Untersuchungen zum In-plane Alignment der ATLAS-Myonkammern

Inaugural-Dissertation

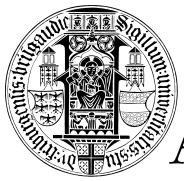
zur

Erlangung des Doktorgrades

der

Fakultät für Mathematik und Physik

der



ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

vorgelegt von

Michael Maaßen

aus

Mönchengladbach

16. Januar 2004

Dekan	Prof. Dr. R. Schneider
Leiter der Arbeit	Prof. Dr. G. Herten
Referent	Prof. Dr. G. Herten
Korreferent	Prof. Dr. K. Königsmann

Tag der Verkündung des Prüfungsergebnisses: 4. März 2004

Inhaltsverzeichnis

Einleitung

Zur Einleitung möchte ich dem Leser helfen, das ATLAS-Experiment, um das es hier geht, einzuordnen. Experimente liefern die Antworten auf die Fragen, welche die Menschen der Natur stellen.

Schon in der Antike haben sich die Völker Gedanken um den Aufbau der Materie und den Zusammenhalt der Welt gemacht. So hat bereits Demokrit[1] vermutet, dass alle Materie aus kleinen unteilbaren Bestandteilen aufgebaut ist. Er nannte sie „ἄτομος“ [2]. Doch diese Annahme blieb lange unbeweisbar. Einen entscheidenden Hinweis lieferte John Dalton (1766-1844) mit seinem „Gesetz der multiplen Proportionen“. Es entwickelte sich von nun an ein Bild der Atome. Durch weitere Arbeiten u.a. von Ernest Rutherford (1871 - 1937) wurde jedoch bald klar, dass die Atome eine innere Struktur haben und somit doch teilbar sind.

Die Beschreibung der Struktur ist leider mit anschaulichen Methoden nicht mehr möglich. Auf diesen kleinen Längenskalen – ein Atom hat etwa einen Durchmesser von $1\,\text{Å} = 10^{-10}\,\text{m}$ – versagen die intuitiven Gesetze der klassischen Physik.

Die moderne Physik beschreibt Teilchen als Quanten verschiedener Felder. Durch diesen mathematischen Apparat wird es möglich, dem „Welle-Teilchen-Dualismus“ Rechnung zu tragen.

Die – nach heutiger Sicht – elementaren Teilchen (Fermionen und Bosonen) werden durch das so genannte „Standard-Modell der Teilchenphysik“ beschrieben. Dieses Modell der Teilchen enthält ihre Eigenschaften und Wechselwirkungen untereinander. Es wurde über mehrere Jahrzehnte entwickelt und steht bis heute in hervorragender Übereinstimmung mit den Messungen. Das Standard-Modell sagt das „Higgs-Boson“ voraus, ein bislang noch unentdecktes Teilchen. Viele aktuelle Projekte, so auch das ATLAS-Experiment, widmen sich unter anderem der Suche nach diesem Teilchen.

Das folgende Kapitel wird das ATLAS-Experiment genauer beleuchten. Dabei wird das ATLAS-Myonsystem beschrieben und es werden Sensoren vorgestellt, welche die mechanischen Bewegungen des Myonsystems messen.

[1] Demokritos von Abdera, griechischer Philosoph (460 - 371 v. u. Z.)
[2] griechisch: nicht teilbar; daher der heutige Begriff “Atome”

Nur mit Hilfe dieser Sensoren kann die hohe Impulsauflösung des ATLAS-Myonsystems erreicht werden.

Im Weiteren wird diese Arbeit einen Teil dieser Sensoren genauer erklären und beschreiben, wie diese Sensoren unter Linux betrieben werden können.

Anhand der Myonkammern, die in Freiburg gebaut werden, werden die Bewegungssensoren untersucht und ihre Messunsicherheiten bestimmt.

Anschließend werden die Daten der Bewegungssensoren interpretiert und aus ihnen Parameter berechnet, welche die Verformung der Kammern beschreiben. Am Beispiel der Freiburger Kammern wird diese Parametrisierung überprüft. Sie schließt die Besonderheiten der Freiburger Kammern ein. Die Messungen zeigen auch, mit welchen Unsicherheiten die Parameter bestimmt werden können.

Kapitel 1

Der ATLAS-Detektor

ATLAS[1] ist einer der Detektoren, die am „Large Hadron Collider“ (*LHC*) nach neuen Teilchen suchen werden. Der Teilchenbeschleuniger *LHC* wird vom europäischen Hochenergie-Physik-Zentrum (*CERN*) bei Genf betrieben. Er soll 2007 seinen Betrieb aufnehmen und befindet sich derzeit im Aufbau.

Bei der Planung und Entwicklung des Beschleunigers und der Experimente stand die Entdeckungsmöglichkeit neuer Teilchen wie des „Higgs-Bosons“ oder Super-Symmetrischer-Teilchen im Vordergrund.

1.1 Der Beschleuniger *LHC*

Der *LHC* ist ein Kreisbeschleuniger und wird mit Protonen (Wasserstoff-Kernen) befüllt. Er wird in einem 27 km langen unterirdischen Tunnel installiert, in dem zuvor der Beschleuniger *LEP*[2] betrieben wurde (siehe Abbildung 1.1). Im *LHC* werden Protonen auf Protonen geschossen und dabei eine Schwerpunktsenergie von 14 TeV erreicht. Im Gegensatz zu bisherigen Beschleunigern werden nicht Protonen mit Antiprotonen, sondern Protonen mit Protonen zur Kollision gebracht. Es ist daher erforderlich, sie in verschiedenen Röhren mit entgegengesetzten Feldern zu beschleunigen. Der *LHC* hat einen Doppelring, dessen Teilchenstrahlen sich in den Experimenten kreuzen und so zur Kollision gebracht werden.

Protonen sind aus Quarks und Gluonen zusammengesetzte Teilchen und bei einer Kollision nimmt meist nur je eins dieser inneren Teilchen an der Wechselwirkung teil. Der größte Teil der Masse (und damit der Energie) der Protonen nimmt gar nicht an der Reaktion teil. Dadurch sinkt die zur Erzeugung neuer Teilchen bereitstehende Energie. Sie ist zudem bei jedem

[1] ATLAS = **A** **T**oroidal **L**HC **A**pparatu**s**

[2] **L**arge-**E**lektron-**P**ositron-Collider, ein Speicherring mit Schwerpunktsenergien bis ca. 200 GeV.

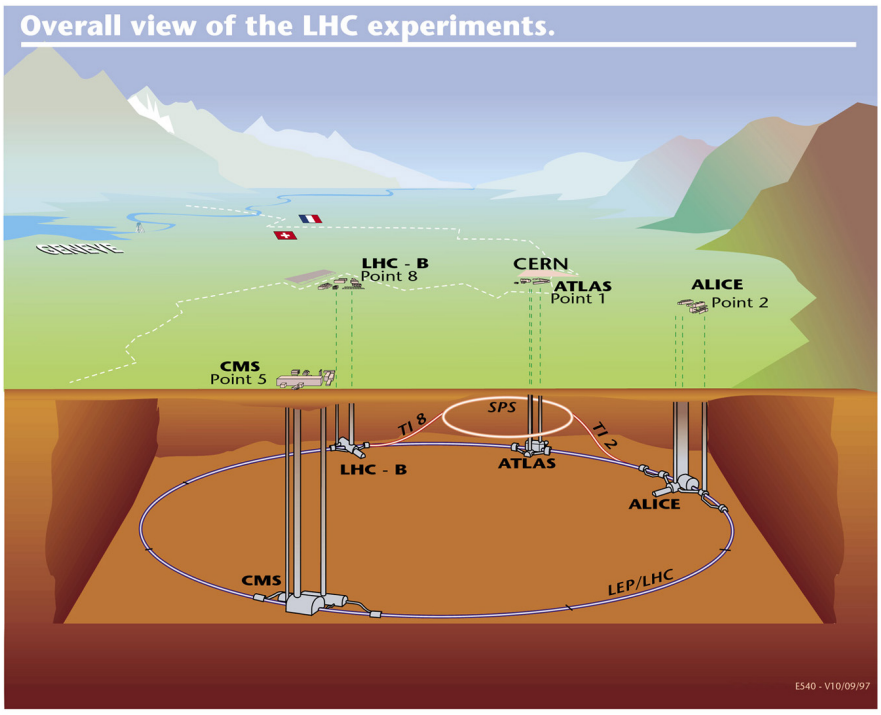


Abbildung 1.1: *Eine Skizze, die die Lage des LHC und seiner Experimente zeigt. Der Speicherring liegt zwischen Jura und Genfer See in ca. 100* m *Tiefe.*

Zusammenstoß unterschiedlich, da die Reaktionspartner jeweils einen anderen Teil des Gesamtimpulses tragen.

Der *LHC* ist so konzipiert, dass Teilchen erzeugt werden können, deren Massen im TeV-Bereich liegen. Das bislang schwerste Elementarteilchen, das „top-Quark“, hat eine Ruheenergie von $180\,\mathsf{GeV} \pm 12\,\mathsf{GeV}$ [K. 03]. Der *LHC* wird in noch unerforschte Energieregionen vorstoßen. Allerdings sind hohe Energien selten und die Erzeugungsrate interessanter Teilchen ist sehr gering; ein Higgs wird etwa alle 10^{10} Reaktionen erwartet (siehe Abbildung 1.2). Damit eine nennenswerte Anzahl dieser Teilchen gemessen werden kann, ist es nötig, sehr, sehr viele Reaktionen zu bekommen, d.h. eine hohe Luminosität zu erreichen. Der Strahl ist beim *LHC* in Pakete – so genannte „Bunches“ – zu je 10^{11} Protonen unterteilt. In den Experimenten werden dann alle $25\,\mathsf{ns}$ zwei Pakete kollidieren, wobei ca. 20 Ereignisse (pro „Bunch crossing“) erwartet werden. Das führt zu einer Luminosität von $10^{34}\,\mathsf{cm}^{-2}\,\mathsf{s}^{-1}$. In der Anfangsphase des Beschleunigers wird diese hohe Luminosität noch nicht erreicht, entsprechend werden nur ein bis fünf Ereignisse pro „Bunch crossing“ erwartet.

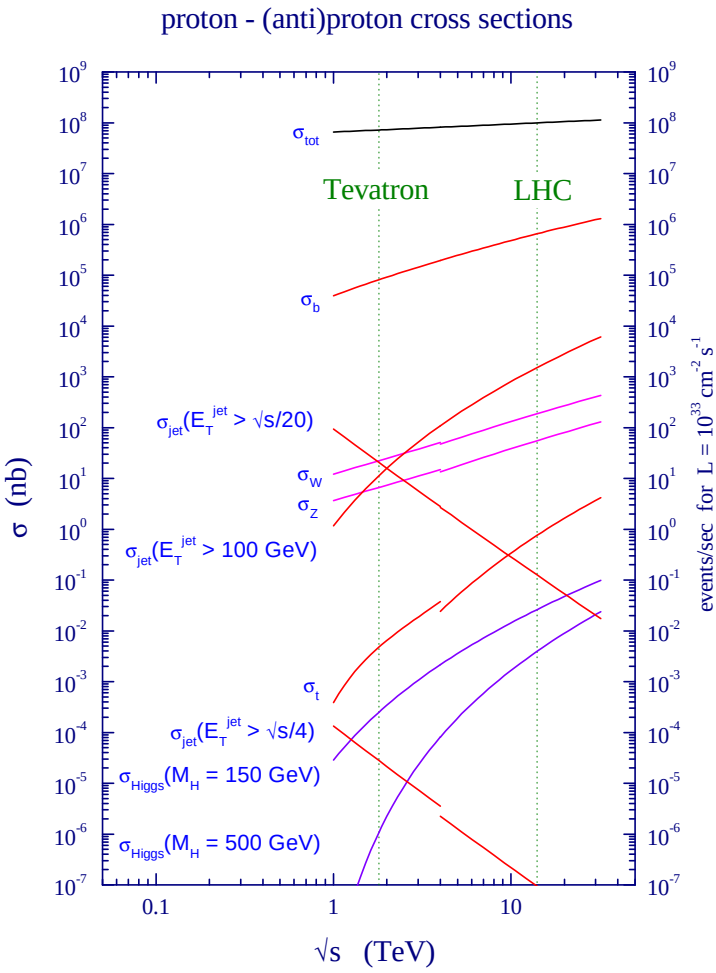


Abbildung 1.2: *Einige partielle Wirkungsquerschnitte bei pp-Kollisionen in Abhängigkeit der Schwerpunktsenergie. So liegt, z. B. beim LHC, der Wirkungsquerschnitt für die Higgs-Produktion etwa 10 Größenordnungen unter dem totalen Wirkungsquerschnitt.*

1.2 Physik bei *LHC*

Der *LHC* ist in der Lage, eine ganze Reihe von Fragestellungen zu beantworten. Die prominenteste ist sicherlich die Suche nach dem Higgs-Boson, welches das Standardmodell komplettieren würde. Aber es gibt auch andere Modelle, die Beschreiben wie die Physik bei hohen Energien „aussehen“ könnte. Beispielsweise die „Supersymmetrie“, die jedem Fermion ein Boson, und jedem Boson ein Fermion zur Seite stellt. Nach diesen Modellen sollte es noch eine Menge weiterer Teilchen zu entdecken geben, von denen einige Massen von unter 1 TeV haben sollten.

Neben der Suche nach „neuer Physik“ sollen am *LHC* auch Präzisionsmessungen an bekannten Teilchen durchgeführt werden. Das top-Quark ist ein Kandidat solcher Messungen, aber auch CP-Verletzungen an B-Mesonen sollen untersucht werden.

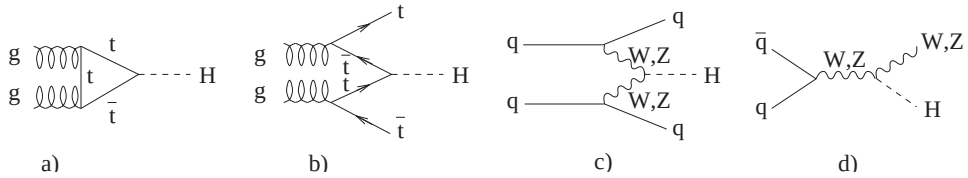


Abbildung 1.3: Die wichtigsten Feynmandiagramme der Higgs-Produktion aus der pp-Streuung bei *LHC*. a) Gluon-Fusion b) $t\bar{t}$-Annihilation c) WW- und ZZ-Fusion d) W- und Z-Bremsstrahlung

1.2.1 Higgs-Boson

Befassen wir uns nun ein wenig mit dem Higgs-Boson (**H**). In Abbildung 1.3 sind die wichtigsten Prozesse zu sehen, die ein Higgs bei pp-Kollisionen erzeugen können. Das Higgs-Feld wurde in das Standardmodell aufgenommen, um die Massen der Teilchen – besonders der schweren Eichbosonen ($\mathbf{W}^{\pm}$, $\mathbf{Z}^0$) – zu erklären [Hig64]. Alle Teilchen mit Masse treten mit diesem Hintergrundfeld in Wechselwirkung und erhalten so ihr „Gewicht“. Aus diesem skalaren Feld (kein Spin) erhält man durch „Quantisierung“ das Higgs-Boson als Teilchen. Im Standardmodell sind alle Eigenschaften dieses Teilchens, bis auf seine Masse, schon festgelegt.

Die möglichen Zerfallskanäle des **H** hängen von seiner Masse ab. Um den erwarteten Massebereich möglichst vollständig abzudecken, wurde der ATLAS-Detektor so ausgelegt, dass er auf verschiedene Zerfälle, die einen Massebereich des Higgs von 80 GeV bis 1 TeV abdecken, sehr sensitiv ist [ATL94]:

- $\mathbf{H} \to b\bar{b}$ $(80 < m_H < 100\,\mathsf{GeV})$
- $\mathbf{H} \to \gamma\gamma$ $(90 < m_H < 150\,\mathsf{GeV})$
- $\mathbf{H} \to \mathbf{ZZ}^* \to 4l^{\pm}$ $(130\,\mathsf{GeV} < m_H < 2m_Z)$
- $\mathbf{H} \to \mathbf{ZZ} \to 4l^{\pm},\ 2l^{\pm}\,2\nu$ $(m_H > 2m_Z)$
- $\mathbf{H} \to \mathbf{WW}, \mathbf{ZZ} \to l^{\pm}\nu\ 2\,\text{jets},\ 2l^{\pm}\ 2\,\text{jets}$ $(m_H > 2m_Z)$

Die möglichen Endzustände sind sehr zahlreich, es gibt also eine Menge zu messen.

1.2.2 Weitere neue Teilchen

Sollte das Standardmodell korrekt (vollständig) sein, könnte es eine weite Energieskala ohne weitere Teilchen geben.

Interessanter ist jedoch der Fall, in dem eine ganze Reihe neuer Teilchen gefunden wird. So sagt z. B. die einfachste supersymmetrische Erweiterung

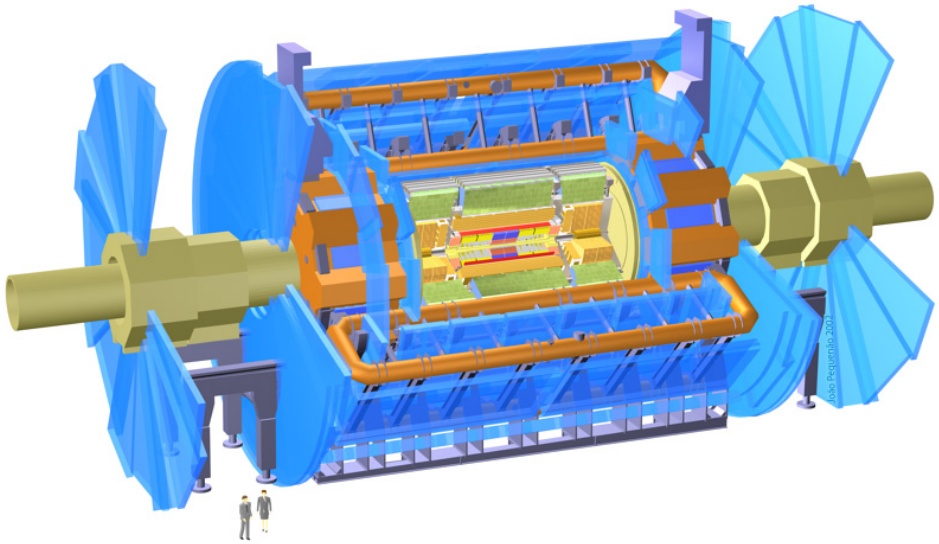


Abbildung 1.4: *Dieses Bild zeigt schematisch den* ATLAS*-Detektor. Die blaugefärbten Bereiche gehören zum Myonsystem, es nimmt das größte Volumen ein. Der Detektor ist 46* m *lang, hat einen Durchmesser von 22* m *und wiegt etwa 7000* t *[ATL99a]. Man sieht im unteren Bereich die Hohlfüße, in die unsere Kammern eingebaut werden. Die Kalorimeter sind grün und gelb gefärbt, in ihnen befinden sich der Solenoid-Magnet und der „inner Detector“.*

des Standardmodells (MSSM, „Minimal Supersymmetric Standard Model“) gleich fünf Higgs-Bosonen ($\mathbf{H}^{\pm}$,$\mathbf{h}$,$\mathbf{H}$,$\mathbf{A}$) voraus [ATL99b]. Aber auch einige „supersymmetrische Partner“ schon bekannter Teilchen könnten entdeckt werden. Es gibt große Hoffnung, zumindest einige von ihnen zu sehen. Supersymmetrische Teilchen spielen auch in der Kosmologie eine Rolle, da sie als Kandidaten für die „dunkle Materie“ gehandelt werden ([Cli03]).

1.3 Das ATLAS-Experiment

Es gibt also viele theoretische Varianten, wie die Natur bei 1 TeV aussehen könnte. Der ATLAS-Detektor (siehe Abbildung 1.4) wurde entworfen, um dieses weite Spektrum zugänglich zu machen [LoI92]. Es sollte ein Detektor entstehen, der möglichst viele der denkbaren Teilchen „sehen“ kann.

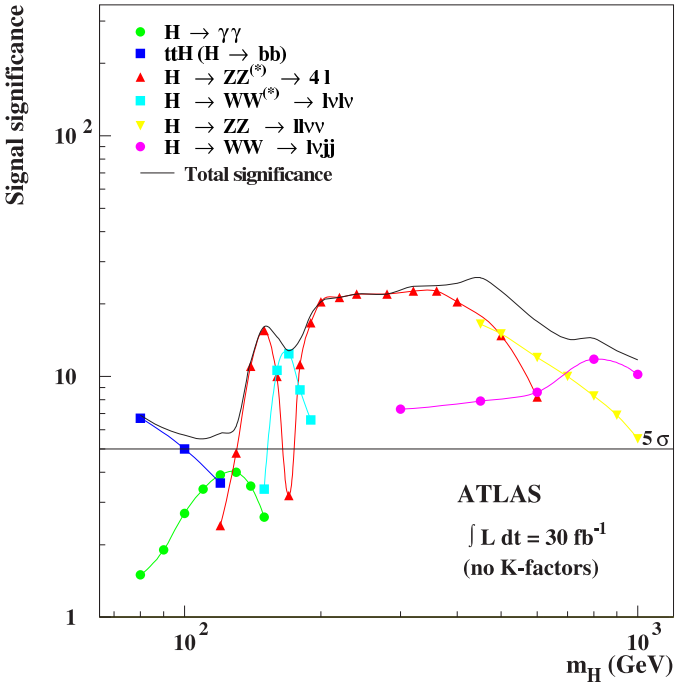


Abbildung 1.5: *Entdeckungssignifikanz des Higgs. Je nach Masse tragen unterschiedliche Zerfälle zur Entdeckung bei. (aus [ATL99b])*

Die hohe Ereignisrate am *LHC* macht einen strahlenharten Detektor erforderlich, der ein großes Datenvolumen verarbeiten kann. Alle 25 ns werden etwa 20 Ereignisse (gleichzeitig) erwartet. Zu einer guten Performanz gehört aber auch, die Signaturen der zu entdeckenden Teilchen effektiv zu erkennen. So hat ATLAS eine gute „secondary vertex“[3] Erkennung (für b und τ Detektion), und ein hochauflösendes Kalorimeter, um Jets und fehlende transversale Energie messen zu können.

In Abbildung 1.5 sind die unterschiedlichen Beiträge der Endzustände zur Entdeckung des Higgs-Bosons eingezeichnet. Eine sehr gute Entdeckungsmöglichkeit verspricht man sich von einem Zerfall in vier Leptonen (speziell Myonen), da dieser gut vom Untergrund getrennt werden kann.

Der ATLAS-Detektor ist nach den „Zwiebelschalenmodell“ aufgebaut. Im Inneren (nahe des Strahls) befinden sich Detektoren mit sehr hoher Ortsauflösung. Sie sollen die Bahnen der verschiedenen Teilchen vermessen. Dieser „Inner Detector“ befindet sich im Feld eines Solenoid-Magneten, sodass die Bahnen von geladenen Teilchen gekrümmt sind. Das ermöglicht die Impulsmessung dieser Teilchen. In den nächsten Schalen befinden sich elektromagnetisches und hadronisches Kalorimeter, die die Energien der Teilchen messen. Die äußere Schale ist das Myonspektrometer, das den Impuls der Myonen misst. Myonen werden wegen ihrer geringen Wechselwirkung mit Materie in den Kalorimetern nicht gestoppt.

[3] Zerfall eines (kurzlebigen) Teilchens in geringer Entfernung vom eigentlichen Wechselwirkungspunkt.

