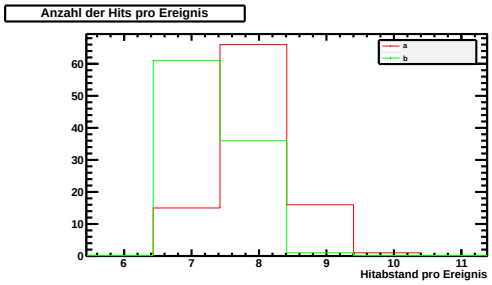
(a) Abstand von 210 µm



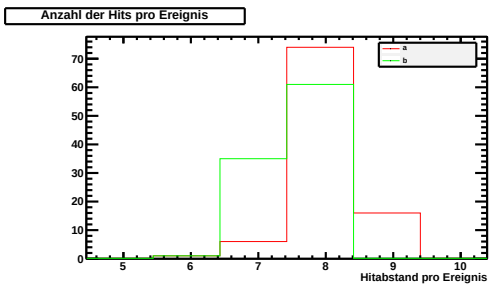
(b) Abstand von 280 µm



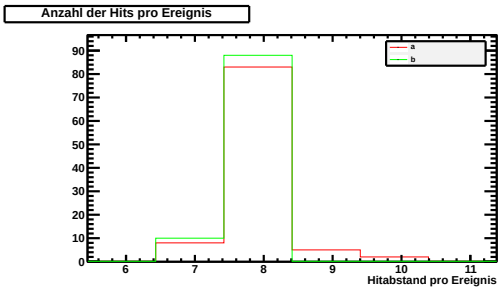
(c) Abstand von 350 µm



(d) Abstand von 420 µm



(e) Abstand von 490 µm



(f) Abstand von 560 µm

Abbildung 5.5: Hits pro Ereignis für simulierte Daten. Die rote Datenreihe (*a*) basiert in jedem Histogramm auf der Analyse von simulierten Daten mit SOURCE EXTRACTOR, die grüne Datenreihe (*b*) auf der gleichen Analysekette mit dem bisher genutzten Clustertrenner.

Die simulierten Daten wurden wie im vorherigen Kapitel jeweils mit SOURCE EXTRACTOR und dem bisher genutzten Clustertrenner analysiert um zu untersuchen welcher Algorithmus bessere Ergebnisse liefert. In Abbildung 5.5 sind diese Ergebnisse zu sehen.
Für einen Abstand von 140 µm können mit SEXTRACTOR ca. (1 ± 1) % der überlappenden Hits getrennt werden, während sich mit dem andere Algorithmus erst bei einem Abstand von 210 µm (1 ± 1) % der überlappenden Hits trennen lassen. Der neue Algorithmus mit SEXTRACTOR trennt hier bereits (12 ± 4) % der sich überschneidenden Hits, allerdings (5 ± 2) % davon in mehr als 8 Hits (Abb. 5.5(a)). Der bisher verwendete Clustertrenner übersteigt mit (13 ± 4) % die 10% Marke für den Anteil der separierten Hits an den separierbaren Hits erst bei einen Hitabstand von 350 µm (Abb. 5.5(c)). Hier tritt auch für diesen Algorithmus in (1 ± 1) % der Fälle eine falsche Separation auf, während dieser Wert für den neuen Algorithmus bei ca. (16 ± 4) % liegt. Allerdings weist der SEXTRACTOR-basierte Algorithmus bei diesem Abstand auch eine Separation in (60 ± 8) % der Fälle auf. Ab diesem Wert sinkt der Anteil der Falsch-Separationen wieder ab, was im Folgenden diskutiert werden soll. Mit dem bisher verwendeten Algorithmus beginnt die Zahl der separierten Hits ab diesem Abstand stark zu steigen, so verdreifacht sich diese nahezu auf eine Separation von (38 ± 6) % der Hits für den nächsten Hitabstand von 420 µm und steigt weiter an, bis sie bei Hitabständen von 560 µm den neuen Clustertrenner erreicht hat (Abb. 5.5(f)).

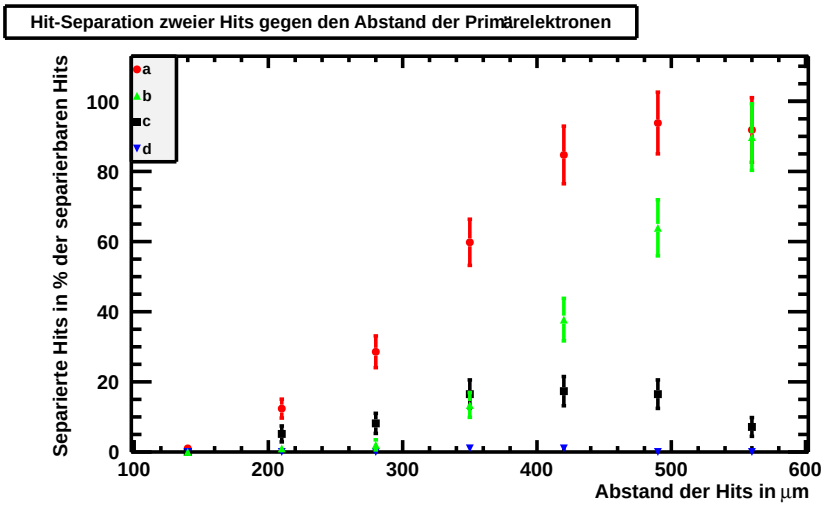


Abbildung 5.6: Hit-Separation der beiden überlappenden Hits gegen den Abstand der Primärelektronen - Hierbei steht
(*a*) für insgesamt separierte Hits mit SEXTRACTOR
(*b*) für insgesamt separierte Hits mit dem bisher verwendeten Algorithmus
(*c*) für zu weit separierte Hits mit SEXTRACTOR
(*d*) für zu weit separierte Hits mit dem bisher verwendeten Algorithmus

Der Umstand, dass SOURCE EXTRACTOR sich überschneidende Hits zum Teil in zu viele Hits trennt, soll jetzt diskutiert werden. In Abbildung 5.6 wurde die Hit-Separation der beiden überlappenden Hits gegen den Abstand der zu Grunde liegenden Primärelektronen aufgetragen.
Der Anteil der mit SEXTRACTOR separierten Hits (Datenreihe *a*) an den separierbaren Hits steigt schneller als die gleiche Größe (Datenreihe *b*) für den bisher verwendeten Trennalgorithmus. Problematisch hierbei ist allerdings der Anteil der mit SEXTRACTOR in zu viele Hits separierten Cluster im Vergleich zu den insgesamt gemachten Separationen (Datenreihe *c*). Dieser Anteil steigt zunächst ebenfalls schneller als der Prozentsatz der mit dem alten Clustertrenner überhaupt separierten Hits, hat ein Maximum im Intervall zwischen 280 µm und 420 µm und sinkt dann wieder ab. Im Gegensatz dazu ist der Prozentsatz der mit dem bisher verwendeten Trennalgorithmus zu weit getrennten Hits vernachlässigbar (Datenreihe *d*).
Es kann anhand dieses Graphen festgehalten werden, das SOURCE EXTRACTOR als Clustertrenner

sich überschneidende Hits früher trennen kann, als der bisher verwendete Algorithmus. Dabei tritt jedoch eine Falschtrennung in einen Hit zu viel auf (in $(8 \pm 3)\,\%$) der Fälle in zwei Hits zu viel). Die fehlerhafte Trennung hängt mit dem Abstand der Primärelektronen und damit dem Abstand der Hitzentren der eigentlich vorliegenden Hits funktional zusammen. Eine Hypothese für den Zusammenhang ist, dass dieser durch den verwendeten Filter hervorgerufen wird. Sie soll anhand von gemessenen Daten im folgenden Kapitel aufgegriffen werden.