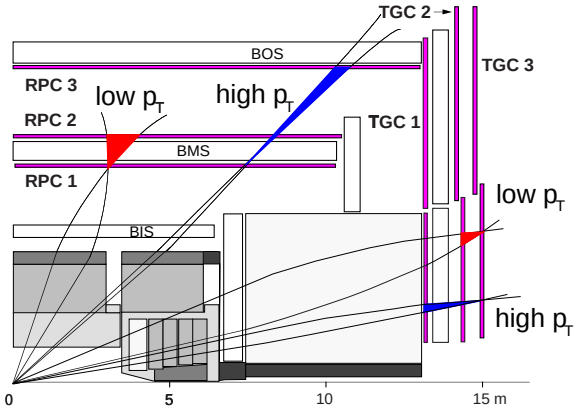


Abbildung 1.6: *Messung des Myonimpulses – hier ist ein Quadrant des* ATLAS*-Detektors gezeigt. Die Myonen mit niedrigem (low p_T) und hohem Impuls (high p_T) werden unterschiedlich stark abgelenkt. Im Myonsystem wird die Bahn an drei Stellen gemessen (hier: BIS, BMS, BOS) und daraus die Ablenkung bestimmt. Die Krümmung ist hier stark übertrieben dargestellt, ein 1* TeV *Myon erfährt bei* ATLAS *eine Ablenkung von ca. 500* μm.

1.4 Das ATLAS-Myonsystem

Auf das Myonspektrometer wurde bei ATLAS besonderen Wert gelegt. Es ist das größte Subsystem (Durchmesser: 22 m; Länge: 46 m) im gesamten Detektor. Das Konzept des Myonsystems ist eine Optimierung zwischen geringen Kosten und den geforderten Fähigkeiten [ATL99a] :

- hohe Nachweiseffizienz
- hohe Impulsauflösung $\frac{\Delta p}{p}$
- wenig Vielfachstreuung, bes. bei geringen Impulsen
- Erkennung von „fake tracks“

Für die Impulsmessung der Myonen verwendet ATLAS einen Toroid-Magneten, d.h. die Feldlinien verlaufen auf konzentrischen Kreisen um die Strahlachse. Der Impuls wird bestimmt, indem die Krümmung – genauer die Sagitta – der Bahn im Magnetfeld gemessen wird (siehe Abbildung 1.6). Ein Myon, das den Detektor verlässt, muss drei Lagen von Myonkammern passieren. Die Kammern messen dabei den Durchstoßpunkt der Myonbahn (durch die Kammerebene). Die Koordinate parallel zur Strahlachse wird mit weit höherer Präzision gemessen, da nur sie zur Bestimmung des Impulses dient. Die Kammern sind so angeordnet, dass sich drei ineinander liegende Zylinderschalen ergeben, die auf jeder Seite mit einem „Deckel“ verschlossen werden. Damit die Vielfachstreuung im Myonsystem möglichst gering bleibt, wird das Magnetfeld durch supraleitende Luftspulen erzeugt.

Der eigentliche Nachweis der Myonen wird durch Ionisation eines Gases erreicht. In der Mitte einer Aluminiumröhre ($\phi = 3$ cm) wird ein Draht ($\phi = 50\,\mu$m) gespannt und auf ein Potential von ca. 3 kV gebracht. Die Ionen

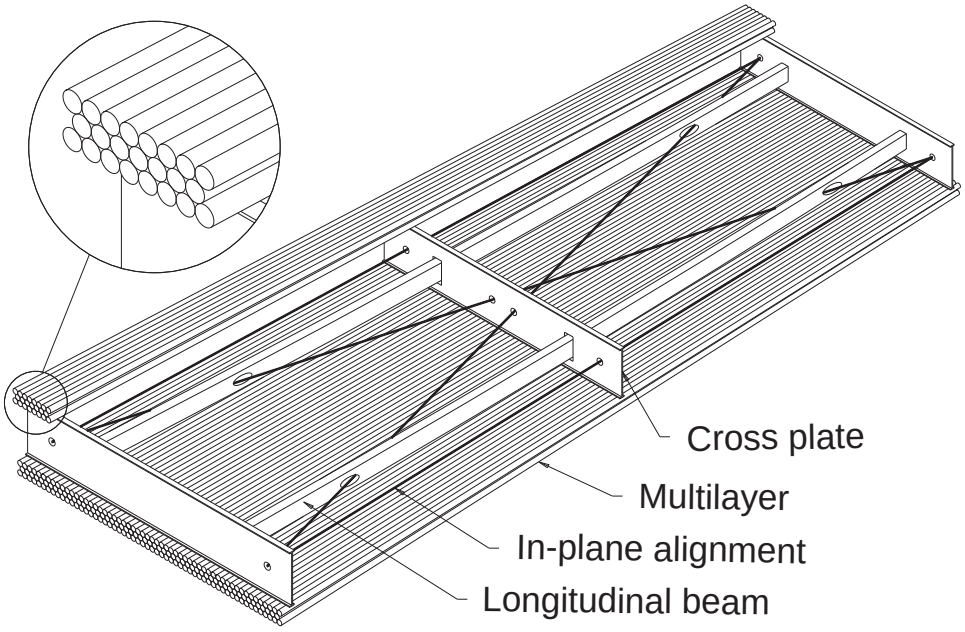


Abbildung 1.7: *Aufbau einer Myonkammer, wie sie im „Fass-Bereich“ von* ATLAS *eingesetzt wird. Man erkennt die aus einzelnen Röhren aufgebauten Lagen und das System zur Bewegungsüberwachung („In-plane Alignment“).*

und Elektronen driften durch das Gas. In der Nähe des Drahtes, wo das Feld am stärksten ist, werden die Elektronen so weit beschleunigt, dass es zu weiterer Ionisation kommt (Gasverstärkung). Durch diesen Effekt lässt sich am Draht ein Spannungspuls messen. Aus der Driftzeit, genauer der Zeit vom Durchgang des Myons bis zum Eintreffen des Pulses, berechnet man den minimalen Abstand der Myonspur vom Draht. Der Beginn der Driftzeit wird durch den Zeitpunkt des Bunch Crossings definiert. Die Zuordnung zu den einzelnen Bunches wird das mit Hilfe spezieller Trigger-Kammern („RPC“) realisiert. Bei ATLAS werden viele dieser Röhren zu einer Kammer verklebt, siehe Abbildung 1.7. Eine Kammer besteht aus einem Rahmen („Spacer“) der auf beiden Seiten Röhren trägt. Die Rahmen sind aus drei Querträgern und zwei Längsträgern zusammengesetzt, wobei die Röhren nur mit den ersteren verbunden sind. Man hat so pro Kammer sechs oder acht Lagen und man kann die Myonspur rekonstruieren. Um die Form der Kammern während des Betriebs zu überwachen, sind in den Rahmen verschiedene Sensoren eingebaut. Dieses Sensorsystem wird „In-plane Alignment“ genannt.

ATLAS soll eine Impulsauflösung von $\frac{\Delta p_T}{p_T} = 1 \cdot 10^{-4}\, p \,/\, \mathsf{GeV}$ erreichen (siehe [ATL97]). Dabei ist p der Impuls des Myons und p_T die Transversalkomponente (senkrecht zur Strahlachse) des Impulses. Dies kann allerdings nur für $p_T > 300\,\mathsf{GeV}$ erreicht werden, da unterhalb dieser Grenze die Vielfachstreuung einen zu großen Einfluss hat. Für ein $1\,\mathsf{TeV}$ My-

on, das senkrecht aus dem Detektor fliegt, soll der transversale Impuls auf $\frac{\Delta p_T}{p_T} = 1 \cdot 10^{-4} \, \frac{1\,\mathsf{TeV}}{\mathsf{GeV}} = 10\%$, also auf $100\,\mathsf{GeV}$ genau bestimmt werden können.

Um diese Auflösung zu gewährleisten, muss jede Kammer die Axialkomponente der Myonbahn auf $50\,\mu\mathsf{m}$ genau bestimmen. Die drei Ebenen von Kammern liefern somit drei Punkte, aus denen dann die Bahnkrümmung berechnet wird. Die große Herausforderung ist, diese extrem hohe Präzision bei einem so riesigen Detektor zu bewerkstelligen.

Es ist nicht möglich, alle Kammern auf $50\,\mu\mathsf{m}$ genau aufzuhängen und zu betreiben, zum einen wirken große Kräfte (Gewicht und magnetisches Feld), zum anderen wird es starke Temperaturunterschiede geben (wärmeerzeugende Elektronik neben Kryo-Magneten). Weiter möchte man so wenig Material wie möglich in das Myonspektrometer einbringen, damit die Vielfachstreuung gering gehalten wird. Wie man sieht, ist es unmöglich, eine Bewegungen der Kammern zu verhindern – das ist auch gar nicht nötig, man muss sie nur *kennen*. Aus diesem Grund hat man sich ein Überwachungssystem überlegt, welches die Positionen der Drähte mit hinreichender Genauigkeit bestimmen kann. Dieses System wird in der ATLAS-Literatur „Alignment“ genannt, dieser Begriff ist etwas irreführend, da *Alignment* eigentlich eine aktive Ausrichtung impliziert.

Das ATLAS Alignment-System ist ein rein passives Netz von Sensoren, das auf verschiedenen Stufen Bewegungen registriert. Die Ziele dieses Systems sind:

- Global die Kammerposition auf $400\,\mu\mathsf{m}$ zu messen.
- Die Sagitta auf $30\,\mu\mathsf{m}$ zu korrigieren.

Das System gliedert sich in mehrere Unterkomponenten (vgl. [Bar01]):

Referenz System Dieses Sensornetz umspannt den gesamten Detektor. Es verbindet die einzelnen Sensorplattformen untereinander und mit den Kammern.

Projektives Alignment Diese Sensorstrecken „zielen“ auf den Wechselwirkungspunkt und unterstützen die Sagittakorrektur, da sie „parallel“ zu den Spuren verlaufen.

Praxiales System Verbindet die Kammern einer Ebene miteinander. Ein Teil misst die Position zur Nachbarkammer sehr genau, der andere Teil bestimmt über mehrere Kammern die Ebenheit.

BIL-BIR Verbindung Spezielles System, das eine Lücke im Inneren des Detektors überbrückt.

In-plane Alignment Das In-plane Alignment ist in jeder Kammer enthalten und dient dazu, Bewegungen einer Kammer in sich zu messen.

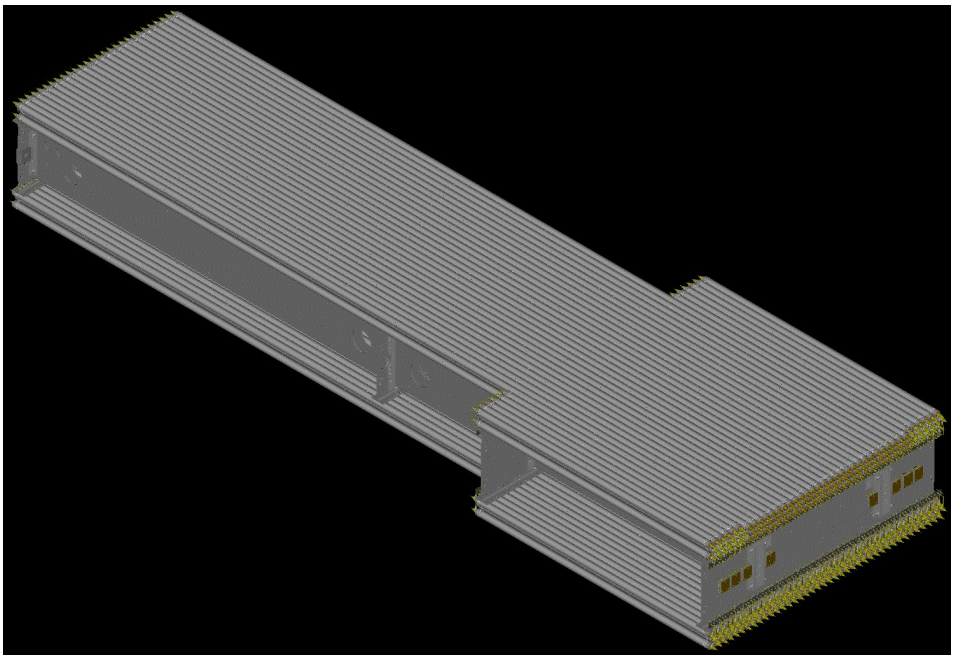


Abbildung 1.8: *Ein 3D-Modell der Freiburger Myonkammern; es wurde direkt aus den CAD-Vorlagen generiert.*

Bei einer „normalen“ Kammer besteht das In-plane Alignment aus zwei (zu den Röhren) parallelen und zwei gekreuzten Strahlen (siehe Abbildung 1.7). Bei den Myonkammern, die in Freiburg gebaut werden (BOG), fällt dieses In-plane Alignment komplizierter aus. In der vorliegenden Arbeit soll es genau um dieses System gehen, daher werde ich es genauer vorstellen.

1.5 Die Freiburger Myonkammern

Die Freiburger Kammern sind Teil der äußeren Zylinderschale und sollen in die Füße des ATLAS-Detektors eingebaut werden. Dabei ist es das Ziel, die nicht abgedeckte Fläche möglichst gering zu halten.

Der beengte Raum in den Füßen erfordert sehr schmale Kammern. Sie sollen zudem genauso lang sein (ca. 4 m) wie die Nachbarkammern (*BOF*). Der Detektorfuß ist allerdings kürzer, sodass die Kammern herausstehen. Diese herausstehenden Teile können nun breiter sein (1,2 m) als die Fußbereiche (ca. 85 cm). Diese Konstruktion erhöht die sensitive Fläche pro Kammer um 12,5 %. In Abbildung 1.8 ist ein Bild der Kammer gezeigt, man erkennt gut die „T-förmige“ Struktur. Diese Form macht einen weiteren Querträger (Cross Plate) im Kammerrahmen erforderlich. Auf ihm enden die 1,2 m langen Rohre, während die 4 m langen keinen Kontakt zu ihm haben. Für die kurzen Rohre ist er die Hochspannungsseite (High Voltage), d.h. an ihm werden die Hochspannungs- und ein Teil der Gasversorgung montiert.

Im Gegensatz zu den „normalen“ Kammern mit drei Querträgern. haben unsere Kammern vier Querträger. Dieser Umstand erfordert ein aufwendigeres Sensoren-Netz im Innern der Kammer. Für diesen zusätzlichen Querträger sind in der Kammer vier weitere Bewegungssensoren eingebaut. Wie der mittlere Querträger auch, ist er mit vier Linsen bestückt. Da die Linsen aber nicht mittig zwischen Masken und Kameras sitzen, werden die Bilder vergrößert abgebildet (ca. 1 : 2). Mit den vier Strecken des mittleren Querträgers besitzt der Kammertyp BOG also doppelt so viele Sensoren (Rasniks, siehe Kapitel 2), wie eine Standardkammer im Fassbereich des Detektors. Diese zusätzlichen Sensoren sind sehr wichtig, da sie die einzige Möglichkeit sind, Informationen über die Enden der kurzen Rohre zu bekommen. Die BOG-Kammern sind, wie die Standardkammern auch, mit den praxialen Sensoren ausgestattet, aber nicht in doppelter Anzahl.

Dieses Kapitel erklärt, dass die gewünschte Messgenauigkeit der Myonimpulse eine präzise Messung der Bahnkrümmung erfordert. Diese Präzision wird durch Driftröhren erreicht, deren Positionen während des Betriebs überwacht werden. Im letzten Abschnitt werden die BOG-Kammern vorgestellt, die ein komplexeres System von Überwachunssensoren haben. Im weiteren Verlauf der vorliegenden Arbeit wird dieses System genauer beschrieben und untersucht.

Kapitel 2

Rasnik und Linux

Beim Myonspektrometer des ATLAS-Detektors werden so genannte „Monitored-Drift-Tubes“ (MDT) eingesetzt. Dabei bedeutet das „Monitored“, dass Bewegungen (z. B. durch Erwärmung, das magnetische Feld, etc.) der Driftröhren und der Kammern zueinander beobachtet werden. Dieses System von Sensoren wird „Alignment“ genannt. Für Bewegungen einer Kammer in sich (Verformungen) ist das „In-plane Alignment“ zuständig, um das es im Folgenden gehen soll.

Als erstes wird das „Rasnik-System“ beschrieben, das beim In-plane Alignment eingesetzt wird. Neben dem Rasnik-System werden beim In-plane Alignment Temperatursensoren verwendet, über welche die Änderungen der Gesamtlängen bzw. Breiten der Kammern messen werden. Im weiteren Verlauf dieses Kapitels wird gezeigt, dass das Rasnik-System durch meine Software unter Linux gesteuert und ausgelesen werden kann.

2.1 Das Rasnik-System

Das Institut NIKHEF[1] hat ein Positionsüberwachungssystem, genannt „Rasnik“[2], entwickelt [vdGGLR00]. Es misst die relative Lage von drei optischen Elementen zueinander, die (nahezu) auf einer Linie liegen. So kann die Verschiebung (senkrecht zur optischen Achse) eines Elements bezüglich der anderen beiden, mit einer Genauigkeit von typischerweise $\approx 1\,\mu\mathsf{m}$, bestimmt werden. Es können auch Rotationen und Längsverschiebungen gemessen werden, allerdings ist die erreichbare Auflösung dabei deutlich geringer.

[1] Nationales niederländisches Kern- und Hochenergiephysik-Institut, http://www.nikhef.nl

[2] Rasnik = **R**ed **A**lignment **S**ystem from **Nik**hef

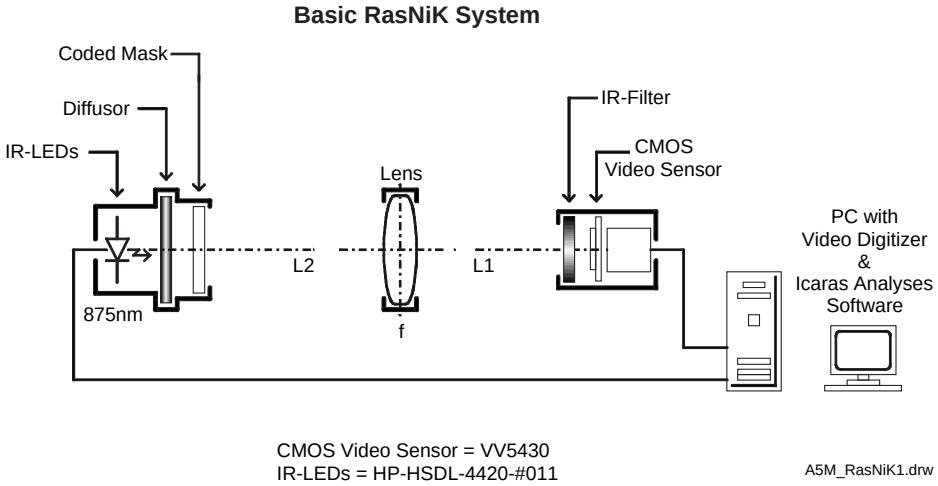


Abbildung 2.1: *Schematische Darstellung eines Rasnik-Systems. Das Licht der LEDs wird durch einen Diffusor „homogenisiert“ und beleuchtet eine Maske. Die Maske wird mit Hilfe der Linse auf den CMOS-Bildaufnehmer abgebildet. Das gewonnene Bild wird von einem Computer ausgewertet (aus [vdGGLR00]).*

2.1.1 Optischer Aufbau

Eine Rasnik-Strecke besteht aus einer Maske, einer Linse und einer „Kamera“ (genau genommen ist es nur ein Bildsensor und keine Kamera). Das Bild der beleuchteten Maske bildet die Linse auf den Bildaufnehmer ab. Die Komponenten werden RasLed [GR02b], RasLens und RasCam [GR99a] genannt (siehe Abbildung 2.1).

Zur Beleuchtung der Maske dient ein 3×3-LED-Feld. Die LEDs senden ein infrarotes Licht mit einer Wellenlänge von $875\,\mathsf{nm}$ aus. Die Verwendung von infrarotem Licht reduziert Störungen durch Umgebungslicht. Es folgt dann ein Diffusor, um eine homogene Beleuchtung für die Maske zu bekommen.

Die Maske besteht aus einer $2 \times 2\,\mathsf{cm}^2$ großen Glasplatte, auf die ein Muster aufgebracht ist. Durch die Größe der Maske wird der dynamische Bereich des Systems festgelegt, d.h. wie weit sich die Komponenten bewegen können, ohne den Messbereich zu verlassen. Die RasCam nimmt nur einen Ausschnitt der Maske auf. Die Lage dieses Ausschnittes lässt sich durch das Muster bestimmen. Das Muster ist aus schwarzen und weißen Quadraten aufgebaut, die kodiert die Position auf der Maske enthalten. Die Interpretation dieses Kodes wird im nächsten Abschnitt beschrieben.

Als Bildsensor wird ein CMOS-Bildaufnehmer (*VV 5430* von VLSI VISION, [VLS98]) verwendet, der hinter einem Infrarot-Filter[3] steckt. Der Fil-

[3] Schott RG830, [Sch97]

ter schirmt den Sensor vor sichtbarem Licht ab, sein Transmissionskoeffizient steigt im Bereich von $800 - 850\,\mathsf{nm}$ stark an. Der CMOS-Sensor ist im Wellenlängenbereich von ca. $430\,\mathsf{nm}$ bis $1000\,\mathsf{nm}$ empfindlich[4] Die aktive Fläche der Kamera hat eine Größe von $4,66 \times 3,54\,\mathsf{mm}$ mit 384×287 Punkten (Pixeln). Ein einzelnes Pixel hat eine Größe von $12 \times 12\,\mu\mathsf{m}$, d.h. die Pixel sind etwa eine Größenordnung kleiner als die Quadrate der Maske (s.u.). Bleibt noch die Linse. Sie wird üblicherweise in der Mitte positioniert, aber kann bei Bedarf entlang der optischen Achse verschoben werden. Die eingesetzten Linsen haben einen Durchmesser von $40\,\mathsf{mm}$ und ihre Brennweite bestimmt sich aus Gegenstands- (g) und Bildweite (b) nach der Abbildungsgleichung:

$$\frac{1}{f} = \frac{1}{g} + \frac{1}{b} \tag{2.1}$$

2.1.2 Bestimmung der Position

Das Muster auf den Masken ist aus mehreren 8×8-Feldern (Schachbrettern) aufgebaut, zwischen denen sich jeweils eine Positionsmarkierung befindet (siehe Abbildung 2.2). Ein „Schachbrett“ mit je einer vertikalen und horizontalen Markierung wird Basisblock (basic building block, B^3 — siehe [Gro94]) genannt. Die Positionsmarkierung besteht aus einer Binärzahl, d.h. acht Bit Positionsinformation und einem Startbit (= Eins). Die Positionsmarkierung zählt von links bzw. unten die Basisblöcke durch, beginnend bei $(0;0)$. Damit die Position eindeutig bestimmt ist, kann eine Maske maximal 255×255 Basisblöcke enthalten. Bei einer üblichen Feldgröße von $120\,\mu\mathsf{m}$ entspricht das einer $275 \times 275\,\mathsf{mm}^2$ großen Maske. Die Masken werden in dieser Größe produziert und anschließend zersägt.

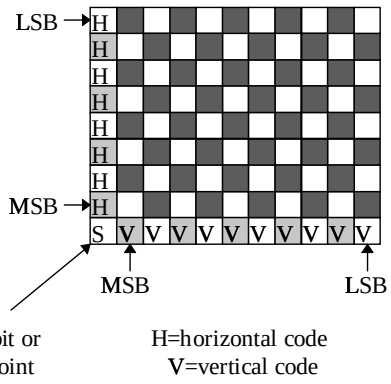


Abbildung 2.2: *Aufbau eines 8×8-Feldes mit zugehöriger Positionsmarkierung.*

Die Kodierung der Binärdaten funktioniert so, dass jede logische Eins das alternierende Muster invertiert — d.h. wenn ein Kode-Feld die gleiche Farbe wie das nebenliegende Schachbrettfeld aufweist, entspricht dies einer log. Eins.

Ist nun die Größe der einzelnen Quadrate bekannt (BOG: $120\,\mu\mathsf{m}$ oder $85\,\mu\mathsf{m}$), so kann die absolute Position des Ausschnittes (Bild) auf der Maske berechnet werden. Genauer gesagt wird die Position des linken oberen Bildpunktes bezüglich der Maske bestimmt. (Es kann auch die Mitte des Bildes als Referenz gewählt werden.) Dazu bestimmt die Analyse zunächst

[4] Effizienz $> 20\%$

die Positionen der Schachbretter ([Gro94]). Es wird dazu ein alternierendes schwarz-weiß Muster dem Bild so angepasst, dass eine möglichst gute Übereinstimmung erreicht wird. Dabei wird die Größe, Orientierung und Lage variiert, die Kodierung wird bei diesem Schritt ignoriert. Nach dieser Anpassung (Fit) ist bereits die Rotation und die Vergrößerung bestimmt. Bei den Rasniks in den BOG-Kammern wird ein Schachbrettfeld auf ca. 10×10 Pixel abgebildet. Dieses Verhältnis von Pixeln zu Feldern, zusammen mit einer Interpolation der Feldkanten durch die Helligkeitswerte, ergibt die Unsicherheit von ca. $1\,\mu\mathsf{m}$. Wenn eine Rasnik-Strecke stark vergrößert oder verkleinert, so ist die Feldgröße der Maske entsprechend anzupassen. (Bei den vergrößernden Rasniks unserer Kammern werden deshalb Masken mit $85\,\mu\mathsf{m}$ Feldern eingesetzt.) Aus den genauen Bildgrößen der Felder kann der Abbildungsmaßstab bestimmt werden.

Der erste Schritt der Analyse liefert zudem die xy-Position modulo der Basisblockgröße. Das heißt, man kennt die Position der linken unteren Ecke (Startbit) des Bildes innerhalb des linken unteren Basisblocks. Als nächstes muss dieser Basisblock bestimmt werden. Dazu werden in dem Bild die Zeilen und Spalten gesucht, die die Kodierung enthalten. In den Spalten ist die horizontale Position kodiert, sodass in einer Spalte immer der gleiche Wert steht. Analog enthält auch jede einzelne Kode*zeile* immer der gleichen Wert. Das erleichtert das Auffinden der Zeilen und Spalten erheblich und erhöht zudem die Erkennungsrate für Bildfehler. Jedes Bild sollte mehrere Kodezeilen und -spalten enthalten, dadurch kann die Monotonie der Koordinaten bestimmt werden. Somit ist es möglich, zu erkennen ob die Maske verdreht oder gespiegelt eingebaut worden ist.

Nach diesen Berechnungen ist nun bekannt, welchen Teil der Maske die Kamera gerade aufnimmt. Die Ergebnisse werden noch auf den gewählten Referenzpunkt umgerechnet. Falls die Mitte des Sensors als Referenz dient, kann die berechnete Position beschrieben werden als der Schnittpunkt der Maske mit der Geraden, die durch die Mittelpunkte von Linse und Kamera geht. Die Position auf der Maske wird zunächst in Einheiten der Basisblockgröße gemessen und dann über die Feldgröße in Millimeter umgerechnet. Eine Änderung der berechneten Position kann durch eine Bewegung von jeder der drei optischen Komponenten hervorgerufen werden (siehe Gleichung (4.2)). Wie stark sich die einzelnen Verschiebungen in den Rekonstruktionswerten niederschlagen, hängt vom Vergrößerungsfaktor der Rasnik-Strecke ab. Eine Bewegung der Linse verschiebt das Bild in die entgegengesetzte Richtung als die Bewegungen der anderen Komponenten.

Der Analyse-Routine kann man mitteilen, ob man die Position der Linse oder der Maske berechnet haben möchte. Im ersten Fall werden die berechneten xy-Werte noch mit dem Faktor $\frac{v}{1+v}$ multipliziert, wobei v der berechneten Vergrößerung entspricht. Im zweiten Fall bleiben die Werte unverändert. Als Ergebnisse liefert die Analyse-Routine die xy-Werte, den Vergrößerungsfaktor sowie die Rotation des Bildes. Aus der Vergrößerung kann

die relative Lage der Linse zwischen Maske und Sensor berechnet werden (vlg. auch Gleichung (4.11)).

Zudem versucht sie die Kippwinkel der Maske bzgl. der x- und y-Achsen zu bestimmen. Allerdings sind die Unsicherheiten bei diesen Werten so groß, dass ich sie nicht verwende und deshalb nicht weiter darauf eingehen möchte.

2.1.3 Elektronische Komponenten

Das gesamte System ist hierarchisch aufgebaut. Die Verbindung zwischen den RasCams bzw. RasLeds und dem steuernden Computer verläuft über mehrere Ebenen von Multiplexern. Dadurch wird die Anzahl der zur Rasnik-Auslese nötigen Computer bei ATLAS auf 4 begrenzt ([vdGGLR00]). Der „RasMux“ ist der Multiplexer auf der letzten Ebene, an ihm sind die RasLeds und RasCams angeschlossen. Da wir jeweils nur eine Kammer auslesen müssen, haben wir den RasMux über ein Interface direkt mit einem Computer verbunden.

Elektrisch bestehen die RasLeds aus neun LEDs und einem Vorwiderstand, die in Serie geschaltet sind. Sie nehmen bei 24 V etwa 65 mA auf. Der RasMux schaltet sie schlicht ein und aus.

Bei den RasCams gibt es für den RasMux mehr zu tun. Zunächst wird auch hier die Versorgungsspannung (12 V und 100 mA) eingeschaltet. Die RasCams beginnen dann sofort mit der automatischen Einstellung (auto-gain/auto-exposure). Diese und weitere Einstellungen können vom RasMux beeinflusst werden, indem er spezielle Befehle über den I^2C-Bus ([Phi00]) an die RasCam schickt. Der RasMux kann so, nach dem Einschalten, die RasCam konfigurieren, was die gesamte Funktionsvielfalt des Bildsensors *VV5430* zugänglich macht. Außerdem muss der RasMux die analogen Bild- und Synchronisationssignale an seinen Ausgang durchschalten. Es kann daher immer nur eine RasCam pro RasMux aktiv sein.

An dieser Stelle soll noch darauf hingewiesen werden, dass es zwei Versionen (V1 und V2) von RasMux und zugehöriger PC-Verbindung gibt. Näheres dazu im Abschnitt 2.2. Die RasLeds und RasCams sind bei beiden Versionen identisch.

Die Bildinformation besteht aus dem Bild- (oder Video-) Signal im sog. *CCIR* Format (entspricht im Wesentlichen einem schwarz/weiß PAL Signal). Da bei diesem Format nur die Zeilensprünge, nicht aber die Grenzen zwischen den Bildpunkten definiert sind, wird zusätzlich noch ein Pixeltakt übertragen. Dies ermöglicht, die einzelnen Pixel einer Zeile zu identifizieren, was sehr wichtig ist, um die horizontale Position der Maske eindeutig zu bestimmen. Andernfalls würde es zu Sprüngen kommen können.

2.2 Verwendung von Rasnik unter Linux

Als ich das Projekt *Rasnik for Linux* begonnen habe (Anfang 2001), stand nur die RasMux Version V1 zur Verfügung. Das Windows-Programm „Icaras“ war die einzige Software, um das Rasnik zu betreiben. Icaras ist das Programm, das NIKHEF für den Betrieb des Rasnik-Systems bereit stellt. Der RasMux V1 wird vom Computer über den Parallelport (Drucker-Port) gesteuert. Die Software muss zur Kommunikation mit dem RasMux das *JTAG*-Protokoll[5] erzeugen.

2.2.1 Portierung auf Linux

Von NIKHEF wurden mir freundlicherweise die Quellen von Icaras (Version 4.3.3.7) und der Analyse-Routine (Versionen 1.3.3 bis 1.3.5) zur Verfügung gestellt. Allerdings wollte ich nicht einfach ein Icaras für Linux programmieren. Ich wollte die einzelnen Komponenten - RasMux-Steuerung, Bildaufnahme (Capture) und Bildanalyse (Fit) – unabhängig voneinander zugänglich machen. Diese „Einzelpakete“ (Programm-Module) lassen sich dann einfach zu den gewünschten Anwendungen verbinden.

Das Paket für die RasMux-Steuerung stellte eine besondere Herausforderung dar. Die beiden RasMux Versionen werden von der Software vollkommen unterschiedlich angesprochen. Für die erste Version muss das Programm das komplizierte *JTAG*-Protokoll erzeugen – bei der neueren Version ist an seine Stelle eine Text-Kommunikation über die *RS-232*-Schnittstelle getreten. Die Anwendungsschnittstelle (*API*) meines Paketes sollte jedoch so ausgelegt sein, dass das Paket problemlos durch ein Modul für den RasMux V2 ersetzt werden kann. Die anderen Pakete und die Anwendung sind davon nicht betroffen.

Steuerung des RasMux

Icaras ist eine C++-Software für Windows und macht intensiven Gebrauch von Windows-typischen Bibliotheken, wie den *MFC*[6] . Von Icaras habe ich die Routinen (oder Klassen) übernommen, welche die Abwicklung des *JTAG*-Protokolls, die Steuerung des RasMux sowie die I^2C-Kommunikation implementieren.

Um diese Klassen (`Jtag`, `RascamUnit`) unter Linux zu übersetzen, war es nötig, verschiedene Routinen aus Windows-Bibliotheken bereitzustellen. Einige konnten auf äquivalente Funktionen aus den *GNU*-Bibliotheken abgebildet werden, andere wurden neu geschrieben. Sie wurden von mir in den „Header-Dateien“ stdafx.h und CString.h zugänglich gemacht, die unter

[5] JTAG oder IEEE 1149.1 beschreibt einen Standard, um elektronische Bauteile im Betrieb kontrollieren, konfigurieren und testen zu können — siehe [ASS00].

[6] MFC = Microsoft Foundation Classes, eine Bibliothek von Windows

```
BOOL ParPort::InitParPort()
{
  if (fd < 0)
    {
      if ((fd = open(LPT_DEV, O_RDWR)) < 0)
        {
          ERR(errno,"open /dev/parport failed");
          return FALSE;
        }
    }
  if (ioctl (fd,PPEXCL) < 0)
    {
      ERR (errno,"PPEXCL");
      close (fd);
      fd = -1;
      return FALSE;
    }
  while (ioctl (fd,PPCLAIM) < 0)
    {
      if ( errno == ENXIO )
        {
          sleep(1);
          continue;
        }
      ERR (errno,"PPCLAIM");
      close (fd);
      fd = -1;
      return FALSE;
    }
  fcntl(fd, F_SETFD, FD_CLOEXEC);
  BYTE byte;
  ioctl(fd, PPRCONTROL,&byte);
  byte |= PARPORT_CONTROL_STROBE; // set STROBE
  ioctl(fd, PPWCONTROL,&byte);
  return TRUE;
}

BOOL ParPort::WriteParPort(BYTE bytes[], int count)
{
  if (fd < 0)
    return FALSE;
  for(int i = 0; i < count; ++i)
    ioctl(fd, PPWDATA,bytes+i);
  return TRUE;
}

BOOL ParPort::ReadParPort(BYTE bytes[], int count)
{
  if (fd < 0)
    return FALSE;
  for(int i = 0; i < count; ++i)
    ioctl(fd, PPRSTATUS, bytes+i);
  return TRUE;
}
```

Abbildung 2.3: *Ein Ausschnitt aus der Klasse* `ParPort` *– die Verbindung zwischen Betriebssystem und RasMux-Treiber. Er zeigt den regen Gebrauch von* `ioctl();`.

Windows eine andere Bedeutung haben und unter Linux nicht vorhanden sind.

So wird die Steuerung des Parallel-Ports – siehe Abbildung 2.3 – durch das Linux-Modul „*pp_user*“ abgewickelt, das es jedem Benutzer ermöglicht, die einzelnen Leitungen des Ports anzusprechen. Der entscheidende Teil fällt dabei der Methode `BOOL ParPort::InitParPort()` zu, da sie den Zugriff auf die Schnittstelle ermöglicht und die Datenleitungen initialisiert. Die Methoden `BOOL ParPort::WriteParPort( BYTE bytes[], int count)` und `BOOL ParPort::ReadParPort( BYTE bytes[], int count)` koordinieren den Datentransport. Die Kommunikation mit dem Linux-Kernel wird im Wesentlichen über die Funktion `ioctl(fd,cmd,data)` abgewickelt (siehe manpage ioctl(2)). Die verfügbaren Befehle sind in [Wau00] aufgelistet.

Damit steht nun eine Klassenhierarchie `RascamUnit` → `Jtag` → `ParPort` zur Verfügung, mit der der RasMux gesteuert werden kann. Im Hinblick auf den RasMux V2 sollte jedoch eine kompatible Schnittstelle geschaffen werden. Der RasMux V2 besitzt einen eigenen Prozessor[7] und wird über „Text-Befehle“ via *RS 232* und nicht über *JTAG* gesteuert. Diese Steuerung ist viel einfacher und aus jeder Programmiersprache heraus möglich, im Gegensatz zum Aufruf von Klassenmethoden. Ich hielt es daher für den elegantesten Weg, den RasMux V1 über die gleichen Text-Befehle – soweit wie möglich – zu steuern. So habe ich eine Routine (`const char *RasCmd(const char *cmd);`) geschrieben, die einen String als Befehl interpretiert und den RasMux V1 entsprechend steuert. Für das Anwendungsprogramm ist es nun unbedeutend, ob der Befehl an einen RasMux V1 oder RasMux V2 geht. Die Routine versteht die meisten Befehle des RasMux V2 (vlg. [GR02a]), wofür einige Prozeduren aus den Icaras-Quellen erweitert werden mussten. So wurde unter anderem die Auslese der I^2C-Register hinzugefügt, die in Icaras überhaupt nicht implementiert ist. Der Zugriff erfordert eine komplizierte Programmierung – es müssen zunächst die *JTAG*-Register des RasMux richtig angesprochen werden, da dieser den I^2C-Bus steuert. Dann müssen aus den entsprechenden Registern die Daten zusammengesucht werden, wobei sich zeigte, dass der RasMux das höchstwertigste Bit nicht zurückliefert. Dadurch kann das 8. Bit des „Coarse exposure“ Werts nicht gelesen werden. Bis auf dieses Bit erlaubt die Routine die vollständige Kontrolle des RasMux V1 – im Einzelnen sind dies:

[7] Genau genommen sitzt der Prozessor im *MiniMasterMux* und nicht im RasMux, aber hier werden beide zusammen als Einheit betrachtet und als „RasMux V2“ bezeichnet. Ebenso ist mit „RasMux V1“ der eigentliche RasMux V1 und der *BricoMux* gemeint — für die Software macht das keinen Unterschied.

L0*ml*	schaltet RasLed l an Muxer m ein.
C0*mc*	schaltet RasCam c an Muxer m ein.
O, U	schaltet alle RasLeds und RasCams ab.
R	liest die Konfigurationsregister der aktiven RasCam aus.
W*rdddd*	schreibt Register r der RasCam über den I^2C-Bus.
S	liefert den Status über die Methode `RascamUnit::getStatus()`.
V	gibt das Produktionsdatum und einen Hinweis zurück. Beispiel: `99/11/10 - MiniMasterMux Interpreter V1.3`

Aus technischen Gründen, z.B. der unterschiedlichen Anzahl von Ausgängen, ergaben sich leichte Abweichungen, die jedoch nur in speziellen Fällen von Bedeutung sind. Für weitere Details sei auf Anhang A verwiesen.

Bildaufnahme — Capture

Neben der RasMux-Steuerung muss der Computer auch das RasCam-Bild aufnehmen. NIKHEF verwendet dazu die „Frame grabber Karte“ *DT 3152* von DATA TRANSLATION (http://www.datx.de/), die sehr flexibel einsetzbar ist und unter anderem auch den Pixel-Takt auswertet. Für diese Karte gab es bis dahin keinen Treiber für Linux. Es sollte jedoch die gleiche Hardware wie unter Windows verwendet werden, zum einen, um kompatibel zu bleiben (Linux und Windows auf dem gleichen Rechner), und zum anderen, um die gleichen Resultate zu gewinnen (keine Veränderung der Messung).

Nach mehreren Kontakten mit dem Hersteller wurde mir freundlicherweise eine firmeninterne Dokumentation [Dat95] zur Verfügung gestellt. Mit dieser Hilfe gelang es mir schließlich, einen Treiber für Linux zu entwickeln. Er sollte die *DT 3152* soweit wie nötig unterstützen und sich dabei nahtlos in die Linux-Architektur einfügen, die eine Video-Schnittstelle (*V4L*)[8] besitzt [Cox00].

Der Treiber ist in der Lage einzelne Bilder aufzunehmen. Die Funktionen der Karte, ganze Filmsequenzen aufzuzeichnen, wurden nicht mit in den Treiber aufgenommen. Sie sind für unser Projekt nicht nötig und hätten ein intensives „Interrupt-Management“ zur Folge gehabt, da nach jedem Bild die Zieladressen der Daten angepasst werden müssen.

Mit dem Treiber können Eingangskanal, Video-Format, Helligkeit, Kontrast und Synchronisationsparameter gesetzt oder die *LUT*[9] geladen werden. Im Folgenden werde ich die Arbeitsweise des Treibers kurz erläutern. Wenn das Modul in den Kernel geladen wird, wird über die Hersteller- und Geräte-ID der *PCI*-Bus nach einer *DT 3152*-Karte durchsucht. Von der gefundenen Karte werden dann die Adressen und Interrupt-Nummern ermittelt (`dt_find_board()`).[10] Ist bis hierhin alles gut verlaufen, so registriert sich der Treiber beim *V4L*-Subsystem als Videogerät. Als nächstes müssen

[8] *Video for Linux*, seit Kernel 2.1

[9] Look Up Table, eine Umrechnungstabelle der Helligkeitswerte.

[10] Die *PCI*- und Interrupt-Behandlung wurde teilweise von einem ähnlichen Projekt

`VIDIOCGCAP`: Abfragen der Treiberfähigkeiten, wie Anzahl der Kanäle und der maximalen Bildgröße.

`VIDIOCGCHAN/VIDIOCSCHAN`: Lesen / Setzen des Eingangskanals, sowie der vordefinierten Videoformate.

`VIDIOCGPICT/VIDIOCSPICT`: Lesen / Setzen der Bildeinstellungen, wie Helligkeit, Kontrast, Farbpalette.

`VIDIOCGCAPTURE/VIDIOCSCAPTURE`: Einstellen des Bildbereiches.

`VIDIOCGINCREMENT/VIDIOCSINCREMENT`: Ermöglicht das Überspringen von Zeilen oder Spalten bei der Bildaufnahme.

`VIDIOCGLUT/VIDIOCSLUT`: Lesen / Schreiben der Look-Up-Table, ermöglicht ein Umrechnen der Videodaten.

`VIDIOCGSYNC/VIDIOCSSYNC`: Zugriff auf die verschiedenen Synchronisationsmöglichkeiten der Karte.

`VIDIOCGPCLK/VIDIOCSPCLK`: Programmieren des Taktgenerators *DP8534* auf der Karte.

`VIDIOCGDIGOUT/VIDIOCSDIGOUT`: Zugriff auf die digitalen Ausgänge.

`VIDIOCSFORMAT`: Ermöglicht das Definieren des Videoformates auf den Pixel genau.

Tabelle 2.1: *Die* `ioctl()`*-Befehle, die mein* DT 3152*-Treiber kennt.*

die Register der Karte auf das verwendete Video-Format programmiert werden. Der Zugriff auf den AD-Konverter (*Bt252*) geschieht dabei über einen I^2C-Bus, der vom *PCI*-Kontroller (*SAA7116*) gesteuert wird. Zur Vereinfachung kann gleich beim Laden des Moduls das richtige Video-Format (z.B. `VV5433` für die RasCam) angegeben werden.

Für das Rasnik werden folgende Einstellungen durch die Funktion `dt_set_VV5430()` gemacht:

- Der aktive Bereich des Bildes wird auf die Spalten 68 – 464 und Zeilen 18 – 306 programmiert.
- Die Synchronisation wird auf den externen Pixel-Takt gesetzt.
- Es werden die Schwellen für den A/D-Wandler und die Synchronisation definiert.

(http://sourceforge.net/projects/dt3155a/) übernommen.

```
extern void initanalysis_ (int *ver);
extern void exitanalysis_ ();
extern void doanalysis_ (struct RasNIN *n_in,struct RasFIN *f_in,
                         char image[], int *err);

void windout_(char *msg,int length);
void fileout_(char *msg,int length);
```

Abbildung 2.4: *Die Prototypen der Funktionen, über die der Zugriff auf die Analyseroutine erfolgt.*

Nun ist die Karte bereit Bilder aufzunehmen. Das Anwendungsprogramm muss zunächst die Gerätedatei (z.B. /dev/video0) öffnen und erhält bei allen folgenden Leseoperationen ein neues Bild. Nach dem Öffnen können einige Parameter der Karte über `ioctl()`-Aufrufe gesetzt oder abgefragt werden. Die Befehle sind in Tabelle 2.1 aufgelistet.

Dieser von mir entwickelte Treiber wurde, wegen seiner allgemeinen Benutzbarkeit, anderen Nutzern über das Internet zugänglich gemacht und steht auf der Webseite http://dt3152.sourceforge.net/ zum Download bereit.

Bildanalyse — Positionsfit

Ein weiteres Paket ist nötig, welches die Rasnik-Bilder auswertet. Es wurde hier die Analyse-Routine von NIKHEF verwendet, die Analyse der Brandeis Universität ist aber ebenso einsetzbar. Letztere wird zur Zeit für die Verwendung unter Linux vorbereitet und soll in Kürze funktionsfähig sein.

Die NIKHEF-Analyse ist in *Fortran* geschrieben. Sie ließ sich nach leichten Änderungen (Präprozessor-Anweisungen und Formate von F 90) mit dem *GNU*-Fortran-Compiler (**g77**) übersetzen. Die Änderungen beziehen sich lediglich auf die Formate der Ausgaben, der gesamte Algorithmus wird *unverändert* übernommen. Mit einer passenden „Header-Datei“ (Abbildung 2.4) lässt sich die Analyse problemlos in einem C++ -Programm verwenden. Allerdings ist die Datenrückgabe, sie ist über „call-back-Funktionen“ gelöst, umständlich.

Die ersten drei Funktionen in Abbildung 2.4 werden in der *Fortran*-Datei definiert, die die Analyse enthält. Die Parameter werden über zwei Strukturen, eine für ganze Zahlen (`RasNIN`) und eine für Fließkommazahlen (`RasFIN`), übergeben. Die Ergebnisse werden dann über die Funktionen `windout_` und `fileout_` an die Anwendung zurückgegeben. Diese beiden müssen in dem Anwendungsprogramm definiert sein und aus den übergebenen Strings die Daten heraussuchen.

Somit sind alle Einzelkomponenten unter Linux vorhanden und können zu einer Anwendung zusammengesetzt werden.

2.2.2 Skriptsteuerung — Tcl/Tk-Einbindung

Es bleibt noch die Frage zu klären, wie und welche Anwendungen geschrieben werden sollen. Sehr bald stellten sich verschiedene Bedürfnisse heraus. Neben einigen Programmen zum Test der einzelnen Hard- und Software-Komponenten wurden auch Programme zur Bildauswertung, Linsenvermessung und Kammerbau benötigt. Die Programme sollten schnell und einfach an die verschiedenen Bedürfnisse angepasst werden können. Zusätzlich sollten auch graphische Oberflächen unkompliziert erstellt werden können. So fiel die Wahl auf die Skriptsprache Tcl/Tk (http://www.tcl.tk), die sowohl schnelle Programmentwicklung als auch graphische Benutzerschnittstellen ermöglicht. Ich habe auch über den Einsatz von *Perl* nachgedacht – *Perl* ist für Berechnungen besser geeignet, aber die Verbindung mit *C* oder C++ ist komplexer und komplizierter. Daher wurde die Unterstützung von *Perl*, nach der Einbindung der RasMux-Klassen, eingestellt.

Tcl/Tk bietet standardmäßig eine Möglichkeit mit Bildern[11] zu arbeiten, somit wird die Handhabung der Rasnik-Bilder stark vereinfacht. Ein weiterer Punkt, der für den Einsatz von Tcl/Tk spricht, ist, dass in unserer Arbeitsgruppe schon einige Erfahrung mit dieser Sprache gesammelt wurde, auf die ich zurückgreifen konnte. Zudem kann so die weitere Pflege der Software durch unsere Arbeitsgruppe erfolgen.