5.3 Hitabstände der gemessenen Daten

Weiterhin kann aus Abbildung 5.6 entnommen werden, dass die Clustertrennung mit SOURCE EXTRACTOR schon im Intervall von Hitabständen zwischen 140 µm und 210 µm messbar ist, während dies für den bisher genutzten Algorithmus erst im Intervall zwischen 210 µm und 280 µm zutrifft. Diese an simulierten Daten gewonnene Aussage lässt sich auch an den gemessenen Daten verifizieren: Abbildung 5.7 zeigt für den analysierten Datensatz aus der Messung, wie viele Hits welchen Abstand zueinander haben. Hierbei steht die Datenreihe (a) für die Abstände der mit SEXTRACTOR getrennten Hits und (b) für die mit dem alten Trennalgorithmus getrennten Hits. Es kann an der Abbildung gesehen werden, dass mit dem neu erprobten Trennalgorithmus im Bereich kleiner Hitabstände mehr Hits als mit dem alten Algorithmus getrennt werden. Dabei fällt auf, dass die Kurve des SEXTRACTOR-basierten Clustertrenners schon bei etwa 350 µm das Plateau erreicht und dort in eine Oszillation übergeht. Dies ist für den anderen Algorithmus erst bei ca. 500 µm Hitabstand der Fall.[13]
In der Abbildung ist außerdem für die mit SEXTRACTOR getrennte Datenreihe eine Schulter bei etwa 280 µm zu sehen. Bei dieser kann es, sich ähnlich wie bei der Schulter in der Datenreihe des anderen Algorithmus zwischen 350 µm und 400 µm, um eine Oszillation vor Erreichen des Plateau handeln.

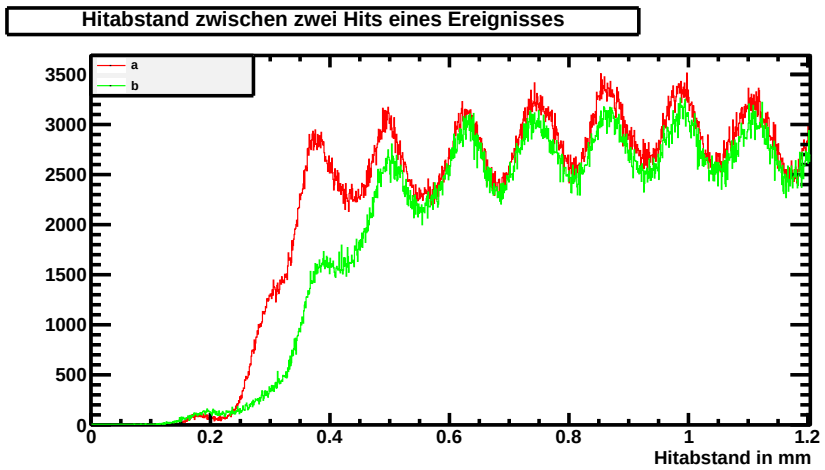


Abbildung 5.7: Hitabstand zwischen zwei Hits pro Ereignis - (a) für den SEXTRACTOR-basierten Algorithmus, (b) für den bisher verwendeten Algorithmus

[13] Die Oszillation ist im GEM Muster begründet.

5.4 Die Analysekette mit verschiedenen Filtern

In Abbildung 5.7 befindet sich ein bisher noch nicht diskutierter, kleiner Peak mit einem Maximum im Hitabstandsintervall zwischen 170 µm und 180 µm, der für beide Clustertrenner zu sehen ist. Dieser kann für den neuen Algorithmus mit den zu viel durchgeführten Separationen erklärt werden.
So ist an Abbildung 5.6 festgestellt worden, dass die Trennung in mehr Hits als eigentlich vorliegend im Bereich von Hitabständen zwischen 280 µm und 420 µm ihr Maximum erreicht. Bei der Betrachtung der simulierten Daten kann man sehen, dass - für die Trennung in einen Hit zu viel - der zusätzlich gefundene Hit in der Mitte zwischen den beiden vorliegenden Hits lokalisiert wird. Damit haben solche Hits einen Hitabstand zwischen 140 µm und 210 µm und sind damit Bestandteil des eben genannten Peaks, der sich über diesen Hitabstandsbereich erstreckt.