Nachdem ich diese grundlegenden Strukturen festgelegt hatte, entwickelte ich drei Tcl/Tk-Pakete (`RasV1Tcl`, `CaptureTcl`, `NikAnaTcl`), die den RasMux steuern, ein Bild einlesen und dieses dann auswerten. Sie sind die Grundlage für alle meine Anwendungen.

Im Folgenden ist ein Programm abgedruckt, das zeigt, wie einfach eine Rasnik-Analyse in Tcl/Tk durchführbar ist:

```
package require RasV1Tcl   ;# lade Rasmux Kontrolle
package require CaptureTcl ;# lade Video Capture
package require NikAnaTcl  ;# lade Bildauswertung

RasCmdV1 L000 C002         ;# schalte LED0 und Cam2 an
after 1000                 ;# 1 sec. warten
Capture -image img         ;# Bild nehmen
RasCmdV1 O                 ;# Rasnik ausschalten
puts [NikAna img -XYMSK .12 -XYPIC .012 -DCCD 1892 -DMASK 1885 \
  -TO_LENS 1]              ;# analysieren und ausgeben
exit                       ;# Ende
```

Das Paket `RasV1Tcl` beinhaltet die Steuerung des RasMux und stellt den Befehl `RasCmdV1` bereit, der alle Argumente als Kommandos an die Prozedur `RasCmd()` übergibt.

[11] siehe manpage image(n)

`-DMASK:`	Der Abstand Linse zu Maske in mm.
`-DCCD:`	Der Abstand Linse zu Kamera in mm.
`-XYPIC:`	Die Pixelgröße der Kamera in mm.
`-XYMSK:`	Die Feldgröße der Maske in mm.
`-IXOK:`	Gibt an, ob die Maske gespiegelt ist.
`-IYOK:`	= 1, wenn die Maske auf dem Kopf steht.
`-TO_CENTER:`	Rekonstruktion der Maskenmitte oder der linken oberen Ecke.
`-TO_LENS:`	Bestimmung der Masken- oder Linsenposition.
`-ANG_XY:`	Berechnung der Winkel einschalten.
`-LEVEL:`	Umfang der „Debug-Ausgabe“.
`-DEBUG:`	„Debug-Ausgabe“ aktivieren.
`-BANDS:`	Bestimmen der Seitenbänder.
`-PLOTS:`	Generierung von Plots.
`-FIX_BW_WB:`	Korrektur der Schwarz-Weiß-Werte.
`-file:`	Tcl/Tk-Kanal, der die „Log-Daten“ bekommt.
`-winvariable:`	Tcl/Tk-Variable, in die die Benutzerausgabe abgelegt wird.

Tabelle 2.2: *Die Optionen des Befehls* `NikAna`*. Die Parameter werden direkt an die Analyse übergeben und dort ausgewertet. Je nach Analyseversion sind einige Optionen überflüssig.*

In dem Paket `CaptureTcl` ist auch nur ein Befehl enthalten. Mit `Capture` wird ein Bild aufgenommen und in einem Tcl/Tk-Image (siehe manpage photo(n)) gespeichert. Der Befehl kann durch mehrere Optionen beeinflusst werden, die in Anhang A beschrieben sind.

Das Paket `NikAnaTcl` stellt die Analyse bereit. Neben dem Bild werden die Parameter als Optionen an den Befehl `NikAna` übergeben. Die möglichen Optionen sind in Tabelle 2.2 beschrieben.

Ein weiteres Paket `NikFile` enthält eine Reihe von Hilfsroutinen, die in Tcl/Tk geschrieben sind. Sie ermöglichen es, Icaras-Dateien zu lesen und zu schreiben. Zudem sind darin einige Prozeduren definiert, die das Bearbeiten (Editieren) der Parameter von Rasnik-Strecken oder die Auswertung von „Log-Dateien“ vereinfachen.

2.3 *Rasnik for Linux* Funktionstest

Die Analyse-Routine wird aus dem gleichen Quellcode wie die Windows-Variante übersetzt, aber das bedeutet noch nicht, dass sie auch exakt die gleichen Ergebnisse liefert. Unter Linux wird der *GNU*-Compiler verwendet, unter Windows der Microsoft-Compiler. Um nun die beiden Routinen zu vergleichen, wurden mit Icaras Bilder analysiert (Ver. 1.3.5) und gespeichert.

Name	**Seq.**	Δx $[mm]$	$\frac{\Delta x}{\sigma x}$	Δy $[mm]$	$\frac{\Delta y}{\sigma y}$
PAL	1	0.000000	0.00	0.000000	0.00
DAL	1	0.000060	0.09	0.000080	0.07
PSL	1	0.000050	0.53	0.000015	0.17
DSL	1	0.000020	0.16	0.000010	0.08
DSR	1	0.000030	0.36	0.000020	0.22
PSR	1	0.000026	0.36	0.000002	0.03
DAR	1	0.000090	0.33	0.000020	0.07
PAR	1	0.000020	0.08	0.000013	0.05
PAL	2	0.000040	0.13	0.000110	0.39
DAL	2	0.000020	0.03	0.000050	0.06
PSL	2	0.000020	0.21	0.000004	0.05
DSL	2	0.000020	0.17	0.000010	0.08
DSR	2	0.000090	1.08	0.000020	0.22
PSR	2	0.000009	0.13	0.000003	0.04
DAR	2	0.000150	0.53	0.000050	0.18
PAR	2	0.000014	0.05	0.000000	0.00
PAL	3	0.000090	0.31	0.000170	0.59
DAL	3	0.000040	0.06	0.000040	0.04
PSL	3	0.000010	0.10	0.000006	0.07
DSL	3	0.000110	0.89	0.000040	0.29
DSR	3	0.000020	0.24	0.000010	0.11
PSR	3	0.000020	0.27	0.000003	0.04
DAR	3	0.000220	0.77	0.000060	0.21
PAR	3	0.000006	0.02	0.000009	0.03

Tabelle 2.3: *In dieser Tabelle sind die Fit-Ergebnisse von Icaras denen unter Linux (g77) gegenübergestellt. Auf beiden Systemen wurden die gleichen Bilder verwendet.*

Diese Bilder wurden dann unter Linux nochmals analysiert (Ver. 1.3.5) und die Fit-Ergebnisse miteinander verglichen. In Tabelle 2.3 sind die Ergebnisse eines Tests abgedruckt. Wie man sieht, gibt es durchaus Unterschiede innerhalb von $0.1\,\mu\mathsf{m}$, obwohl es sich um exakt den gleichen Algorithmus handelt. Das ist aber nicht weiter störend, da diese Unterschiede deutlich kleiner als die Genauigkeiten des Systems sind. Jedoch sollten die Abweichungen der Ergebnisse bei *unterschiedlichen Compilern* oder *Codeversionen* als Anlass genommen werden, die **Null-Positionen** als **Bilder** *zu speichern!* Damit kann bei einem Wechsel der Analyse die Null-Position erneut bestimmt werden.

Der Test der Bildaufnahme ist schwieriger, da nicht zweimal exakt das gleiche Bild aufgenommen werden kann. Im Rahmen der Messgenauigkeit ($< 1\,\mu\mathsf{m}$) liegen die Fit-Ergebnisse zusammen. Es wurden zwei Messreihen des Kammerrahmens in kurzem Abstand hintereinander durchgeführt, einmal mit Icaras und einmal unter Linux. Die Tabelle 2.4 zeigt zwei Ausschnitte aus den „Log-Dateien“. Es wurden jeweils nur die (gekürzten) Zei-

```
st,000001,2002/10/31,13:58:13, DAL ,0.0.0.,0045,0135
ud 0 8.776646E+01 3.114481E+01 2.259062E+00 -1.3202E-01
st,000001,2002/10/31,13:58:14, DAR ,0.3.0.,0045,0135
ud 0 4.803424E+01 7.088607E+01 2.260036E+00 -4.9513E-01
st,000001,2002/10/31,13:58:16, DSL ,0.1.0.,0045,0135
ud 0 6.667234E+01 5.337379E+01 1.005538E+00 -7.9098E-03
st,000001,2002/10/31,13:58:18, DSR ,0.2.0.,0045,0135
ud 0 6.331377E+01 3.152140E+01 1.004809E+00 -3.1715E-03
st,000001,2002/10/31,13:58:20, PAL ,0.0.0.,0045,0135
ud 0 2.954435E+01 5.186744E+01 2.254664E+00 -4.1851E-02
st,000001,2002/10/31,13:58:22, PAR ,0.3.0.,0045,0135
ud 0 8.624084E+00 1.163056E+01 2.258161E+00 4.6875E-02
st,000001,2002/10/31,13:58:24, PSL ,0.1.0.,0045,0135
ud 0 4.410853E+01 1.297465E+01 9.944661E-01 -1.9750E-02
st,000001,2002/10/31,13:58:25, PSR ,0.2.0.,0045,0135
ud 0 6.208799E+00 1.032393E+01 9.976185E-01 1.1091E-02
```

```
st,000001,2002/10/31,14:07:11, DAL ,0.0.0.6,0011,0135
ud 0 8.776648E+01 3.114479E+01 2.259172E+00 -6.6721E-02
st,000001,2002/10/31,14:07:26, DAR ,0.3.0.1,0011,0135
ud 0 4.803424E+01 7.088607E+01 2.260036E+00 -4.4292E-01
st,000001,2002/10/31,14:07:39, DSL ,0.1.0.7,0011,0135
ud 0 6.667233E+01 5.337378E+01 1.005542E+00 2.2434E-03
st,000001,2002/10/31,14:08:01, DSR ,0.2.0.0,0011,0135
ud 0 6.331376E+01 3.152140E+01 1.004808E+00 -2.0662E-03
st,000001,2002/10/31,14:08:16, PAL ,0.0.0.2,0011,0135
ud 0 2.954434E+01 5.186745E+01 2.254669E+00 -7.6607E-02
st,000001,2002/10/31,14:08:31, PAR ,0.3.0.5,0011,0135
ud 0 8.624881E+00 1.163069E+01 2.257333E+00 -5.1478E-02
st,000001,2002/10/31,14:08:48, PSL ,0.1.0.3,0011,0135
ud 0 4.410855E+01 1.297465E+01 9.944679E-01 -2.0440E-03
st,000001,2002/10/31,14:09:03, PSR ,0.2.0.4,0011,0135
ud 0 6.208788E+00 1.032393E+01 9.976153E-01 2.9135E-03
```

Tabelle 2.4: *Ausschnitte aus zwei „Log-Dateien“, die mit Icaras (oben) und mit Rasnik for Linux (unten) erzeugt wurden; aufgenommen mit kurzem zeitlichen Abstand. Die „st“-Zeilen enthalten unter anderem die Zeit und den Namen (z. B.* `DAL` *in der ersten Zeile). Im 3. bis 5. Feld der „ud“-Zeilen stehen x, y und Vergrößerung. Wie man sieht, unterscheiden sich die x- und y-Werte nur gering, d.h. im Sub-μm-Bereich. Man erkennt auch einen Programmfehler in Icaras (Ver. 4.5), es fehlt die Nummer der RasLed (z. B. in der 1. Zeile die* `6`*).*

len eingetragen, die den Namen oder die Fit-Ergebnisse angeben. Die „ud“-Zeilen enthalten im dritten und vierten Feld die x, y-Positionen des Fits. Den größten Unterschied findet man bei dem Sensor *PAR* (Zeile 12), er beträgt $0,8\,\mu\mathsf{m}$, sonst liegt er unter $0,1\,\mu\mathsf{m}$. Die beiden Programme liefern praktisch die gleichen Werte, womit gezeigt ist, dass die Auswertungskette unter Linux äquivalent zu der Windows-Variante ist.

Die Analyse-Routine lässt sich unter Linux auch mit dem INTEL-Fortran-Compiler übersetzen. Der damit erzeugte Code ist um fast 50% schneller als die **g77** Variante. Leider stellte sich dabei heraus, dass, je nach verwendeten Compiler-Optionen des INTEL-Übersetzers, die Analyse verschiedene Ergebnisse liefert, sogar bei den identischen Bildern. Das heißt, dass der Code des INTEL-Compilers stark vom Übersetzungsprozess abhängt Beim **g77** wurden diese Abhängigkeiten von den Optionen nicht beobachtet. Ich entschied mich daher, ausschließlich den *GNU*-Compiler zu verwenden, um die Anzahl der Analyse-Varianten nicht noch durch verschiedene Compiler und deren Optionen zu vergrößern.

In diesem Kapitel wird die Funktionsweise der Rasnik-Sensoren dargelegt. Es wird meine Software zur Benutzung dieser Sensoren mit Linux vorgestellt. Der Vergleich mit dem Windowsprogramm Icaras zeigt leichte Unterschiede auf, die auf die unterschiedlichen Compiler zurückzuführen sind. Diese Abweichungen schränken die Verwendungsmöglichkeit nicht ein, sie erschweren nur den Datenvergleich zwischen beiden Systemen.

Kapitel 3

Untersuchungen zum Rasnik-System

Dieses Kapitel stellt zunächst einige Untersuchungen vor, die mit Rasnik-Komponenten auf einer „optischen Bank“ durchgeführt wurden. Dieser Aufbau wurde dann zu einem Linsenmessstand erweitert. Im weiteren Verlauf werden die Rasniks im „BOG-Spacer“ eingehend betrachtet.

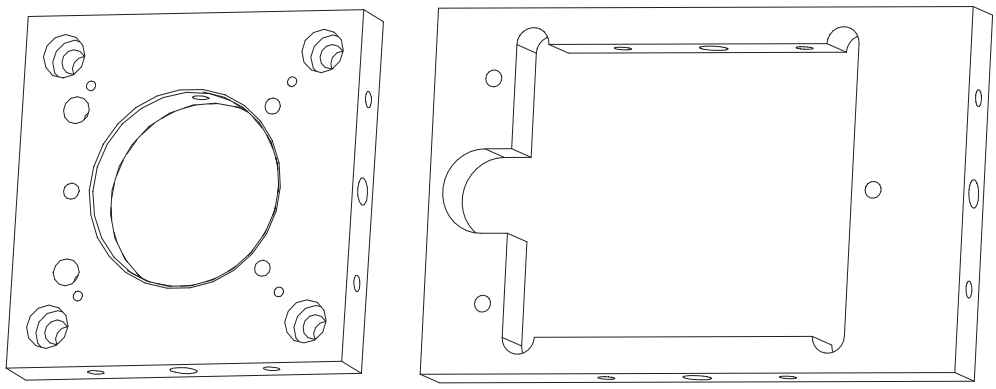
3.1 Die *Rasnik-Bank*

Zunächst soll das Rasnik-System auf der „optischen Bank“ – kurz „*Rasnik-Bank*“ – beschrieben werden. Für den Aufbau wurden Komponenten des *SYS 65* von der Firma Owis (http://www.owis-staufen.de) verwendet. Als optischen Bank wurden zwei 2 m lange Rohre (Typ: *Profil 65-4*) fest miteinander verschraubt. Die Gesamtlänge von 4 m ermöglicht es, alle Rasniks der BOG-Kammern aufzubauen. Ich habe Halterungen entwickelt, die zum einen auf die Trägerplatten des *SYS 65* passen und zum anderen die einzelnen Rasnik-Komponenten aufnehmen (siehe Abbildung 3.1). Die Lochmuster für die optischen Komponenten entsprechen denen an dem Spacer. Die Masken können jedoch direkt auf den Halter – d.h. ohne Abstandsröllchen – geschraubt werden. Diese kompaktere Bauweise erforderte weitere Löcher, damit die LED-Platine aufgesteckt werden kann.

Beim hier verwendeten *SYS 65* liegt die optische Achse genau $32,5\,\mathrm{mm}$ über den Trägerplatten. Die RasCam ist also zu groß für das *SYS 65*, so dass das gesamte System um 90° gedreht werden musste.

Die Halter für die RasLens sind so konstruiert, dass entweder die „nackten“ Linsen genau in die Mitte des Halters passen oder die verklebten Linsen aufgesteckt werden können.

Mit diesen Komponenten wurden nun alle unterschiedlichen Rasnik-Strecken der BOG-Kammern nachgebaut. Es standen dazu pro Brennweite



RasLed-Halter *RasCam-Halter*

Abbildung 3.1: *Diese Abbildung zeigt die Aufnahmeplatten für die RasLed (links) und die RasCam (rechts). Die Rasnik-Komponenten werden jeweils mit drei Schrauben an die Halter geschraubt, welche selbst kompatibel zum* SYS 65 *sind.*

je vier Linsen zur Verfügung, die vorab geliefert wurden. Die einzelnen Systemgrößen sind hier noch einmal zusammengetragen:

Name	d_{Mask}	d_{CCD}	d_{SYS}	M	f
PS	1853	1853	3706	1	927
DS	1892	1892	3784	1	946
PA	1136	2570	3706	2.3	788
DA	1164	2634	3798	2.3	807

Alle Längen sind in mm angegeben, es bedeuten dabei:

d_{Mask}	Der Abstand zwischen Maske und Linse.
d_{CCD}	Der Abstand zwischen Kamera und Linse.
d_{SYS}	Länge des gesamten Systems (Kamera – Maske).
M	Die Vergrößerung des Systems.
f	Die Brennweite der Linse.

Bei den Namen wird nach parallel (P) zu den Röhren oder diagonal (D), und nach symmetrisch (S) oder asymmetrisch (A), d.h. vergrößernd unterschieden.

Die Messungen zeigten eine Produktionsstreuung der Brennweiten von 3,5 mm. Es ist daher erforderlich, die Maskenposition einzeln anzupassen, indem die Abstandsröllchen entsprechend gewählt werden.

In Konsequenz bedeutet das, dass jede einzelne Linse vermessen werden muss. So wurde entschieden, einen Messstand für die Brennweiten aufzubauen.

Eine weitere Möglichkeit der *Rasnik-Bank* soll hier noch kurz erwähnt werden. Für den späteren Einbau der Rasniks kann es hilfreich sein, die Positionen auf der Maske zu kennen, die die Rekonstruktion ausgibt, wenn sie die

Mitte der Maske „sieht“. Dies wird erreicht, indem ein spezieller RasLed-Halter mit einer bekannten Maske justiert wird. Anschließend können die unbekannten Masken aufgesteckt und vermessen werden. Der verwendete Halter ist eine umgebaute LED-Platine, die elektrisch isoliert auf einen X/Y-Tisch geschraubt ist. Mit dieser Methode kann die Maskenmitte auf ca. $\frac{1}{2}$ mm genau bestimmt werden. Das ist ausreichend, um später die Lage der Linse in der Kammer zu kontrollieren.

3.2 Bestimmung der Linsenbrennweite

Es ist recht mühsam, die Brennweite einer Linse von Hand zu messen. An verschiedenen Stellen muss dazu die Schärfe eines Bildes beurteilt und die Position gemessen werden – das Ganze zudem noch mit infrarotem Licht! Die Brennweite der Linse hängt von der Wellenlänge ab.

3.2.1 Aufbau des Messstandes

Die Idee einer automatischen Messanordnung drängte sich förmlich auf. Zwei Probleme sind dabei zu lösen:

1. automatisches Verfahren der Linse oder Maske,

2. Definieren eines Maßes für die Schärfe.

Punkt 1 war schnell gelöst. Es standen verschiedene Motoren der Firma Isel (http://www.isel.com) mit Kontroller zur Verfügung. Sie lassen sich einfach vom Computer aus steuern und bieten eine gute Positioniergenauigkeit. Der Motor (Typ *234 403 0 109*) hat einen Verfahrweg von etwas über einem Meter. Somit können alle Linsen durch Verschieben der Maske bei einer 1 : 1-Abbildung vermessen werden. (Es werden alle Linsen 1 : 1 vermessen, damit dieselbe Maske verwendet werden kann.) Vier Linsenhalter und die RasCam sind dabei ortsfest, sodass beim Wechsel der Brennweite keine weiteren Kalibrationen nötig werden.

Punkt 2 erwies sich als deutlich schwieriger. Ich versuchte die Bilddaten statistisch auszuwerten, um daraus ein Schärfemaß zu definieren. Die Maske selbst ist ein schwarz/weiß Bild, das Rasnik liefert dann ein Graustufenbild mit 8-Bit Auflösung. Im Idealfall würde man also zwei Peaks im Histogramm erwarten, die bei zunehmender Unschärfe zusammenfließen.

Einige Messungen zeigten jedoch, dass wir weit von diesem „Idealfall“ entfernt sind. Bei unscharfen Bildern sind die Grauwerte tatsächlich gaußverteilt. Schärfere Bilder haben eine flachere und breitere Verteilung, die sich aber nicht in zwei Peaks aufteilt. Breite oder Höhe dieser Verteilung erwiesen sich ebenso ungeeignet wie Spektralanalysen mittels FFT[1].

[1] Schnelle Fourier-Transformation

Schlussendlich führte folgender Algorithmus zum Erfolg: Zunächst werden alle Differenzen der Helligkeiten zwischen nebeneinanderliegenden Punkten gebildet, also „linker minus rechtem Pixel“. Die Standardabweichung dieser Differenzen ist nun ein gutes Maß für die Schärfe eines Bildes, in dem Sinne, dass sich damit die schärfste Position finden lässt. Ich weise ausdrücklich darauf hin, dass es sich hierbei nicht um ein *absolutes* Maß handelt, da der Wert u.a. vom Umgebungslicht, wie auch von der Geometrie abhängt. Man kann also Bilder von verschiedenen Rasniks nicht miteinander vergleichen – das Schärfemaß dient nur der Positionsbestimmung.

Trägt man diese Größe gegen die Maskenposition auf, so ergibt sich ein nahezu gaußförmiger Verlauf. Der Schwerpunkt eines Gaußfits liefert nun den Punkt größter Schärfe.

3.2.2 Qualitätstest

Es stellt sich nun die Frage, wie genau man die Brennweite mit dem hier beschriebenen Aufbau bestimmen kann. Zur Erläuterung beschreibe ich kurz die Kalibration.

Die erforderlichen Abstände werden mit sog. „Parallelendmaßen“ eingestellt. Das sind sehr genaue Stahlstangen von definierter Länge. Die Längen der optischen Bank wie auch die der Endmaße unterliegen natürlich Temperaturschwankungen. Deshalb wird bei der Messung der Brennweiten und der Kalibration die Temperatur aufgezeichnet. Die Kalibration wurde bei einer Temperatur von $22,2\,^{\circ}\mathsf{C} \pm 0,2\,^{\circ}\mathsf{C}$ wie folgt durchgeführt:

1. Der Motor mit Maske wird auf Position 0 gefahren, die sog. Referenzposition. Sie wird durch einen Endschalter im Motor definiert (Wiederholgenauigkeit laut Hersteller $\pm 0,02\,\mathsf{mm}$).

2. Mit einem Endmaß wird der Abstand zwischen Maske (Glasplatte) und RasCam-Halter auf $3\,\mathsf{m}$ eingestellt. Das ergibt einen Abstand von Maske zum Kamera-Chip von $3007,0\,\mathsf{mm} \pm 0,1\,\mathsf{mm}$.

3. Die Position der Linsenhalter wird, ebenfalls mit Endmaßen, vom RasCam-Halter aus bestimmt. Dabei ist der Abstand der Halter um $12\,\mathsf{mm}$ kürzer zu wählen als der zwischen Chip und Linsenmitte. Ich gehe hier von einer $4\,\mathsf{mm}$ dicken Linse aus, sodass ihre Mitte mit der des Halters übereinstimmt ($5\,\mathsf{mm}$). Die RasCam liefert noch eine Verschiebung von $7\,\mathsf{mm}$, das sind zusammen $12\,\mathsf{mm}$.

Auf diese Weise wurden vier Linsenhalter aufgestellt, die an den Positionen $1892\,\mathsf{mm}$, $1853\,\mathsf{mm}$, $1614,5\,\mathsf{mm}$ und $1575,5\,\mathsf{mm}$ sitzen.

Eine Messreihe, bei der die gleiche Linse 10 mal aus- und eingebaut wurde, ergab eine Reproduzierbarkeit von $\sigma = 0,043\,\mathsf{mm}$. Zusammen mit der Unsicherheit des Aufbaus kann man also sagen, dass sich die Brennweite mit dem Messstand besser als $\pm 0,1\,\mathsf{mm}$ bestimmen lässt.

Id	f_{soll}	f_{ist}	Δ_{Mask}	σ	d_{Mask}	d_{CCD}	**Temp.**
807.001	807.30	809.6	5	5	1624	1614	22.2
807.002	807.30	809.5	5	5	1623	1614	22.1
807.003	807.30	808.9	3	5	1621	1614	22.1
807.004	807.30	808.7	3	5	1620	1614	22.1
788.001	787.80	790.7	6	5	1587	1576	22.2
788.002	787.80	790.5	6	5	1586	1576	22.2
788.003	787.80	790.1	5	5	1585	1576	22.2
788.004	787.80	790.6	6	5	1587	1576	22.2
946.001	946.00	944.2	-7	6	1885	1892	22.2
946.002	946.00	944.2	-7	6	1885	1892	22.2
946.003	946.00	945.1	-3	6	1889	1892	22.2
946.004	946.00	945.1	-4	6	1888	1892	22.2
927.001	926.50	930.1	15	6	1868	1853	22.2
927.002	926.50	928.5	8	6	1861	1853	22.2
927.003	926.50	929.7	13	6	1866	1853	22.2
927.004	926.50	929.3	11	6	1864	1853	22.2

Tabelle 3.1: *Die Messergebnisse der Linsen aus der ersten Lieferung. Alle Längen in mm, Temperaturen in* °C.

3.2.3 Die BOG-Linsen

Mit dem Programm LensMeasure.tcl (siehe Anhang A) wurden alle Linsen der ersten Lieferung (Batch) vermessen. Das Programm berechnet aus der Sollbrennweite sowie der Linsenposition die Maskenposition und sucht dann im Bereich ±25 mm mit 1 mm-Schritten (beides einstellbar) nach der schärfsten Stelle.

Die Ergebnisse werden dann zusammen mit einer eindeutigen Linsenbezeichnung (Id) in einer Datenbank abgelegt. (Die Bezeichnung steht bei den Linsen auf dem Rand und wird beim Einkleben auf die Halter übertragen.) Die „MDT Production Data Base“ (*MMPDB*), die von einem römischen Institut definiert ist, wurde um einige Tabellen erweitert, damit diese Daten aufgenommen werden können (siehe Anhang B). Ein Auszug der Datenbank mit den ersten Linsen ist in Tabelle 3.1 zu sehen. Die wichtigsten Daten sind die gemessene Brennweite (f_{ist}) und die Verschiebung der Maske aus der Sollposition (Δ_{Mask}). Die Temperatur ist die gleiche wie bei der Kalibration, so dass Längenänderungen nicht zu erwarten sind. Das eingetragene σ ist die Breite des Gaußfits bei der Schärfebestimmung.

Wenn man die Tabelle genauer betrachtet, so sieht man, dass die Brennweiten bis ca. $3,5$ mm (bei Linse 927.001) von dem Sollwert abweichen. Bei der ersten Lieferung sind das $1,8$ mm RMS. Für die Masken bedeutet das, dass sie zwischen -7 mm und $+15$ mm verschoben werden müssen. Die minimale Verschiebung liegt bei -10 mm, da Halter der LED-Platine 2 mm Abstand benötigt. Bei diesem Satz Linsen bleiben uns somit 3 mm „Luft“.

Abbildung 3.2: *Diese Bilder zeigen die Ausrichtung einer Linse. Im linken Bild ist die Linse exakt ausgerichtet, durch einen Anschlag auf der optischen Bank. Rechts wurde die Linse nur durch Begutachten des Bildes ausgerichtet, so wie es auch beim Einbau der Linsen in den Spacer geschieht. Nachgemessen ergab sich eine Abweichung von weniger als 0,3°.*

Es sind alle Linsen verwendbar und wir haben nach diesen Spezifikationen Linsen für alle Freiburger Kammern bestellt, d.h. je 40 Stück pro Brennweite. Zum Test dieser Lieferung wurden die Linsen mit 946 mm Sollbrennweite vermessen. Die Brennweiten schwanken nur um 0.55 mm *RMS*, d.h. geringer als bei der ersten Lieferung. Die Maskenverschiebung liegt bei diesen 40 Linsen zwischen -6 mm und $+2$ mm.

3.3 Messungen zum BOG-Rasnik

Der folgende Abschnitt beschreibt den Einbau der Rasniks in die erste Kammer (bzw. Spacer). Es wurde dabei überprüft, wie reproduzierbar (Stabilität) die Rasnik-Ergebnisse sind, und ob es Unterschiede zwischen den Strecken gibt. Gleichzeitig wurde geprüft, ob die „CAD-Entwürfe“ realisierbar sind.

3.3.1 Einbau der Rasniks

Für den Aufbau der ersten Kammer waren die Linsen aus Tabelle 3.1 vorgesehen. Sie wurden zwischenzeitlich in ihre kugelförmigen Halter[2] geklebt, wobei als Kleber „Araldite 2019“ (Komponenten: *AY 103*/*HY 991*) verwendet wurde. Die Halter ermöglichen es, die Linsen unter den benötigten Winkeln relativ zur Cross Plate zu montieren. Für den ersten Aufbau sollen die Winkel flexibel gehalten werden. Die Linsenhalter können also noch nicht mit den zugehörigen Flanschen[3] verklebt werden. Stattdessen werden sie

[2] siehe Zeichnung „*HOUSE D40*“ – CERN Id. Nr.: ATLMMACG_0003
[3] siehe Zeichnung „*FLANGE*“ – CERN Id. Nr.: ATLMMACG_0002

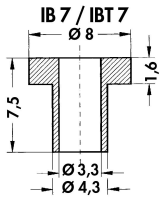
zwischen zwei Flansche geklemmt, die mit langen Schrauben an den Spacer geschraubt werden.

Mit Hilfe eines Hohlzylinders, der in den Halter geschoben wird, kann nun die Linse ausgerichtet und gleichzeitig das Bild begutachtet werden. Zur groben Justage werden die gut sichtbaren Verzerrungen im Bild minimiert. Der physiologische Schärfeeindruck dient dann zur feineren Einstellung. Darüber hinaus sollte man die Linse so justieren, dass möglichst viele Positionsmarkierungen im Bild vollständig sichtbar sind, was später der Analyse hilft. Diese Methode erlaubt es, die Linsen hinreichend genau auszurichten (siehe Abbildung 3.2).

Ist der Spacer zuvor ausgerichtet worden, z.B. auf der Koordinaten-Messmaschine, dann kann man nun die Linse so nachstellen, dass die Mitte der Maske zu sehen ist. Es muss dafür eine vermessene Maske verwendet werden, d.h. die Mitte muss bekannt sein. Der Rahmen ist so genau gefertigt, dass die Rasniks in ihren Sollpositionen genau in der Mitte stehen. Mit der Rasnik-Auslese kann so die Linse sehr genau justiert werden. (In Abbildung 3.2 erkennt man auch, dass sich die Bilder verschoben haben.)

Das Einbauen der Masken gestaltet sich recht einfach. Durch die Linse wird ja die Verschiebung der Maske festgelegt. Sie wird durch die Abstandsröllchen eingestellt, indem zur nominalen Länge von 12 mm die Verschiebung addiert wird. Die erforderliche Länge wird vom Programm mit den Werten aus der Datenbank berechnet. Dann wird dieser Abstand aus Röllchen mit Längen von 1 mm, 4 mm oder 12 mm zusammengesetzt. Die Abstandsröllchen und die Schrauben sind aus Messing, damit sie das Magnetfeld im ATLAS-Detektor nicht stören.

Die LED-Platine wird anschließend aufgesteckt, sie ist elektrisch isoliert (siehe [vdGGLR00]). Zuletzt wird die Verbindung zum RasMux (V2) mit einem vierpoligen Kabel hergestellt.



Als nächstes beschreibe ich den Einbau der RasCams. Das Wesentliche, was bei den RasCams zu beachten ist, ist die Isolierung. Die Isolation verhindert Masseschleifen im Detektor, die zu Störungen der Auslese führen würden. Sie werden bei uns mit Messingschrauben (M3×20) befestigt. Zwischen Schraube und Gehäuse kommen auf beiden Seiten „Isolierbuchsen“, wie sie z. B. zur Befestigung von Transistoren auf Kühlkörpern verwendet werden. Wir verwenden zur Montage der Kameras die Isolierbuchsen *IB 7* der Firma FISCHER ELEKTRONIK (http://www.fischerelektronik.de). Sie ragen ca. 6 mm in die RasCam hinein und passen exakt in die Bohrungen. Die RasCam wird dadurch um $1,6$ mm von der Cross Plate entfernt (kann durch Verschieben der Maske korrigiert werden). Die Buchsen gewährleisten die Isolierung von der Schraube und der Kammer.

3.3.2 Statistische Beurteilung

Im abschließenden Abschnitt des Kapitels wollen wir uns mit der erreichbaren Genauigkeit, genauer der Standardabweichung, der Rasniks in den BOG-Kammern beschäftigen. Die Unsicherheit der Rasnik-Auslesen hängt zum einen von der Bildqualität, wie Schärfe und Kontrast, und zum anderen von der Länge der Strecke ab. Die Analyse wird bei den MDTs so betrieben, dass sie die Position der Linse ausgibt. Die Bildposition wird mit Hilfe des Strahlensatzes auf die Linse umgerechnet, so dass sich eine Längenabhängigkeit ergibt. Da mein Programm auch die Warnungen der Analyse in die Ausgabedatei schreibt, konnten diese unsicheren Werte verworden werden.

Für diese Messung wurde nun der Spacer auf einem Granittisch aufgebaut, der sich in einem klimatisierten Raum befindet. Die Luft wird dort (laut Spezifikation) auf $\pm 0,3\,\mathsf{°C}$ und $\pm 5\%$ rel. Feuchte konstant gehalten. Mit einem Wärmeausdehnungskoeffizienten von etwa $25 \cdot 10^{-6}\,\mathsf{K}^{-1}$ bei Aluminium entspricht dies einer maximalen Längenänderung der Kammer von $\pm 30\,\mu\mathsf{m}$ in Längsrichtung und $\pm 9\,\mu\mathsf{m}$ quer dazu. Die Materialtemperatur schwankte im Laufe dieser Messung nur zwischen $21,97\,\mathsf{°C}$ und $22,00\,\mathsf{°C}$. Sie wird daher nicht weiter berücksichtigt.

Die Standardabweichung der meisten Rasniks liegt unter $1\,\mu\mathsf{m}$, nur die Rasniks, die diagonal verlaufen und vergrößern (*DA*), haben eine Standardabweichung von $3\,\mu\mathsf{m}$ bzw. $5\,\mu\mathsf{m}$. Es liegen aber alle Rasniks unter den geforderten $10\,\mu\mathsf{m}$ (siehe [ATL97]).

Der erste Aufbau des Spacers erreichte bei den beiden DA-Rasniks RMS-Werte bis $13\,\mu\mathsf{m}$. Zunächst war nicht klar, woher diese große Schwankung kam. So wurde eine Rasnik-Strecke mit gleicher Geometrie direkt auf dem Granittisch aufgebaut und mit den Rasniks im Spacer verglichen (siehe Abbildung 3.4). Der Aufbau zeigte, dass mit dieser Anordnung (Winkel der RasCam und RasLed ca. 13° gegen die optische Achse und $2:1$ Vergrößerung) eine Standardabweichung von knapp unter $5\,\mu\mathsf{m}$ erreicht werden kann. Es stellte sich dann bald heraus, dass es aufgrund von Produktionstoleranzen des Längsträgers zu Abschattungen des Sichtfeldes kam. Nachdem das Problem beseitigt war, erreichten alle Sensoren im Spacer ein σ von knapp unter $5\,\mu\mathsf{m}$.

Es ist damit gezeigt, dass die Rasniks in der BOG-Kammer eine Genauigkeit erreichen, die mindestens doppelt so gut ist, wie im *TDR* gefordert wird. Zu dieser geringen Unsicherheit trägt auch die genaue Kenntnis der Linsenbrennweite bei, die mit meinem Messverfahren bestimmt wird.

Als nächstes kommt die Frage auf, welche Aussagen die Sensordaten über die tatsächliche Form der Kammer machen können – doch darauf werde ich im nächsten Kapitel eingehen.

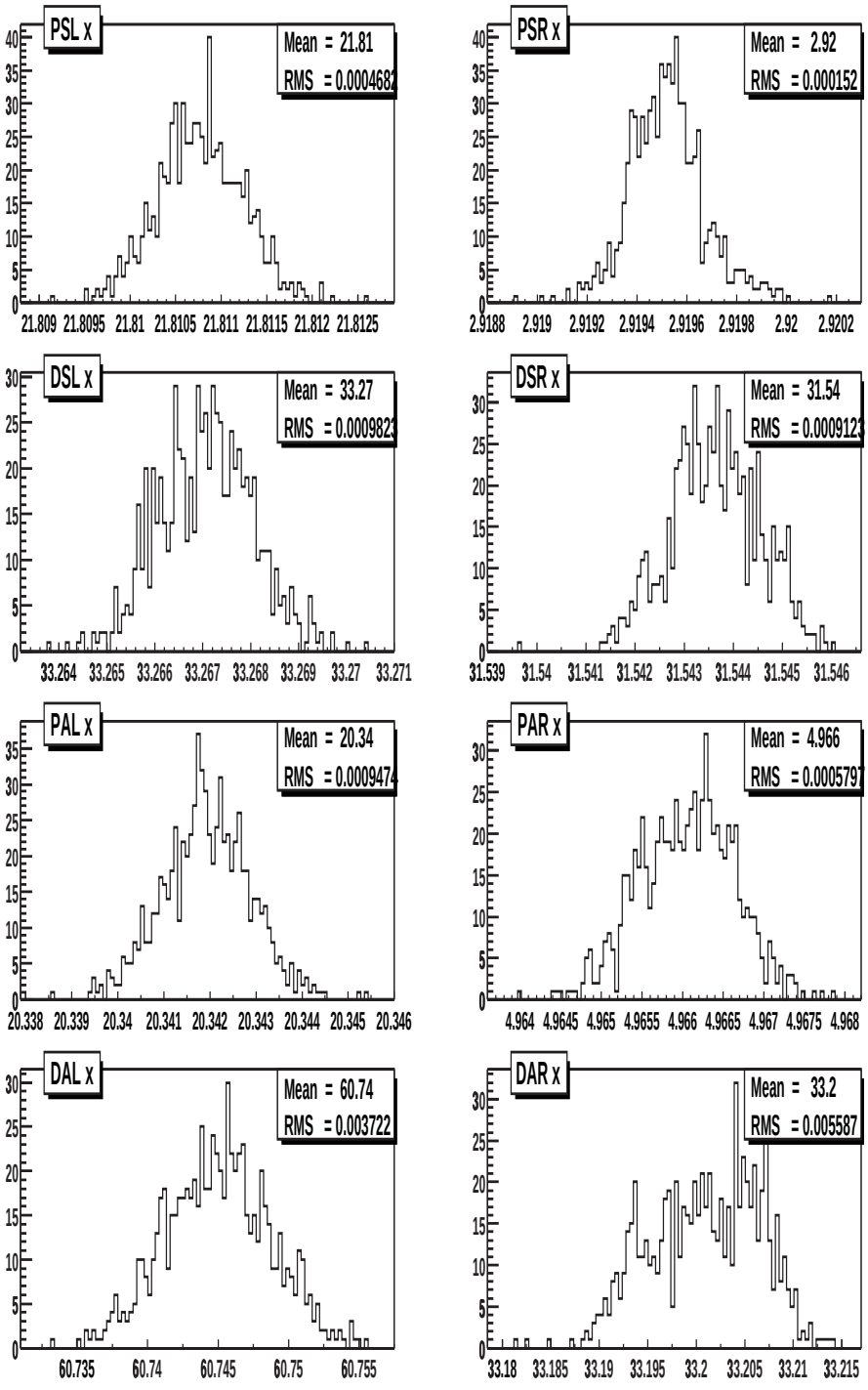


Abbildung 3.3: *Alle Rasniks der BOG während der Kammerrahmen flach auf einem Granittisch liegt. Die Daten wurden in einem Zeitraum von ca. 60 Std. genommen, mit einer Pause von 5 min. zwischen den Messungen. Wie man sieht, schwanken die* RMS*-Werte zwischen 0,15* μm *und 5,6* μm.

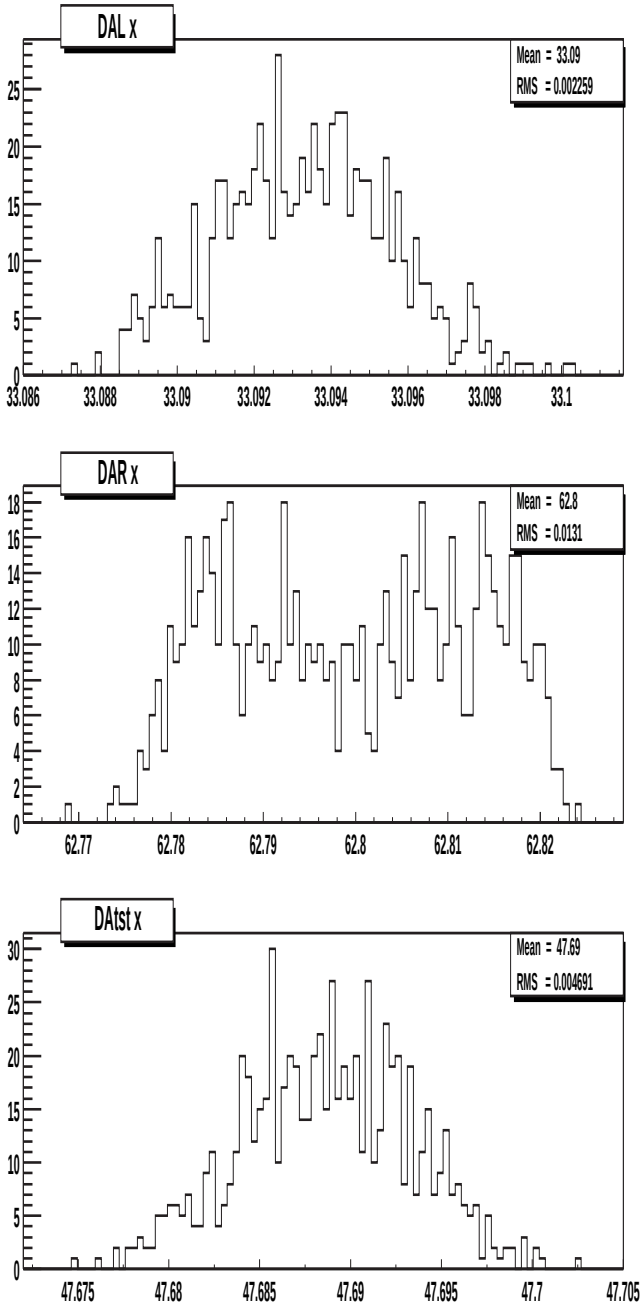


Abbildung 3.4: *Vergleich der beiden diagonalen asymmetrischen Rasniks (DAL - oben, DAR - Mitte) in der Kammer (mit Abschattungen durch die Longbeams) mit einem Rasnik auf dem Granittisch (unten). Alle Rasniks haben die gleiche Geometrie, aber deutlich unterschiedliche* RMS-*Werte.*

Kapitel 4

Bestimmung der Kammerdynamik

Nachdem nun der Betrieb der Rasnik-Sensoren unter Linux beschrieben wurde, widmen wir uns jetzt dem Thema der Kammerform bzw. ihrer Veränderung. Es soll geklärt werden, welche Verformungen durch das In-plane Alignment wahrgenommen werden können. Da alle Teile der Kammer fest miteinander verbunden sind, müssen sich die Quer- oder Längsträger verformen, damit es zu Bewegungen kommt. Gesucht sind Parameter, die diese Bewegungen beschreiben und durch das In-plane Alignment bestimmt werden können.

4.1 Untersuchte Parametrisierungen

Zunächst werde ich einige Begriffe klären, die im Zusammenhang mit der Simulation der Kammerdynamik stehen.

Für die erste Betrachtung der Kammer verwende ich ein (stark) vereinfachtes Modell, bei dem die Querträger (Cross Plates) als starr angenommen werden. Die Längsträger (Longbeams) halten in diesem Modell die Cross Plates auf ihrem Abstand zueinander, ansonsten sind die Cross Plates frei beweglich – natürlich nur um kleine Beträge. Als weitere Einschränkung für diese erste Betrachtung wurden nur die Rasnik-y-Koordinaten berücksichtigt. Das ist ebenso die y-Richtung im Kammerkoordinatensystem (vgl. [ATL97] und Abbildung 4.1), also senkrecht zur Röhrenebene. Sie ist bei weitem die interessanteste Koordinate, da sie senkrecht zur Ebene der Rasniks steht und somit auch Drehungen beschreiben kann.

Zur Erinnerung beschreibe ich hier den Aufbau des In-plane Alignment bei den BOG-Kammern. Es gibt insgesamt acht Rasnik-Strecken (siehe Abbildung 4.1), von denen vier ihre Linsen in dem mittleren Querträger haben, während die anderen vier Linsen in der „Spezial Cross Plate“ sitzen. Zu diesen acht Strecken gehören auch acht Masken, die alle in der ersten

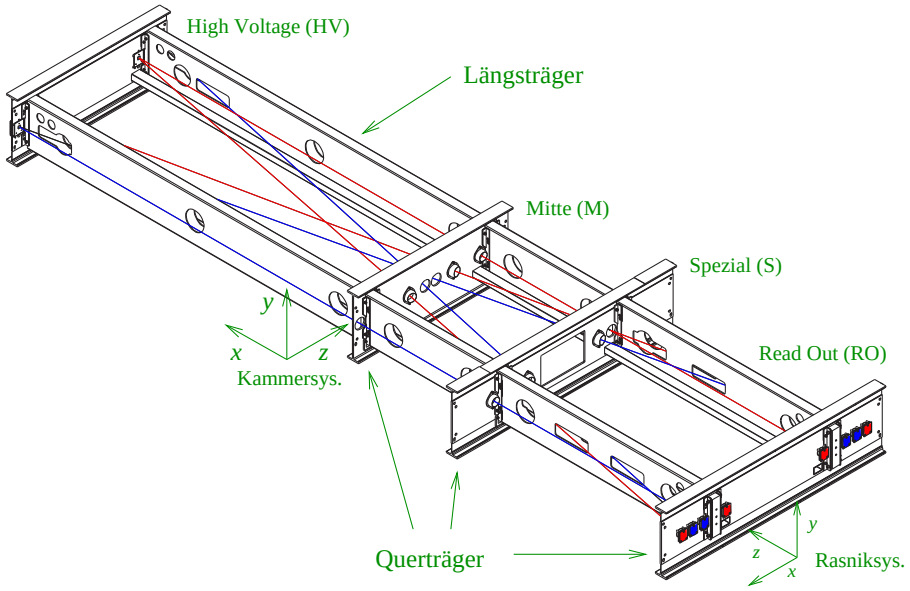


Abbildung 4.1: *Ein 3D-Modell des Rahmens der BOG-Kammern. Farblich hervorgehoben sind die Rasnik-Strecken. Zudem sind die Koordinatensysteme (der Kammer und der Rasniks) eingezeichnet.*

Cross Plate (Read Out) eingebaut sind. Bei den BOG-Kammern werden jedoch nur vier RasCams benötigt, da jede RasCam für zwei Rasnik-Strecken zuständig ist. Wie bereits erwähnt, gibt es pro innerem Querträger zwei parallele und zwei diagonale Rasniks. Jede RasCam nimmt das Bild jeweils von einer parallelen und einer diagonalen Strecke auf.

Die Koordinatensysteme der Rasniks sind um die (gemeinsame) y-Achse gedreht, so dass die z-Achse mit der optischen Achse zusammenfällt.