Weiterhin wurde in Abschnitt 5.2 dieses Kapitels angemerkt, dass der funktionale Zusammenhang zwischen dem Hitabstand und den falsch separierten Hits durch den verwendeten Filter gegeben sein könnte. Um diese Hypothese bestätigen oder verwerfen zu können wurde die Analyse aller Datensätze mit SOURCE EXTRACTOR weitere zwei mal durchgeführt und dazu - bei sonst gleicher Konfiguration - zwei andere Filter mit Mexicanhatprofil genutzt. Um hierbei die Fehlseparationen besser untersuchen zu können, wurden Filter mit einer kleineren Halbwertsbreite (*mexhat_1.5_5x5.conv*, *mexhat_2.5_7x7.conv*) verwendet. Für diese Filter wurde bei den Tests im Rahmen von Kapitel 4, Abschnitt 4.2.2 ermittelt, dass sie ein-Hit-Cluster in mehr als einen Hit trennen. Dementsprechend wird ein höheres Maximum der Kurve, die die Mittlere Anzahl von Hits pro Spurlänge beschreibt, für die beiden Filter erwartet.

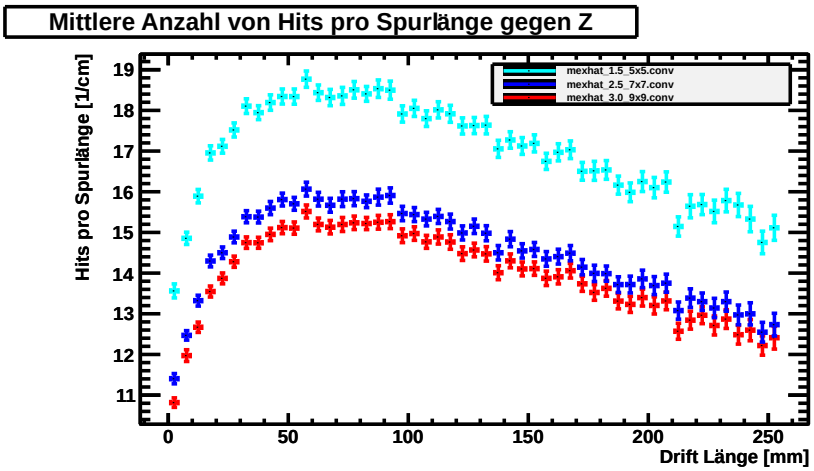


Abbildung 5.8: Mittlere Anzahl von Hits pro Spurlänge für verschiedenen Filtern mit Mexicanhatprofil

Diese Erwartung erfüllt sich. In Abbildung 5.8 ist die Größe „Mittlere Anzahl von Hits pro Spurlänge“ für die beiden neu getesteten Filter, sowie für den bisher verwendeten Filter zu sehen. Dabei ist gut zu erkennen, das die Anzahl der gefunden Hits für Filter mit kleineren Halbwertsbreiten, beziehungsweise Matrizenformen, stark zunimmt. Gleiches muss im Umkehrschluss auch für die Zahl der zu viel separierten Hits gelten. Um die These des Fehlseparations-Peaks zu stützen, sollte der Peak, der als durch die Fehlseparationen bedingt angenommen wurde, für die beiden „kleineren“ Filter größer sein, als für den Filter der optimierten Konfiguration. Es werden so durch die Trennung von eigentlich schon fertig separierten Hits, sowie die zu weit gehende Trennung von Clustern, eine größere Zahl von Hits mit kleinen Abständen erwartet.
Auch dies ist der Fall. Abbildung 5.9 zeigt die Kurven „Hitabstand zwischen zwei Hits pro Ereignis“ für alle drei Filter und es wird deutlich, dass mit dem Filter mit kleinster Halbwertsbreite

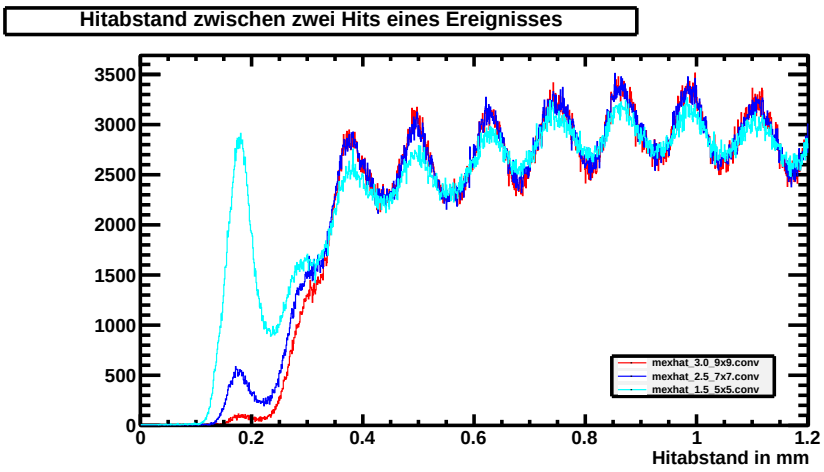


Abbildung 5.9: Hitabstand zwischen zwei Hits pro Ereignis für verschiedenen Filtern mit Mexicanhatprofil

und Matrizenform eine um einiges größere Anzahl an Hits falsch getrennt wird, als mit dem nächst „größeren“ Filter oder dem, bei den Analysen zuvor eingesetzten, Filter. Der weitere Verlauf der Kurven entspricht der Kurve der optimierten Konfiguration: Alle drei Kurven haben nach dem Fehlseparations-Peak (∼180 µm) eine Schulter bei etwa 300 µm, die auf eine erste Oszillation vor dem Erreichen des Plateaus zurück zu führen sein könnte. Nach dieser Schulter wird das Plateau für alle drei Filter für Hitabstände von ca. 370 µm erreicht.

Mit diesen beiden zusätzlichen Analyse Durchläufen konnte somit bestätigt werden, dass der erste Peak der Graphen für die Hitabstände zweier Hits eines Ereignisses auf die Fehlseparationen durch SOURCE EXTRACTOR zurück zu führen ist. Bei der aktuellen Konfiguration sind diese im hohen Maße von der Wahl des Filters abhängig, wobei die Fehlseparationen für den Filter der optimierten Konfiguration nur einen geringen Anteil der Hits ausmachen. Außerdem kann festgestellt werden, dass der mittlere Abstand der falsch separierten Hits zu anderen Hits unabhängig von der Wahl des Filter ist, wohl aber die Häufigkeit, mit der solche Falschtrennungen vorgenommen werden, vom Filter abhängt.
Da auch für den bisher verwendeten Algorithmus Cluster, bei kleinen Hitabständen, in zu viele Hits getrennt wurden liegt die Vermutung nahe, dass auch für diesen Algorithmus ein Fehlseparations-Peak vorliegt. Die Tatsache das es an der entsprechenden Stelle einen Peak gibt (Abb. 5.7), der kleiner als der entsprechende Peak der SEXTRACTOR-basierten Trennung ist, stützt diese Vermutung und deckt sich damit, dass für diesen Clustertrenner eine Anzahl falsch separierter Hits festgestellt wurde, die kleiner als die entsprechende Zahl des neuen Algorithmus ist. Abschließend ließe sich dies bei der Untersuchung größere Datensätze simulierter Daten mit einer bekannten Zahl von Hits pro Ereignis klären.

Kapitel 6

Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde das Programm Source Extractor erstmals als Clustertrenner für mit einer TPC aufgezeichnete Teilchenspuren getestet. Die Eingabeparameter des Programms wurden dahingehend angepasst, dass Source Extractor zum einen einzelne Hits nicht weiter separiert und zum anderen Gebiete mit vielen sich überschneidenden Hits gut auflöst.
Mit diesen Eingabeparamtern wurden Source Extractor als Clustertrenner einer MarlinTPC Analysekette sowohl auf einem großen Datensatz gemessener Daten, als auch auf simulierten Daten mit einer bekannten Anzahl von Hits getestet. Dabei konnte gezeigt werden, dass der auf Source Extractor-basierende Trennalgorithmus Cluster besser in Hits auftrennt als der bisher verwendete Algorithmus. Außerdem ließen sich mit Source Extractor überschneidende Hits bereits bei kleineren Abständen der Hit-Zentren trennen als mit dem alten Clustertrenner.
Anderseits besteht im auf Source Extractor aufbauenden Algorithmus noch der Fehler, dass Pixel falschen Hits zugeordnet werden. Darüber hinaus wurde bei der Verwendung von Source Extractor zur Analyse von simulierten Daten deutlich, dass für bestimmte Abstände der Hitzentren aus mehreren Hits bestehende Cluster in zu viele Hits getrennt werden. Es wurde gezeigt das die Häufigkeit dieses Effekts mit der Halbwertsbreite sowie der Matrizengröße des verwendeten Filters skaliert.
Trotz dieser Fehler lassen sich mit der vorgestellten Konfiguration von Source Extractor bereits mehr Spurpunkte rekonstruieren, als mit dem alten Clustertrenner. Durch Beheben der eben genannten Fehler, sollte sich die Qualität der bei der Analyse erzielten Resultate weiter steigern lassen.