4.1.1 Ein Höhenmodell

Die BOG-Kammer hat mit ihren acht Rasniks auch acht y-Werte, die ich zu einem Vektor $\boldsymbol{y}_{\mathrm{Ras}}$ zusammenfassen möchte. Die Reihenfolge der Werte in diesem Vektor entspricht der Anordnung der Masken auf der Cross Plate mit steigender z-Koordinate (im Kammerkoordinatensystem).

Auf der anderen Seite der Simulation steht die Beschreibung der augenblicklichen Kammerform. Die Cross Plates werden, wie gesagt, als starre Körper angenommen, sodass die Angabe ihrer Eckpositionen, genauer der Endpunkte der Verbindungslinie der optischen Elemente, ihre Lage vollständig beschreibt. Zur vereinfachten mathematischen Beschreibung fasse ich auch diese Werte zu einem Vektor $\boldsymbol{y}_{\mathrm{CP}}$ zusammen. Es sind je zwei Werte pro Cross Plate (mit steigendem z) und dann die verschiedenen Cross Plates mit wachsendem x.

$$\boldsymbol{M} = 0.1 \begin{pmatrix} -4.58 & -0.42 & 0.00 & 0.00 & 6.42 & 3.58 & -0.92 & -4.08 \\ -6.04 & -0.90 & 6.60 & 3.40 & 0.00 & 0.00 & -0.13 & -2.94 \\ -5.69 & -1.24 & 8.21 & 1.79 & 0.00 & 0.00 & -2.94 & -0.13 \\ -3.60 & -1.40 & 0.00 & 0.00 & 8.15 & 1.85 & -4.08 & -0.92 \\ -1.40 & -3.60 & 0.00 & 0.00 & 1.85 & 8.15 & -0.92 & -4.08 \\ -1.24 & -5.69 & 1.79 & 8.21 & 0.00 & 0.00 & -0.13 & -2.94 \\ -0.90 & -6.04 & 3.40 & 6.60 & 0.00 & 0.00 & -2.94 & -0.13 \\ -0.42 & -4.58 & 0.00 & 0.00 & 3.58 & 6.42 & -4.08 & -0.92 \end{pmatrix}$$

Abbildung 4.2: *Mit Hilfe dieser Matrix werden die Rasnik-Werte bei bekannten Cross Plate Positionen berechnet.*

Die Darstellung als Vektoren hat nun den Vorteil, dass sich die Ruhelage der Kammer, also der „gerade“ Zustand, als Null schreiben lässt: $\boldsymbol{y}_{\mathrm{CP}} = \mathbf{0}$. Ebenso einfach lassen sich auch die „Nullwerte“ der Auslese abziehen:

$$\boldsymbol{y}_{\mathrm{Ras}} = \boldsymbol{y}_{\mathrm{Ras,readout}} - \boldsymbol{y}_{\mathrm{Ras,Null}}$$

Man kann noch einen Schritt weiter gehen, wenn man einen linearen Zusammenhang zwischen Bewegung und Rasnik annimmt, und eine Matrizengleichung aufstellen:

$$\boldsymbol{y}_{\mathrm{Ras}} = \boldsymbol{M} \cdot \boldsymbol{y}_{\mathrm{CP}} \tag{4.1}$$

Dann haben wir die Aufgabe, die Matrix $\boldsymbol{M}$ aufzustellen. Dazu schauen wir uns zunächst an, wie die Rasnik-Werte zustande kommen. Wir brauchen eine Gleichung, die uns die Beziehung zwischen den drei (y-)Positionen der optischen Komponenten und dem Auslesewert liefert:

$$y = y_{\mathrm{Linse}} - \frac{L_{\mathrm{Mask}}}{L_{\mathrm{Sys}}}\, y_{\mathrm{Cam}} - \frac{L_{\mathrm{Cam}}}{L_{\mathrm{Sys}}}\, y_{\mathrm{Mask}} - y_0 \tag{4.2}$$

Wir verwenden die Analyse hier so, dass sie die Position der Linse berechnet, daher geht die Linsenposition y_{Linse} ohne Vorfaktor in die Gleichung ein. Die Konstante y_0 beschreibt die Nulllage, die von der Maske abhängt. Sie wird nach dem Einbau bestimmt, indem man die Kammer, bzw. den Rahmen ausrichtet und die Rasniks ausliest. Hier gehen auch die verschiedenen Abstände Maske–Linse (L_{Mask}), Kamera–Linse (L_{Cam}) und Maske–Kamera ($L_{\mathrm{Sys}} = L_{\mathrm{Mask}} + L_{\mathrm{Cam}}$) ein. Diese geometrischen Konstanten werden im Bauplan der Kammer festgelegt und gelten wegen der Symmetrie jeweils für zwei Sensorstrecken. Dabei wird die Verschiebung der Masken vernachlässigt, inwieweit das gerechtfertigt ist, wird später noch untersucht.

Als nächstes sind die (y-)Positionen der optischen Komponenten selbst aus $\boldsymbol{y}_{\mathrm{CP}}$ zu bestimmen. Wir betrachten zuerst eine Cross Plate. Wir kennen

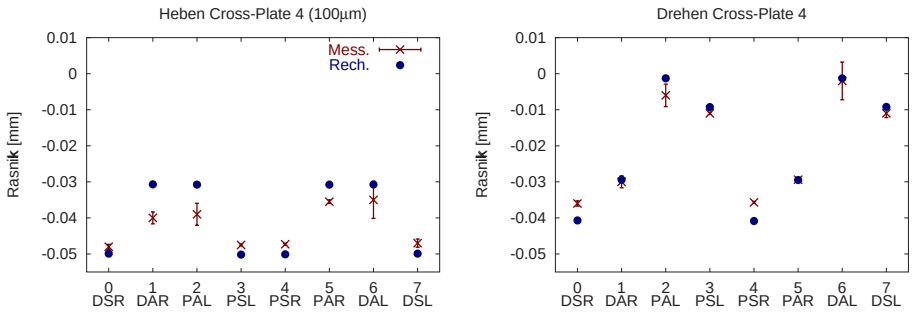


Abbildung 4.3: *Vergleich zwischen Rasnik-Messung und Simulation bei Anheben der HV Cross Plate um 100 µm (links) und einseitiges Anheben (Drehung) um 100 µm (rechts). Die Fehlerbalken entsprechen der Standardabweichung der Messwerte.*

die Höhen der beiden Enden der Cross Plate (y_1 und y_2) und ihre Länge L_{CP}. Dann wissen wir noch, wo sich auf der Cross Plate das optische Bauteil befindet (L_{opt}) und können jetzt seine Höhe angeben:

$$y_{\mathrm{opt}} = \frac{L_{\mathrm{opt}}}{L_{\mathrm{CP}}}\, y_1 + \frac{L_{\mathrm{CP}} - L_{\mathrm{opt}}}{L_{\mathrm{CP}}}\, y_2 \tag{4.3}$$

Hierbei steht y_1 für die Ecke mit der kleineren z-Komponente und L_{opt} für den Abstand von dieser, welcher wiederum aus dem Bauplan zu entnehmen ist.

Wie man an den letzten beiden Gleichungen sieht, ist der Bezug zwischen Cross Plates und Rasnik-Werten in der Tat linear. Es ist jetzt möglich, die Matrix aus Gleichung (4.1) zu bestimmen – wir müssen nur für jedes Rasnik Gleichung (4.2) anwenden und die darin auftretenden y-Werte nach Gleichung (4.3) für die entsprechende Cross Plate berechnen. Wir erhalten so eine 8×8-Matrix, die bei bekannten Cross Plate Lagen die Rasnik-Werte eindeutig bestimmt (siehe Abbildung 4.2). Diese Matrix ist für alle BOG-Kammern die Gleiche, sofern man die Verschiebung der Masken zur Korrektur der Linsenbrennweite vernachlässigt.

4.1.2 Messungen zum Höhenmodell

Die eben beschriebene Matrix wird zum Test der verschiedenen Simulationen benutzt und daher soll zunächst geprüft werden, ob sie denn die Kammer richtig beschreibt. Es wurden von mir einige Messungen durchgeführt, bei denen einzelne Cross Plates in definierter Weise gehoben wurden. In Abbildung 4.3 ist eine Messung gezeigt, bei der die vierte Cross Plate (High Voltage) um 100 µm angehoben wurde. Da wir die Linsenposition berechnen, erhalten wir dabei negative Werte! Neben den Messdaten (Kreuze) sind auch die über die Matrix $\boldsymbol{M}$ berechneten Werte eingetragen. Man erkennt

sofort, dass die Messdaten näher beieinander liegen als die Simulation. Das ist aber kein Fehler – in der Simulation wird ja angenommen, dass die beiden mittleren Cross Plates sich nicht bewegen, sie werden aber durch die Längsträger mit angehoben. Werden die Cross Plates in der Simulation entsprechend um $2\,\mu\mathsf{m}$ bzw. $5\,\mu\mathsf{m}$ verschoben, so liegen die Werte aufeinander. Diese Bewegungen können leider mit der Koordinaten-Messmaschine nicht aufgelöst werden.

Das zweite Bild in Abbildung 4.3 zeigt den Vergleich bei einer Rotation der vierten Cross Plate, wobei die Seite mit größerer z-Koordinate um $100\,\mu\mathsf{m}$ angehoben worden ist. Auch hier liegen die berechneten Werte sehr dicht bei den Messungen. Man kann also davon ausgehen, dass im Rahmen unserer Auflösung das Modell die BOG-Kammer gut beschreibt.

4.1.3 Interpretation der Rasnik-Daten

Für den Betrieb im ATLAS-Detektor wird naheliegenderweise die umgekehrte Richtung benötigt, d.h. von den Rasnik-Daten zur Kammerform. Ich habe für diesen Weg mehrere Ansätze verfolgt, von denen ich nun einige kurz skizzieren möchte. Die meisten sind zwar gescheitert, haben aber dennoch wertvolle Hinweise zur Lösung geliefert.

Man ist sofort versucht die Matrix $\boldsymbol{M}$ zu invertieren. Es fällt jedoch schnell auf, dass der Vektor $\boldsymbol{y}_{\mathrm{CP}}$ absolute Koordinaten im Raum enthält (bzgl. eines beliebigen aber festen Koordinatensystems). So beschreibt z.B. $\boldsymbol{y}_{\mathrm{CP}} = (1; 1; \ldots 1)$ eine um $1\,\mathsf{mm}$ nach oben verschobene Kammer, was die Rasniks aber unmöglich „wahrnehmen“ können. Die inverse Matrix, sofern sie existiert, kann die Kammer gar nicht richtig beschreiben. Ich habe daher die erste Cross Plate (Read Out) als fest angenommen und erhielt dann einen Vektor $\boldsymbol{y}_{\mathrm{CP}}$ mit 6 Komponenten. Die Frage ist jetzt, wie man diese mit den 8 Komponenten von $\boldsymbol{y}_{\mathrm{Ras}}$ verrechnet. Man kann eine 8×6-Matrix verwenden oder zwei Werte aus $\boldsymbol{y}_{\mathrm{Ras}}$ streichen. Im ersten Fall erhält man eine Überbestimmung, d.h. mehr Messwerte als Parameter. Diesen Fall behandelt man besser mit einem Fit-Algorithmus. Im zweiten Fall verschenkt man Messwerte, wofür sich die zwei ungenauesten Rasniks anbieten (bei uns sind das DAL und DAR) .

Oder man kann – was ich auch versucht habe – eine „virtuelle“ Cross Plate einführen, d.h. ich behandele sechs der acht Rasniks wie zuvor auf einer ortsfesten Cross Plate und setze zwei auf eine „virtuelle“, bewegliche Cross Plate. In diesem Modell können sich diese zwei RasCams gegen die anderen bewegen. Ich habe somit wieder acht Werte in $\boldsymbol{y}_{\mathrm{CP}}$, wobei die ersten beiden Komponenten nun diese „virtuelle“ Cross Plate bestimmen. Man erhält dadurch eine Möglichkeit, die Rechnung zu beurteilen. Je weiter die ersten beiden Komponenten von Null abweichen, desto schlechter ist die Rekonstruktion.

Es ist nun in der Tat so, dass die letzten beiden Verfahren in der Simulation die Kammer richtig berechnen. Mit Simulation bezeichne ich hier ein Verfahren, bei dem der Vektor $\boldsymbol{y}_{\mathrm{CP}}$ vorgegeben oder (zufällig) bestimmt wird, und dann, über die Matrix $\boldsymbol{M}$, Rasnik-Werte berechnet werden. Diese Werte werden anschließend mit der zu testenden Methode wieder in Cross Plate Positionen umgerechnet und mit dem ursprünglichen Vektor verglichen.

Die Versuche, die Kammerform mit den „echten“ Messwerten zu bestimmen, schlugen jedoch fehl. So liefert z. B. die Methode mit der virtuellen Cross Plate für die Messung in Abbildung 4.3 (links) eine Spacerform von:

$$\boldsymbol{y}_{\mathrm{CP}} = \begin{pmatrix} 0.00 & 0.00 & -0.05 & -0.05 & -0.07 & -0.07 & -0.04 & -0.04 \end{pmatrix}$$

Erwartet wird ein Vektor, bei dem die letzten beiden Komponenten (4. Cross Plate) einen Wert von ca. $0,1$ ($100\,\mu\mathsf{m}$) haben und alle anderen unter $0,01$ liegen. Die Diskrepanz zwischen Messung und Simulation ist auf die Messfehler zurückzuführen.

Diese Methoden sind numerisch instabil und genauere Untersuchungen zeigten, dass wegen der inversen Matrizen Differenzen sehr großer Zahlen auftreten. Daraus erklärt sich, wie aus den kleinen Unterschieden in den Rasnik-Werten große Unterschiede in den Cross Plate Positionen entstehen.

Aufgrund dieser Erkenntnis scheiden lineare Berechnungsmethoden zunächst aus. Daher untersuchte ich als nächstes Fit-Methoden, d.h. Verfahren, bei denen versucht wird, einen (möglichst guten) Vektor iterativ zu verbessern. Man hat hier einen Vektor $\boldsymbol{y}_{\mathrm{CP}}$ für die Cross Plates, mit dem man Rasnik-Werte berechnet. Diese werden dann mit den gemessenen Werten verglichen, und daraus wird eine neue Schätzung für $\boldsymbol{y}_{\mathrm{CP}}$ bestimmt. Zwei Probleme gilt es daher zu lösen:

1. einen guten Startvektor zu „raten“,
2. eine konvergierende Iteration zu finden.

Es wurden verschiedene achtdimensionale Intervallschachtelungen durchgeführt. Aber auch Iterationen über die Ecken einzeln, wobei jeweils die „schlechteste“ Ecke optimiert wurde. Man braucht dazu jeweils ein Maß, das angibt, wie gut der Vektor ist. Als Maße kamen in Frage:

1. Differenz zwischen berechnetem und gegebenem Rasnik,
2. mechanische Spannung der Longbeams, bei der rekonstruierten Form,
3. Durchhang und Verdrehung der Kammer,
4. Abweichungen zu einfachen Modellen (wie Polynome) der Longbeams.

Diese Methoden lieferten ganz unterschiedliche Ergebnisse. Anfänglich wurde der Suchraum eingeschränkt, sodass die Ruhelage eine Grenze markierte und das erwartete Ergebnis im Intervall lag. Es zeigten sich starke Abhängigkeiten von internen Parametern, wie vom Faktor, um den die Intervalle verkleinert wurden, oder von der Gewichtung der Maße. So ließen sich einige Verfahren auf bestimmte Verformungen, wie Heben oder Drehen einer Cross Plate, optimieren. Eine generelle Methode fand sich jedoch nicht. Außerdem scheiterten alle an einem um die Nulllage symmetrischen Suchraum. Die speziellen Optimierungen gaben aber Hinweise, dass unterschiedliche Bewegungen getrennt zu untersuchen sind.

4.2 Rotation und Translation

Es wird jetzt der Versuch, die Kammer als Ganzes zu beschreiben, zurückgestellt. Stattdessen betrachten wir „elementare“ Formveränderungen. Als erstes werden Rotationen der einzelnen Cross Plates um die Längsachse der Kammer untersucht. Ich nehme dabei an, dass die Schwerpunkte aller Cross Plates auf einer Geraden liegen und diese sich um die Gerade drehen. Die erste Cross Plate (RO) wird wieder als fest angenommen.

4.2.1 Auswertung der y-Werte

Zuerst ein paar qualitative Überlegungen, wie diese Rotationen in den Rasniks „kodiert“ sein könnten. Betrachten wir eine Cross Plate mit Linsen und drehen sie ein bisschen gegen den Uhrzeigersinn, so dass die beiden rechten Linsen nach oben gehen und die linken nach unten. Es liegt nun die Vermutung nahe, die Rotation aus der Differenz „rechts − links“ abzuleiten. Die gleiche Überlegung für die vierte Cross Plate (HV) führt zu etwas wie „parallel − diagonal“.

A priori ist allerdings nicht klar, ob sich die Bewegungen der beiden Cross Plates überhaupt voneinander unterscheiden lassen. Ich habe dazu Berechnungen durchgeführt, bei denen die Rasnik-Werte einmal gegen die Drehung der mittleren und einmal gegen die der vierten Cross Plate aufgetragen wurden. In Abbildung 4.4 ist deutlich zu sehen, dass die beiden Bewegungen eine unterschiedliche Signatur haben. Bei der High Voltage Cross Plate liegen die RasCams so dicht beieinander, dass die verschiedenen Rasniks praktisch die gleichen Werte liefern. Die Linsen haben deutlich größere Abstände voneinander, sodass sich hier unterschiedliche Abhängigkeiten ergeben. Man kann die Cross Plates bezüglich ihrer Rotation also deutlich unterscheiden.

Wenn man sich nun genau überlegt, wie die Winkel aus den Rasniks zu berechnen sind, so erhält man folgende Formel:

$$\sin\alpha_L = \frac{1}{\frac{L_\mathrm{p}}{m_\mathrm{p}\,C} - \frac{L_\mathrm{d}}{m_\mathrm{d}\,C}} \left(-\frac{y_\mathrm{pr} - y_\mathrm{pl}}{2m_\mathrm{p}\,C} - \frac{y_\mathrm{dr} - y_\mathrm{dl}}{2m_\mathrm{d}\,C} \right) \tag{4.4}$$

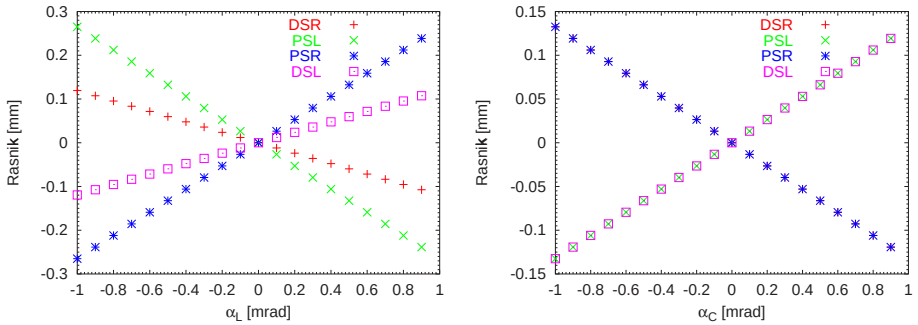


Abbildung 4.4: *Die beiden Diagramme zeigen die berechneten Rasnik-Werte der symmetrischen Rasniks, wenn man die mittlere (links) oder die High Voltage Cross Plate (rechts) dreht. Die Bewegungen weisen eine unterschiedliche Signatur auf.*

$$\sin\alpha_C = \frac{1}{\frac{m_\mathrm{p}\ C}{L_\mathrm{p}} - \frac{m_\mathrm{d}\ C}{L_\mathrm{d}}} \left(-\frac{y_\mathrm{pr} - y_\mathrm{pl}}{2L_\mathrm{p}} + \frac{y_\mathrm{dr} - y_\mathrm{dl}}{2L_\mathrm{d}} \right) \tag{4.5}$$

Besonders müssen die Vorzeichen vor den y-Werten beachtet werden. Zuvor will ich noch kurz die Variablen und Konstanten erklären, die sich alle aus der Geometrie bestimmen:

α_L Winkel der Linsen Cross Plate

α_C Winkel der Cross Plate mit den RasCams

$y_{..}$ y-Werte der Rasnik-Auslese

$L_\mathrm{p,d}$ Abstand der Linsen von der Drehachse (bzw. Schwerpunkt)

C Abstand der RasCams von der Drehachse (identisch für parallel und diagonal)

$m_\mathrm{p,d}$ relative Linsenposition des Rasniks $\left(-\frac{L_\mathrm{Mask}}{L_\mathrm{Sys}}\right)$

Der Index gibt an, ob es sich um ein paralleles oder diagonales Rasnik handelt, wobei eine Rechts-links-Symmetrie vorausgesetzt wird. Somit sind die Abstände gleich und die Rechnung vereinfacht sich erheblich.

Wenn man sich nun die Vorzeichen genauer anschaut und die Konstanten vernachlässigt, findet man diese Abhängigkeiten:

$$\sin\alpha_L \sim -(y_\mathrm{pr} - y_\mathrm{pl}) - (y_\mathrm{dr} - y_\mathrm{dl})$$

$$\sin\alpha_C \sim -(y_\mathrm{pr} - y_\mathrm{pl}) + (y_\mathrm{dr} - y_\mathrm{dl})$$

Man kann sich nun fragen, ob es noch weitere leicht zu interpretierende Kombinationen gibt. Die Translation des Schwerpunktes der Linsen Cross Plate wird durch die Summe aller y-Werte bestimmt:

$$\Delta y_L = \frac{1}{4} \ (y_{\mathrm{pr}} + y_{\mathrm{pl}} + y_{\mathrm{dr}} + y_{\mathrm{dl}}) \tag{4.6}$$

Da wir vier Werte messen, sollte man annehmen, dass es auch vier Resultate gibt – es fehlt folglich noch eins. Ein weiterer wünschenswerter Parameter ist sicherlich die Translation der High Voltage Cross Plate. Überlegen wir uns nun, wie sich diese Translation in den Rasniks widerspiegelt. Heben wir die Cross Plate um 2 mm an, so müssen wir das erst in die entsprechende Bewegung für die Linsen übersetzen, da die Analyse die Positionen der Linsen rekonstruiert. Nehmen wir an, dass die Linsen in der Mitte (der Kammer) sitzen, so senken sich diese scheinbar um 1 mm und zwar sowohl die parallelen als auch die diagonalen. Das bedeutet aber, dass eine Verschiebung der vierten Cross Plate nicht von einer Verschiebung der mittleren Cross Plate unterschieden werden kann. Somit kann der freie Parameter nicht die Verschiebung der High Voltage Cross Plate darstellen.

Man muss sich daher für *eine* Cross Plate entscheiden, die rekonstruiert werden soll, der Schwerpunkt der anderen wird damit als fest angenommen. Sinnvollerweise nimmt man die vierte Cross Plate als fest an, da ihr Schwerpunkt durch die Aufhängung der Kammer bestimmt ist, oder durch Sensoren zur Nachbarkammer (vlg. *Praxial System* in [ATL97]) gemessen werden kann. Wir nehmen im Folgenden den Schwerpunkt der vierten Cross Plate (High Voltage) als fest an.

So bleibt immer noch ein freier Parameter übrig. Wenn man sich die Positionen der Linsen auf einer der mittleren Cross Plates ansieht, so fällt auf, dass jeweils zwei recht mittig liegen und die anderen beiden weit außen sitzen. Daraus kann man ein Maß für die Durchbiegung dieser Cross Plates ableiten. Es fällt damit implizit die Annahme, dass die Cross Plates starr sind – sie „dürfen“ sich in der yz-Ebene durchbiegen. Qualitativ ist sie bestimmt durch die Differenz der Mittelwerte der parallelen und diagonalen Rasniks:

$$\delta y_L \sim (y_{\mathrm{pr}} + y_{\mathrm{pl}}) - (y_{\mathrm{dr}} + y_{\mathrm{dl}})$$

Die Größe δy_L beschreibt den Abstand der Verbindungslinien der Linsen von parallelen und diagonalen Rasniks. Die Verbindungslinien werden als parallel angenommen, was einer symmetrischen Durchbiegung entspricht. Für eine „gerade“ Cross Plate ist $\delta y_L = 0$.