Im Hinblick auf die weitere Nutzung von Source Extractor als Clusterseparator besteht ebenfalls die Möglichkeit, durch Optimieren der Werte der Eingabeparameter, dessen Separationskraft zu erhöhen. Außerdem bietet das Programm noch weitere Möglichkeiten im Hinblick auf die Analyse, die aus Zeitgründen in dieser Arbeit nicht erprobt werden konnten. Source Extractor ist zum Beispiel in der Lage photometrische Analysen von astronomischen Aufnahmen durchzuführen, und kann zum Beispiel die Intensität eines Hits zu berechnen. Durch das Finden von Analogien zwischen charakteristischen Größen von Hits und astronomischen Objekten, wie zum Beispiel „Helligkeit - Ladung“, sollte es mit Source Extractor möglich sein, durch geschickte Wahl der Parameter die Ladung eines Hits auszugeben.

Abschließend kann gesagt werden, dass Source Extractor das Potential hat, neben seiner Anwendung in der Astronomie, zu einem nützlichen Instrument bei der Spuranalyse in der Hochenergiephysik zu werden.

Anhang A

Anhang

A.1 Konfigurationsdatei

```
# (*) indicates parameters which can be omitted from this config file.

#-------------------------------- Catalog -----------------------------------

# name of the output catalog
CATALOG_NAME     TPCDataCat_FY_DTC_DMA9_c.txt
CATALOG_TYPE     ASCII_HEAD   # "NONE","ASCII_HEAD","ASCII","FITS_1.0"
                              # or "FITS_LDAC"

PARAMETERS_NAME lcio_2.param  # name of the file containing catalog contents

#-------------------------------- Extraction ----------------------------------

DETECT_TYPE     PHOTO         # "CCD" or "PHOTO" (*)
FLAG_IMAGE      flag.fits     # filename for an input FLAG-image
DETECT_MINAREA  9             # minimum number of pixels above threshold
DETECT_THRESH   0.0001        # <sigmas> or <threshold>,<ZP> in mag.arcsec-2
ANALYSIS_THRESH 0.0001        # <sigmas> or <threshold>,<ZP> in mag.arcsec-2

THRESH_TYPE     ABSOLUTE

FILTER          Y             # apply filter for detection ("Y" or "N")?
FILTER_NAME     mexhat_3.0_9x9.conv # name of the file containing the filter
FILTER_THRESH   0.0001

DEBLEND_NTHRESH 64            # Number of deblending sub-thresholds
DEBLEND_MINCONT 0.00000001    # Minimum contrast parameter for deblendinge

CLEAN           N             # Clean spurious detections? (Y or N)?
CLEAN_PARAM     1.0           # Cleaning efficiency

MASK_TYPE       CORRECT       # type of detection MASKing: can be one of
                              # "NONE", "BLANK" or "CORRECT"

#------------------------------- Photometry ----------------------------------
```

```
PHOT_APERTURES  5              # MAG_APER aperture diameter(s) in pixels
PHOT_AUTOPARAMS 2.5, 3.5       # MAG_AUTO parameters: <Kron_fact>,<min_radius>

SATUR_LEVEL 50000.0            # level (in ADUs) at which arises saturation

MAG_ZEROPOINT   0.0            # magnitude zero-point
MAG_GAMMA       4.0            # gamma of emulsion (for photographic scans)
GAIN            0.0            # detector gain in e-/ADU.
PIXEL_SCALE     1.0            # size of pixel in arcsec (0=use FITS WCS info).

#------------------------- Star/Galaxy Separation ----------------------------

SEEING_FWHM     1.2            # stellar FWHM in arcsec
STARNNW_NAME   default.nnw    # Neural-Network_Weight table filename

#------------------------------ Background -----------------------------------

BACK_SIZE       1              # Background mesh: <size> or <width>,<height>
BACK_FILTERSIZE 10             # Background filter: <size> or <width>,<height>
BACK_TYPE       MANUAL
BACK_VALUE      0.0

BACKPHOTO_TYPE  GLOBAL         # can be "GLOBAL" or "LOCAL" (*)
BACKPHOTO_THICK 24             # thickness of the background LOCAL annulus (*)

#------------------------------ Check Image ----------------------------------

CHECKIMAGE_TYPE NONE           # can be one of "NONE", "BACKGROUND",
                               # "MINIBACKGROUND", "-BACKGROUND", "OBJECTS",
                               # "-OBJECTS", "SEGMENTATION", "APERTURES",
                               # or "FILTERED" (*)
CHECKIMAGE_NAME chek_seg.fits # Filename for the check-image (*)

#--------------------- Memory (change with caution!) -------------------------

MEMORY_OBJSTACK 2000           # number of objects in stack
MEMORY_PIXSTACK 100000         # number of pixels in stack
MEMORY_BUFSIZE  1024           # number of lines in buffer

#------------------------------ Miscellaneous --------------------------------

VERBOSE_TYPE QUIET            # can be "QUIET", "NORMAL" or "FULL" (*)
```