Vielleicht ist δy_L eine unpassende Größe und man ist eher an der maximalen Abweichung von einer unverbogenen Cross Plate interessiert. Für eine symmetrische Durchbiegung entspricht das der Auslenkung in der Mitte, d.h. es muss die Cross Plate in der Mitte interpoliert werden, wofür allerdings

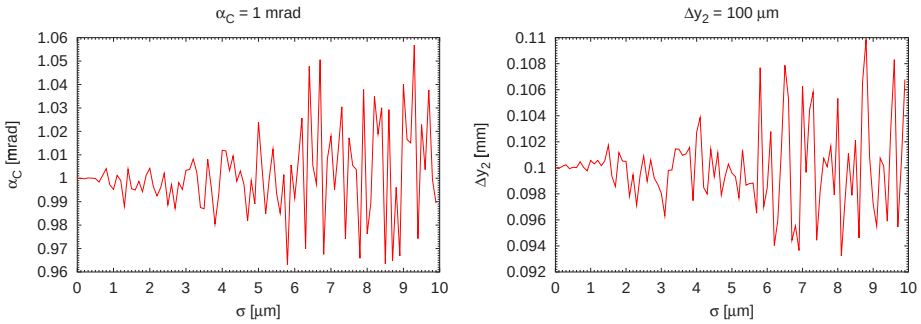


Abbildung 4.5: *Hier wurden exemplarisch die beiden Werte α_C und Δy_L für die zweite Cross Plate (Spezial) auf Ungenauigkeiten der Rasnik-Werte hin untersucht. Es wurde einmal die vierte Cross Plate um 1* mrad *gedreht und im anderen Plot die zweite Cross Plate um 100 µm angehoben. Die daraus berechneten Rasnik-Werte wurden dann mit zunehmenden gaußverteilten Unsicherheiten belegt und anschließend wieder die Größen α_C und Δy_L berechnet. Es zeigen sich keine überproportionalen Abhängigkeiten von den Messfehlern.*

ein Modell erforderlich ist. Wir brauchen also ein Modell, das symmetrisch ist und mit nur einem Parameter eine Krümmung beschreibt. Der einfachste Ansatz ist sicherlich ein Polynom 2. Grades. In Formeln:

$$y(z_c) = a\ z_c^2 + c$$

$$y'(0) = 0$$

Wobei z_c die z-Position relativ zur Mitte angibt. Die zweite Zeile folgt aus der Symmetrie. Man erhält dann für die Auslenkung in der Mitte der Cross Plate folgende Gleichung:

$$\varepsilon_{\mathrm{y}} = \frac{(y_{\mathrm{pr}} + y_{\mathrm{pl}}) - (y_{\mathrm{dr}} + y_{\mathrm{dl}})}{4} - a\ L_{\mathrm{p}}^2 \tag{4.7}$$

$$a = \frac{(y_{\mathrm{pr}} + y_{\mathrm{pl}}) - (y_{\mathrm{dr}} + y_{\mathrm{dl}})}{2\ (L_{\mathrm{p}}^2 - L_{\mathrm{d}}^2)}$$

Die Gleichungen 4.4, 4.5, 4.6 und 4.7 beschreiben zusammen vier geometrische Parameter, die aus vier Rasnik-Werten *analytisch* berechnet werden. Die BOG-Kammern haben zwei Cross Plates mit Linsen, sodass die zugehörigen Parameter (α_{L}, Δy_{L} und ε_{y}) zweimal zu berechnen sind. (Zur Unterscheidung werde ich die Variablen auch mit der Nummer der Cross Plate indizieren, z. B. α_3 für den Winkel der dritten Cross Plate). Einzig α_C ist doppelt bestimmt. Wie wir noch sehen werden, wird α_C aus den asymmetrischen Rasniks deutlich schlechter (ungenauer) berechnet als aus den symmetrischen, so dass wir diesen Wert einfach verwerfen.

Nach den Misserfolgen mit den Matrizen-Gleichungen soll nun auch hier geprüft werden, wie empfindlich diese Größen auf Messfehler reagieren. Ich habe dafür wieder numerische Untersuchungen durchgeführt. Dieses Mal wird die Kammerform vorgegeben. Dann werden daraus die Rasnik-Daten berechnet und anschließend mit einem zufälligen (normalverteilten) Fehler „verschmiert“. Aus den so berechneten Daten wird dann die Kammerform rekonstruiert. Einige Ergebnisse sind in Abbildung 4.5 zu sehen. Man sieht gut, dass die Ergebnisse nicht überproportional mit der Unsicherheit σ steigen. Aus diesen Simulationen liest man bei einer Unsicherheit von $5\,\mu\mathsf{m}$ pro Rasnik eine Ungenauigkeit für $\alpha_{\,\mathrm{c}}$ von $20\,\mu\mathsf{rad}$ und für Δy_2 von $5\,\mu\mathsf{m}$ ab. Die bei unserer Kammer tatsächlich erreichbare Genauigkeit wird im nächsten Abschnitt untersucht. Zunächst soll noch mal festgehalten werden, dass wir die maximal mögliche Anzahl an Ergebnissen (acht) aus den ebenfalls acht Rasnik y-Werten bestimmt haben.

4.2.2 Die z-Richtung

Schauen wir uns nun die Rasnik x-Werte (z-Richtung im Kammersystem) an und überlegen wir uns, welche Information darin „kodiert“ ist. Zu Winkelmessungen eignen sie sich nicht, da ihre Messrichtung parallel zu ihrer Verbindungsachse liegt.

Analog zu den y-Werten kann auch hier, aus dem arithmetischen Mittel, die Position des Schwerpunktes der jeweiligen Linsen Cross Plate bestimmt werden. Sie ist ebenfalls ununterscheidbar von einer Bewegung der vierten Cross Plate, weshalb diese auch in der z-Richtung als fest angenommen wird. Für die Bewegung des mittleren Querträgers erhält man:

$$\Delta z_{\mathrm{L}} = \frac{1}{4}\ (z_{\mathrm{pr}} + z_{\mathrm{pl}} + z_{\mathrm{dr}} + z_{\mathrm{dl}}) \tag{4.8}$$

Alle anderen Kombinationen der $z_{\,\cdot\cdot}$ (x-Werte der Rasniks) lassen sich, unter der Annahme einer starren Cross Plate, nicht interpretieren. Es handelt sich also auch hier um dynamische Größen einer einzelnen Cross Plate. Wir können somit die Änderung des Abstandes der optischen Komponenten messen, d.h. eine Verlängerung der Cross Plates. Solche Änderungen könnten durch Temperatureffekte hervorgerufen werden. Bei $530\,\mathsf{mm}$ Aluminium, dem Abstand der parallelen symmetrischen Rasniks, entspricht das ca. $14\,\mu\mathsf{m}\ \mathsf{K}^{-1}$.

In Abbildung 4.6 sind (wieder für die symmetrischen Rasniks) die Abhängigkeiten von den Temperaturen der dritten und vierten Cross Plate aufgetragen. Auch hier gilt, dass wir keine absoluten Größen – hier Temperaturen – messen können, sondern nur Unterschiede zwischen den Cross Plates. Analog zur Auswertung der y-Koordinaten soll wieder die erste Cross Plate (Read Out) als Referenz genommen werden, das heißt, die Temperaturwerte geben den Unterschied zur ersten Cross Plate an.

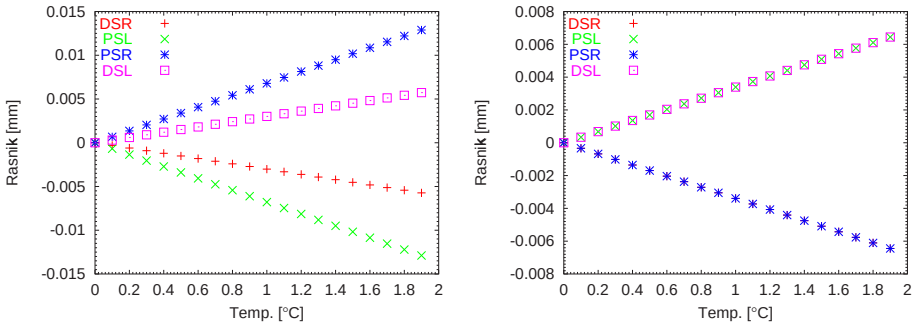


Abbildung 4.6: *Die zwei Graphiken zeigen die berechneten Rasnik-Werte der symmetrischen Rasniks, wenn man die mittlere (links) oder die High Voltage Cross Plate (rechts) erwärmt. Beide Fälle lassen sich deutlich voneinander unterscheiden. Diese Berechnungen wurden mit einem Wärmeausdehnungs-Koeffizienten von* $25,6 \cdot 10^{-6}\,\mathrm{K}^{-1}$ *durchgeführt ([Kam95]).*

Um die Berechnung zu vereinfachen, bietet es sich an, nicht die Temperatur selbst, sondern die Streckung oder Stauchung γ anzugeben. Sie ist wie folgt definiert:

$$L = (1+\gamma)\, L_0$$

Hierbei ist L_0 die Ursprungslänge der Cross Plate und L die momentane Länge. Für eine Linsen Cross Plate berechnet man die Streckung mit:

$$\gamma_{\mathrm{L}} = \frac{z_{\mathrm{pr}} - z_{\mathrm{pl}} + z_{\mathrm{dr}} - z_{\mathrm{dl}}}{2\,(L_{\mathrm{p}} + L_{\mathrm{d}})} \tag{4.9}$$

Die $z_{..}$ sind die entsprechenden Werte aus den verschiedenen Rasniks. Man kann nun auch diese Größe für die vierte Cross Plate bestimmen:

$$\gamma_{\mathrm{C}} = \frac{\frac{1}{2}(z_{\mathrm{pr}} - z_{\mathrm{pl}}) - \gamma_{\mathrm{L}}\, L_{\mathrm{p}}}{-C\ \frac{L_{\mathrm{Mask}}}{L_{\mathrm{Sys}}}} \tag{4.10}$$

Diese Formeln sind nicht ganz exakt, da hier die jeweiligen Brüche $\frac{L_{\mathrm{Mask}}}{L_{\mathrm{Sys}}}$ für parallel und diagonal gleichgesetzt wurden. Die Unterschiede sind bei den BOG-Kammern jedoch in der Größenordnung von 10^{-4}.

Bevor wir nun weiter über Temperaturmessungen mit Rasniks reden, sollten wir erst einmal klären, inwieweit das überhaupt Sinn macht. Zum einen werden ja spezielle Temperatursensoren auf den Kammern eingesetzt und zum anderen sollten wir die Messpunkte und die erreichbare Genauigkeit beachten. Als Messpunkte können die Rasniks nur einen Wert pro Cross Plate liefern, zudem auch nur relativ zur ersten Cross Plate. Die Temperatursensoren hingegen liefern absolute Werte. Es werden auch mehr als drei Sensoren pro Kammer sein, die mit einer Präzision von $0.2\,\mathrm{K}$ messen ([vE99]). Für die Rasniks berechnen wir die Genauigkeit mit Hilfe des

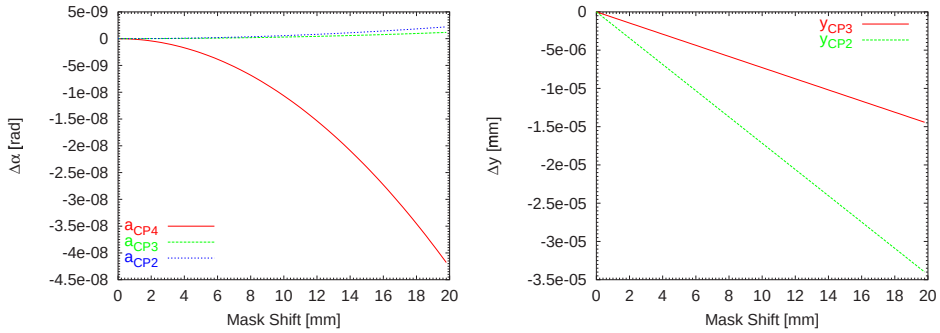


Abbildung 4.7: *In diesen Bildern ist dargestellt, wie die rekonstruierten Rotationen (links) und Translationen (rechts) von der Maskenverschiebung abhängen. Bei diesen Berechnungen wird eine maximale Links-rechts-Asymmetrie angenommen. Die Verschiebung ändert die Ergebnisse um weniger als 0,1 μm bzw. 0,1 μrad.*

kleinsten Abstandes (zwischen den Linsen) und der Ortsgenauigkeit von 5 μm. Der kleinste Linsenabstand beträgt 24 cm bei der mittleren Cross Plate. Bei einem Wärmeausdehnungskoeffizienten von $25 \cdot 10^{-6}\,\mathrm{K}^{-1}$ ([Kam95]) entspricht das $6\,\mu\mathrm{m\ K}^{-1}$. Mit den Rasniks erreicht man also ungefähr 1 K Temperaturauflösung.

Aufgrund dieses Ergebnisses sollten wir die Temperaturmessung den dafür vorgesehenen Sensoren überlassen. Wir werten von den Rasnik x-Werten also nur das arithmetische Mittel aus, um die Verschiebung der Cross Plate in z-Richtung zu bestimmen. Die anderen Größen kann man noch für Konsistenzüberprüfungen benutzen, z.B. ob die Analyse stabil läuft, oder ob sich optische Komponenten verschoben haben.

4.2.3 Einfluss der Maskenverschiebung

Bei den Berechnungen in den vorangegangenen Abschnitten wurde eine Links-rechts-Symmetrie der Rasniks angenommen. Diese Symmetrie wird jedoch durch die Verschiebung der Masken gebrochen. Diese Verschiebung ist nötig, um die Abweichungen der Linsen von ihren Sollbrennweiten zu kompensieren. Es wurde daher überprüft, ob die Maskenverschiebung einen störenden Einfluss auf die Bestimmung der Parameter hat.

Für die Überprüfung wurde eine maximale Links-rechts-Asymmetrie unterstellt, d.h. ich habe die linken Masken in positive x-Richtung verschoben und die rechten in negative x-Richtung. Die Länge der Verschiebung wurde dabei zwischen 0 mm und 20 mm variiert. Damit ist der gesamte Bereich abgedeckt, der bei den BOG-Kammern erwartet wird. Die Abhängigkeiten wurden wie folgt berechnet. Zuerst werden für eine feste Kammerform die Rasnik-Werte generiert, wobei die exakten Positionen der Masken berück-

sichtigt werden. Dann werden die Maskenverschiebungen auf Null gesetzt und die Verformungen aus den gewonnenen Rasnik-Werten rekonstruiert. Anschließend werden die berechneten Translationen und Rotationen von den vorgegebenen Werten subtrahiert. Man erhält somit den Fehler, den die Rekonstruktion macht, wenn sie die Maskenverschiebung nicht berücksichtigt.

In Abbildung 4.7 sind die Simulationsergebnisse für den Fall gezeigt, dass die High Voltage Cross Plate um 1 mrad gedreht wird. Bei allen Simulationen bleiben die Abweichungen unter 0,1 μm bzw. 0,1 μrad. Sie sind daher im Vergleich mit den erreichten Genauigkeiten vernachlässigbar. Es können folglich die vereinfachten Formeln verwendet werden, die die Symmetrie voraussetzen.

4.2.4 Die weiteren Rasnik-Werte

Die Analyse eines Rasnik-Bildes liefert mehr Information als nur die beiden x, y-Koordinaten. So wird auch die Vergrößerung des Bildes bestimmt, indem die Maskengröße mit dem Bild auf den Pixeln verglichen wird. Über die Vergrößerung kann nun die Position der Linse auf der Strahlachse bestimmt werden. Auch hier gibt es wieder zwei Möglichkeiten – je nachdem, ob die Linse oder die Maske berechnet wird. Diese Bezeichnungen sind hier jedoch etwas unglücklich. Betrachten wir das Problem einmal genauer. Wir haben die drei Abstände Linse - Maske (L_{Mask}) und Linse - Kamera (L_{Cam}) sowie die Gesamtlänge ($L_{\mathrm{Sys}} = L_{\mathrm{Mask}} + L_{\mathrm{Cam}}$). Das sind zwei Freiheitsgrade, wovon einer durch die Vergrößerung (M) bestimmt werden kann , der andere muss als konstant angenommen werden. Wenn wir die Linse berechnen, so werden die Maske und die Kamera als fest betrachtet, d.h. $L_{\mathrm{Sys}} = \mathrm{konst.}$ Für die Verschiebung der Linse aus der Mitte ergibt sich dann:

$$\Delta x_{\mathrm{L}} = \frac{1-M}{2(1+M)} L_{\mathrm{Sys}} \tag{4.11}$$

Falls die Analyse die Maskenposition rekonstruiert, wird L_{Cam} als konstant angenommen. Es ändert sich dann L_{Mask} und somit auch die Gesamtlänge. Diese Annahmen über die Bewegung sind analog zu den x- und y-Werten.

Wir berechnen, wie schon gesagt, die Linsenposition, und wir sollten nun überlegen, welche Bewegungen zu einer Verschiebung der Linse führen. Eine Änderung der Kammerlänge, wie beispielsweise bei einer Temperaturerhöhung, ist nicht sichtbar, da die Linse weiterhin in der Mitte zwischen Maske und Kamera sitzt. Nur eine Änderung auf einer Kammerseite wäre erkennbar. Eine weitere Möglichkeit, die x-Richtung zu ändern, ist eine Scherung der Kammer, also das Schieben einer Cross Plate in z-Richtung. Wenn sich die mittleren Cross Plates linear mitbewegen, so ist auch diese Bewegung „unsichtbar“. Andernfalls wird man eine solche Veränderung besser mit den z-Werten messen können. Es bleibt eigentlich nur noch eine Verschiebung der Cross Plate relativ zum Rest der Kammer. Dazu muss aber die

Verbindung zwischen Cross Plate und Longbeam gelöst sein. Wenn man also eine Änderung der x-Koordinate misst, so heißt das, dass die Kammer nicht richtig verschraubt[1] bzw. geklebt ist. Es stellt sich wieder einmal die Frage nach der erreichbaren Genauigkeit dieser Koordinate. Testmessungen auf der optischen Bank ergaben für die z-Koordinate eine Standardabweichung von ca. $250\,\mu\mathsf{m}$. Dieser extrem große Wert führt dazu, dass diese Werte nicht weiter ausgewertet werden können. Man kann höchstens feststellen, dass die Aufhängung der Cross Plate nicht in Ordnung ist.

Kommen wir nun zu den letzten potentiellen Werten, der Rotation um die optische Achse. Das System kann natürlich eine Drehung der Linse nicht feststellen, bleibt also nur die Drehung der vierten Cross Plate relativ zur ersten Cross Plate. Sofern die Cross Plate starr ist, sollten alle Rasniks den gleichen Winkel messen. Das ist der gleiche Winkel $\alpha_{\,\mathrm{C}}$, den wir schon aus den y-Werten berechnet haben. Wie im folgenden Abschnitt gezeigt wird, ist die Berechnung über die y-Werte deutlich genauer als die Winkel aller Rasniks zusammen (vgl. Abbildung 4.13). Wir können also auch diese Werte, ohne Information zu verlieren, einfach außer Acht lassen.

4.3 Überprüfung der Parameter

In diesem Abschnitt werden einige von mir durchgeführte Messungen vorgestellt. Diese Messungen sollen zeigen, dass die beschriebene Parametrisierung der Kammerform zu guten Ergebnissen führt.

Zunächst will ich beschreiben, wie diese Messungen abliefen. Das Problem ist, dass ich zum Prüfen der Rasniks Messungen zum Vergleichen brauche. Zu diesem Zweck wurde der erste Kammerrahmen in einem klimatisierten Raum aufgebaut. In diesem Raum ist eine *Koordinatenmessmaschine* (*CMM*) installiert. Mit dieser Maschine können Objekte – hier vor allem Kugeln – sehr genau gemessen werden, d.h. ihre Positionen im Raum bestimmt werden. Kugeln eignen sich hier besonders gut, da ihre Positionen unabhängig sind, von irgendwelchen Winkeln, wie z.B. die Antastrichtung der Messmaschine. Und man kann sie von vielen Richtungen antasten, was die Fehler der Messung klein hält. Es werden auch Flächen angetastet; so werden beispielsweise die Stirnseiten der Cross Plates ausgerichtet, aber eben mit größerem Fehler. Die erreichbare Präzision liegt zwischen $1{,}5\,\mu\mathsf{m}$ und $10\,\mu\mathsf{m}$, je nachdem wie groß die zu bestimmenden Distanzen sind. Um die Kammerform reproduzierbar bestimmen zu können, habe ich auf alle Cross Plates mehrere Kugeln aus gehärtetem Stahl ($\phi = 10\,\mathsf{mm}$) aufgeklebt. Diese Kugeln werden dann von der Messmaschine ca. 10 mal angetastet und ihre Positionen gemessen. Das ist weitaus genauer als die Cross Plates selber anzutasten, weil diese eine recht raue Oberfläche haben. In Abbildung 4.8 ist

[1] Die mittleren Cross Plates können in y-Richtung verschoben werden, um den gravitativen Durchhang der Kammer den Drähten anzupassen.

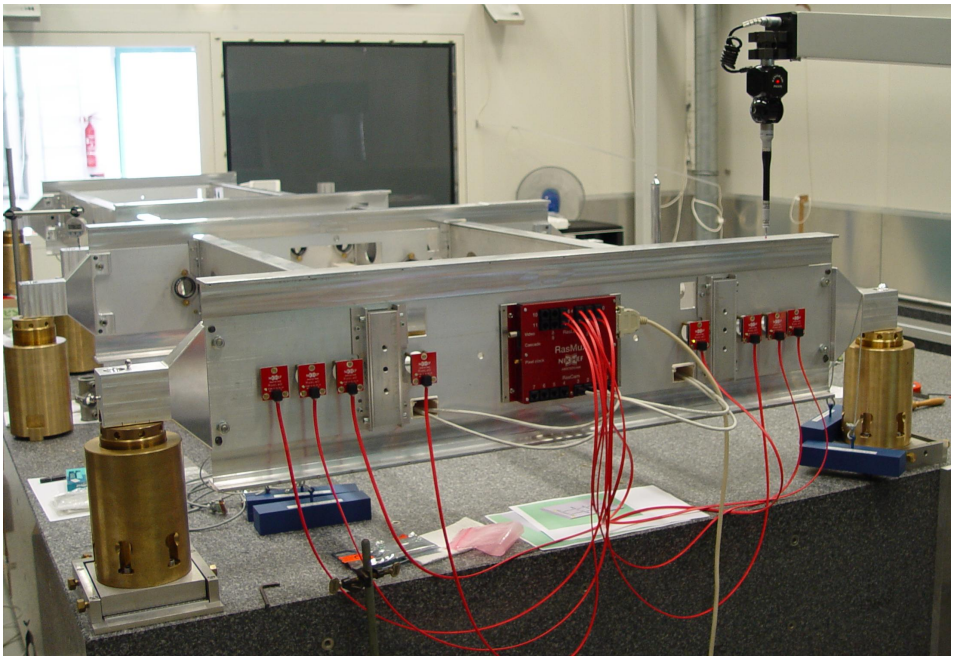


Abbildung 4.8: *Dieses Photo zeigt den Spacer im klimatisierten Raum bei einer Messung der Koordinatenmessmaschine.*

ein Bild dieses Aufbaus zu sehen. Die Kammer liegt auf höhenverstellbaren Füßen, sodass es leicht möglich ist, die Enden jeder Cross Plate zu heben oder zu senken.

Ein weiteres Messsystem wurde zum Vergleich von mir herangezogen. Ich habe drei sehr präzise elektronische Neigungssensoren (Leica, *NIVEL20*) auslesen können. Diese 2-dimensionalen „Wasserwaagen“ befanden sich an der ersten, dritten und vierten Cross Plate und waren so in der Lage, die Winkel α_4 und α_3 zu messen.