Literaturverzeichnis

[1] *S. Zimmermann - Diplomarbeit: „Data Reconstruction and Analysis of GEM-Based Time Projection Chambers with Pixel Readout“, 2008*

[2] *W. Blum, L. Rolandi - „Particle detection with drift chambers“, Springer, 1993*

[3] *GAS DETECTORS DEVELOPMENT GROUP* `http://gdd.web.cern.ch/GDD/`

[4] *GEM/PixelGEM Tracking Detectors*
`http://www.e18.ph.tum.de/research/compass/gempixelgem-tracking-detectors/`

[5] *C. Brezina et.al - „Operation of a GEM-TPC with pixel readout“ in Nuclear Science Symposium Confrence Record, 2011; NSS '11IEEE, Oct. 2011 N19-04, pp1006-1013, 2011*

[6] *S. Biagi, CERN, Magboltz -* `http://www.consult.cern.ch/writeup/magboltz/`

[7] *F. Gaede, T. Behnke, N. Graf, T. Johnson - „LCIO - A persistency framework for linear collider simulation studies“, arXiv:physics/0306114v1 [physics.data-an], June 2003*

[8] *Internetseite des ILCSoft Projektes -* `http://ilcsoft.desy.de/portal/`

[9] *J. Abernathy, K. Dehmelt, R. Diener, J. Engels, J. Hunt, M. Killenberg, T. Krautscheid, A. Munnich, S. Zimmermann, M. Ummenhofer, A. Vogel, P. Wienemann - „MarlinTPC: A common software framework for TPC development“, Nuclear Science Symposium Conference Record, 2008. NSS '08. IEEE, 2008*

[10] *C. Brezina - Doktorarbeit in Vorbereitung: „Operation of a GEM based TPC with pixelized read-out“ (Arbeitstitel), Geplante Veröffentlichung: 2012*

[11] *Dr. Benne W. Holwerda - „Source Extractor for Dummies“*

[12] *E. Bertin - Draft: „SExtracor v2.13 User's Manual“*

[13] *Internetseite des CFITSIO-Projekts, NASA's HEASARC: Software*
`http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html`

[14] *Quelle des* Source Extractor *Installationspakete, entsprechender Installationshinweise und Anleitungen zur Problemlösung -*
`http://code.google.com/p/chimera-summer-2010/wiki/SExtractor_Problems`

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate kenntlich gemacht habe.

Bonn, den 13.2.2011
Ort, Datum

Unterschrift