Die Auslese-Software wurde für diese Testreihen von mir so erweitert, dass die aktuellen Daten der Messmaschine, der Wasserwaagen und die Umgebungsparameter (Temperaturen, Luftfeuchte, etc.) automatisch mit aufgezeichnet werden. Man kann sogar die Rasnik-Auslese auf diese Datenquellen synchronisieren. Auf diese Weise ist der zeitliche Bezug der unterschiedlichen Systeme sichergestellt und man kann zuverlässig Korrelationen (z. B. mit der Temperatur) erkennen. Die Berechnung der Winkel und Verschiebungen aus den Rasnik-Daten wurde ebenfalls in die Auslese-Software integriert. In den folgenden Diagrammen sind diese Rechenergebnisse mit „BOG“ bezeichnet. Die Beschreibung der Kammer und der Rasniks sowie die Berechnungen wurden von mir in mehrere C++-Klassen verlagert. Die Berechnung der Form findet in der Klasse `RasShape` statt, deren Deklaration in Abbildung 4.9 zu sehen ist. Sie greift auf die Klasse `Rasnik` zurück,

```
class RasShape
{
  Rasnik RP;                  // rasnik geometry; parallel
  RasVal PL,PR;               // rasnik vals
  Rasnik RD;                  // rasnik geometry; diagonal
  RasVal DL,DR;               // rasnik vals

public:
  RasShape(Rasnik P,Rasnik D,
           const char *nPR = 0x0,const char *nPL = 0x0,
           const char *nDR = 0x0,const char *nDL = 0x0);

  int set_Par (Rasnik r);     // set rasnik geom
  int set_Diag(Rasnik r);     // set rasnik geom
  int set_ValP(RasVal r,RasVal l); // set values
  int set_ValD(RasVal r,RasVal l); // set values

  RasVal & pl();              // get ref. to LP
  RasVal & pr();              // get ref. to RP
  RasVal & dl();              // get ref. to LD
  RasVal & dr();              // get ref. to RD

  double CpLensY();           // get fit values
  double CpLensZ();
  double CpLensAx();
  double CpCamAx();

  friend ostream& operator<<(ostream& outs,RasShape &Shap);
};
```

Abbildung 4.9: *Die Methoden dieser Klasse berechnen die Winkel und Verschiebungen aus den Rasnik-Daten.*

die die Geometrie einer Strecke beschreibt. Die Klasse `RasVal` dient der Verwaltung der Rasnik-Werte, d.h. sie kann die Null-Positionen abziehen oder mehrere Daten mitteln.

Wie zuvor diskutiert, dient die erste Cross Plate als Referenz, sie wird folglich nicht berechnet. Es werden zuerst die Messungen am vierten und dritten Querträger vorgestellt, d.h. die Cross Plates, die die langen Rohre tragen. Die Rekonstruktion dieser Cross Plates entspricht auch der Situation bei den rechteckigen Kammern im Myonspektrometer.

4.3.1 Messungen zur HV Cross Plate

Die erste Messung lief nun so ab, dass ich die vierte Cross Plate an einer Seite gesenkt und gehoben habe. Das ändert den Winkel zur ersten Cross Plate, was die Rekonstruktion dann berechnen sollte (siehe Abbildung 4.10).

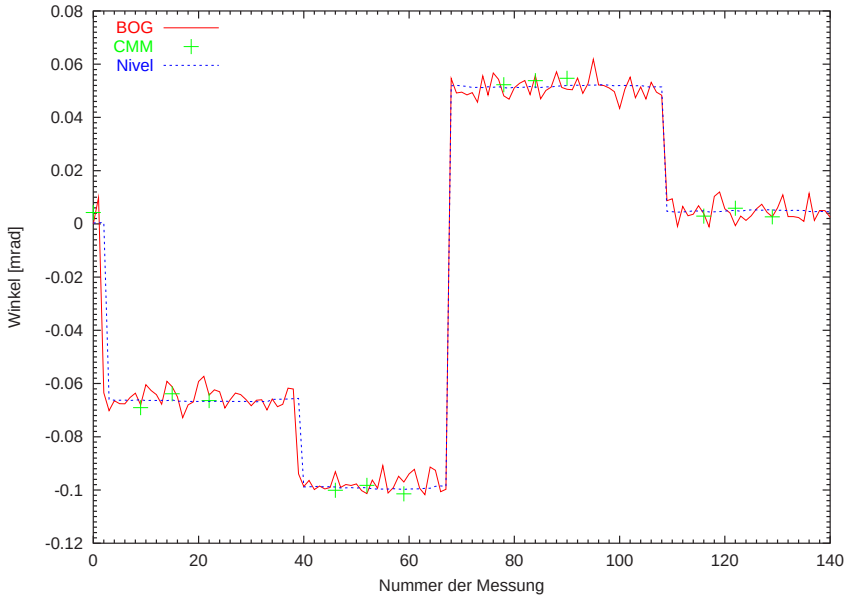


Abbildung 4.10: *In dieser Messreihe wurde der Winkel α_4 aus den Rasnik-Werten berechnet (durchgezogene Linie) und synchron dazu die Wasserwaagen ausgelesen (gestrichelte Linie). Diese Messungen erfolgten ca. alle 90* s. *Die Kreuze markieren die Daten der Messmaschine zu den entsprechenden Zeiten.*

Die beschriebenen Neigungssensoren können diesen Winkel direkt messen, indem man einfach die Werte der vierten und ersten Cross Plate voneinander subtrahiert. Bei den Daten der Messmaschine ist die Sache komplizierter. Es muss aus den gemessenen Kugeln erst die Lage der jeweiligen Cross Plate bestimmt werden, um dann die Winkel zwischen ihnen zu berechnen. Die Kugeln sind etwa auf der gleichen Höhe (y-Koordinate) wie die optischen Elemente. Den Winkel zwischen den Cross Plates berechne ich, indem ich den Verbindungsvektor zweier Kugeln bestimme und dann den Winkel zwischen diesen Vektoren messe. Die Vektoren werden in einem zum Kammersystem parallelen Koordinatensystem gemessen, sodass sich der Winkel in einem 2D-Raum ermitteln lässt.

Schauen wir uns Abbildung 4.10 genauer an. Aus den „konstanten“ Bereichen kann man ablesen, wie stabil die Rekonstruktion den Winkel bestimmt. Ich habe den Wert zu $\sigma = 0,0035\,\mathrm{mrad}$ berechnet. An den Enden der Cross Plate entspricht dies einem Höhenunterschied von $1{,}47\,\mu\mathrm{m}$. Als nächstes vergleichen wir die rekonstruierten Werte mit den Messpunkten der Messmaschine. Ich ermittle dazu von den berechneten Winkeln in den Bereichen, in denen sie konstant sein sollen, Mittelwert $\bar{\alpha}$ und Varianz σ^2. Dann summiere ich die quadrierten Differenzen von Mittel- und Messmaschinenwerten, dividiert durch die Varianz. Aus den Daten berechne ich ein

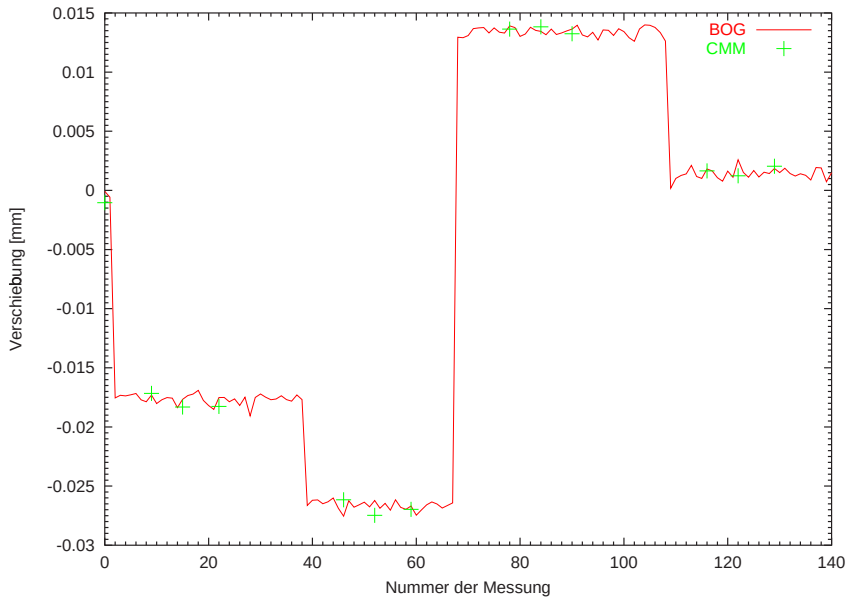


Abbildung 4.11: *Es werden hier die berechneten Werte für ΔY_3 bei Verschieben der vierten Cross Plate gezeigt.*

$\chi^2 = 5,8$ bei insgesamt 12 Punkten. Das ist ein sehr kleiner Wert, und das obwohl ich die Fehler der Messmaschine nicht berücksichtigt habe. Ihr Fehler ist auch sehr gering, da es hier im Wesentlichen nur auf die Reproduzierbarkeit ankommt und keine großen Strecken gemessen werden.

Für einen weiteren Vergleich können die installierten Wasserwaagen dienen. Ihre Messmethode ist recht träge und ihre Auslese weist eine hohe Auflösung auf. Die Werte sind sehr stabil und haben ein geringes Rauschen. Ich habe bei jeder Rasnik-Auslese auch die Wasserwaagen mit ausgelesen, damit zu jedem Fit-Wert eine Referenz existiert. Der Vergleich dieser Daten liefert ein χ^2 von 128 bei 119 Messpunkten.

Wir können daraus schließen, dass mit Hilfe der Rasniks und meines Algorithmus der Winkel der vierten Cross Plate auf etwa 3,5 μrad genau bestimmt werden kann.

Durch das Heben einer Seite wird auch der Schwerpunkt mit angehoben. Der Schwerpunkt der vierten Cross Plate wird aber als Referenz für die Längsachse der Kammer verwendet, sodass folglich die beiden mittleren Cross Plates nach unten „wandern“. Dieser Effekt ist gut in Abbildung 4.11 zu sehen, wo der rekonstruierte Wert ΔY_3 aufgetragen ist, aber in Wirklichkeit die vierte Cross Plate bewegt wurde. Es zeigt sich auch hier, dass die rekonstruierte Höhe perfekt mit den Werten der Messmaschine zusammen passt. Auf die dritte Cross Plate wird in einem separaten Abschnitt noch genauer eingegangen.

Bevor wir mit der vierten Cross Plate abschließen, soll noch auf die Rasnik-Winkel eingegangen werden. Wie bereits erwähnt, liefert die Analyse eines Bildes auch ein Maß für die Drehung. Diese Drehung lässt sich als Winkel zwischen erster und vierter Cross Plate interpretieren. In Abbildung 4.12 ist zum Vergleich zusätzlich der Winkel von der Strecke „PSL“ eingetragen. Auch diese Kurve wurde so verschoben, dass sie auf den anderen liegt. Die Rasnik-Winkel weisen eine Standardabweichung von $\sigma = 0,038\,\mathsf{mrad}$ auf. Das ist eine Größenordnung schlechter als die durch meinen Algorithmus berechneten Werte. Mit diesem σ berechnet man $\chi^2 = 131$ bei 122 Messpunkten. Das heißt, die Messungen passen zum angegebenen Fehler, aber die Werte sind doch zu ungenau.

Ich habe als nächstes untersucht, ob sich die Rasnik-Winkel verbessern lassen, indem man mehrere Werte mittelt. Das Ergebnis für die symmetrischen vier Rasniks ist in Abbildung 4.13 dargestellt. Die Auswertung liefert immer noch eine Standardabweichung von $\sigma = 0,023\,\mathsf{mrad}$, d.h. obwohl nun vier Werte zusammen genommen wurden, hat sich der Fehler nicht einmal halbiert. Ich komme daher zu dem Schluss, dass sich die Rasnik-Winkelauslese für die Myonkammern nicht lohnt, und stattdessen der Winkel der vierten Cross Plate aus den Rasnik-y-Werten berechnet wird. Mehr Information über die Hochspannungsseite liefern die Rasniks leider nicht, so dass wir uns nun der nächsten Cross Plate zuwenden.

4.3.2 Rekonstruktion der mittleren Cross Plate

Auch bei dieser Cross Plate wollen wir zunächst die Drehung um die Längsachse der Kammer betrachten. Bei den Messungen war auch die mittlere Cross Plate mit einer elektronischen Wasserwaage ausgerüstet. Somit besteht ebenfalls die Möglichkeit, den rekonstruierten Winkel mit zwei unabhängigen Messgeräten zu vergleichen. In Abbildung 4.14 sind wieder alle drei Werte für den Winkel eingetragen. Man sieht sofort, dass hier die Abweichungen zur Wasserwaage deutlich größer sind. Die Messmaschinenwerte passen jedoch recht gut zu den rekonstruierten Winkeln. Betrachten wir nun, was die Statistik zu den Daten sagt. Die berechneten Winkel haben einen Standardfehler von $\sigma = 1,8\,\mu\mathsf{rad}$. Das ist erstaunlicherweise geringer als bei der vierten Cross Plate. Da zwei Linsen einen kleinen Abstand von der Drehachse haben, würde man hier einen größeren Fehler erwarten. Andererseits ist hier der Fehler auf die y-Werte selber kleiner, denn er entspricht hier dem Fehler der Analyse selbst, während er bei der vierten Cross Plate um den Faktor $\frac{L_{\mathrm{Sys}}}{L_{\mathrm{Cam}}} > 1$ größer ist.

Schauen wir uns wieder die quadratische Abweichung (χ^2) der Messmaschinenpunkte vom jeweiligen Mittelwert des Fits an. Für die in Abbildung 4.14 dargestellten Daten berechne ich $\chi^2 = 54,4$ bei 54 Freiheitsgraden und einer Standardabweichung von $3\,\mu\mathsf{rad}$. Den Fehler auf die Messmaschinenwerte kann man, im Vergleich zur geringen Standardabweichung, nicht

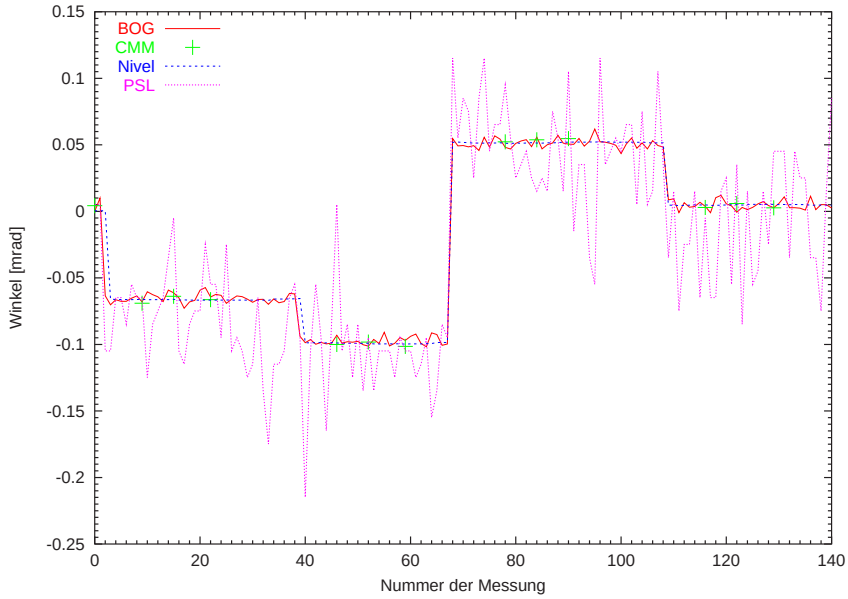


Abbildung 4.12: *In diesem Plot werden die berechneten Werte für α_4 mit Messungen der CMM und der elektronischen Wasserwaagen verglichen. Zudem ist die Winkelauslese des Rasniks „PSL“ eingetragen.*

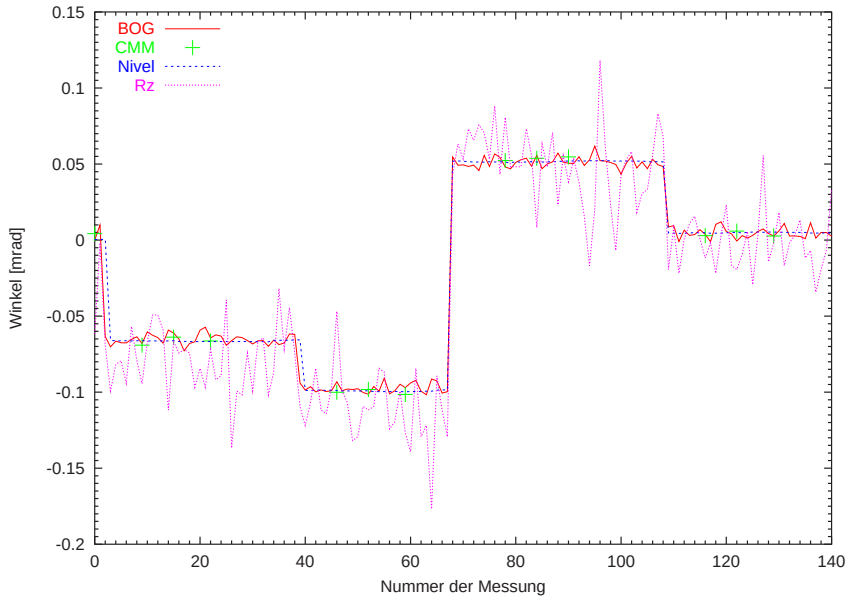


Abbildung 4.13: *In diesem Bild sind die Winkel α_4 aus unterschiedlichen Datenquellen gezeigt. Mit Rz ist der Mittelwert der Winkelauslese aller symmetrischen Rasniks bezeichnet. Die anderen Kurven entsprechen denen aus den vorigen Plots.*

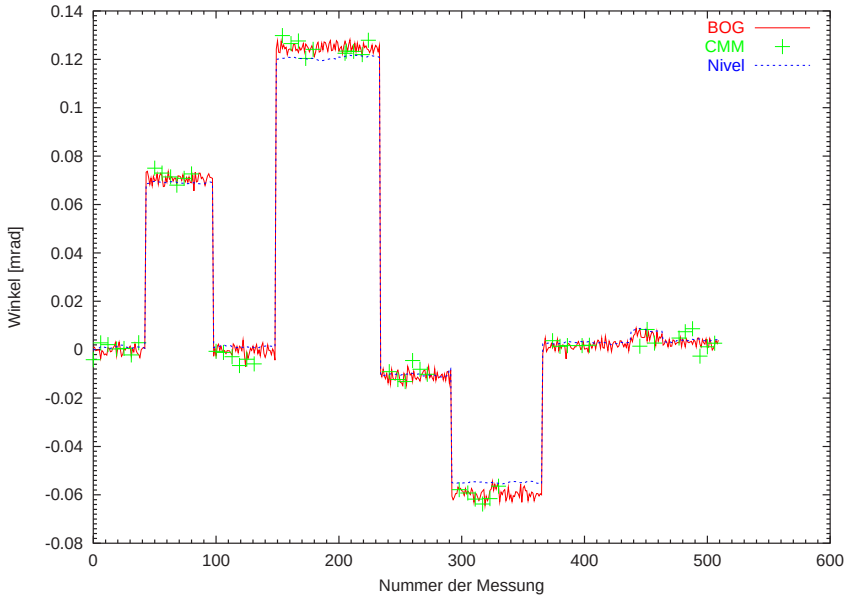


Abbildung 4.14: *Der Vergleich von berechnetem Wert für α_3 mit Messungen der Koordinaten-Messmaschine und elektronischer Wasserwaagen.*

mehr vernachlässigen. Ich nehme für die kombinierte Genauigkeit $3\,\mu\text{rad}$ an, wobei die Koordinaten-Messmaschine und die Rekonstruktion zu gleichen Teilen beitragen. Die Wahrscheinlichkeit für ein so großes χ^2 liegt bei $45,9\%$.

Betrachten wir jetzt die Wasserwaagen. Es fällt auf, dass die Wasserwaage nicht so große Winkel zeigt wie die anderen Sensoren. Man muss aber auch sehen, dass die Amplitude etwa doppelt so groß wie bei der vierten Cross Plate ist. Sehen wir uns im vierten Bereich (151 – 232) den rekonstruierten Winkel genauer an. Er liegt bei $124,7\,\mu\text{rad} \pm 1,6\,\mu\text{rad}$. Die Wasserwaagen liefern hier: $120,7\,\mu\text{rad} \pm 0,6\,\mu\text{rad}$. Das sind $4\,\mu\text{rad}$ oder 3% Abweichung. Laut Hersteller (Leica) haben unsere Wasserwaagen (*NIVEL20*) eine Genauigkeit von $\pm 0,005\,\text{mrad} + 0,5\%$ (siehe http://www.leica-geosystems.com/ims/product/spec_nivel20.htm). Somit liegt die Abweichung gut innerhalb der Spezifikation der Wasserwaagen. Doch vergleichen wir wie gehabt die Rekonstruktion mit den Werten der Wasserwaagen: $\chi^2 = 1507$ bei 490 Freiheitsgraden. Dieses Ergebnis ist erwartungsgemäß nicht ganz so gut. Wir können aber im Rahmen unserer Überprüfungsmöglichkeiten schließen, dass wir die Winkel α_3 der dritten Cross Plate auf mindestens $\pm 5\,\mu\text{rad}$ genau bestimmen können. Das entspricht dann einem maximalen Höhenfehler von $\pm 2,1\,\mu\text{m}$ an den Enden der Cross Plate.

Bei dieser Cross Plate kann nun auch die Translation des Mittelpunktes berechnet werden (siehe Abbildung 4.15). Die Werte für die Verschiebung wurden aus dem gleichen Datensatz rekonstruiert wie der Winkel, sodass sie

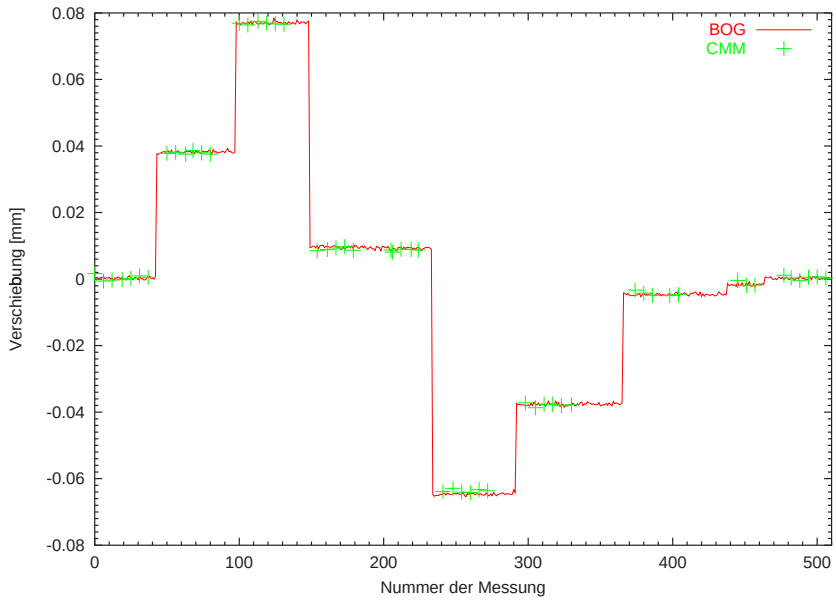


Abbildung 4.15: *Die Messung der Verschiebung ΔY_3, einmal aus den Rasnik-Werten und einmal von der CMM.*

direkt vergleichbar sind. Leider konnten diese Werte nur mit der Messmaschine verglichen werden – ein weiteres unabhängiges Messsystem war nicht vorhanden.

Ich nehme eine Genauigkeit von 1,5 μm für die Koordinaten-Messmaschine an. Den Standardfehler der berechneten Position von $\sigma = 0,6\,\mu$m habe ich bei der folgenden Betrachtung nicht berücksichtigt. Das χ^2 berechnet sich mit diesen Daten zu 44,2 (54 Freiheitsgrade). Die Wahrscheinlichkeit dafür liegt bei 83%, das χ^2 ist also gut verträglich mit der Analysemethode.

Nun hat der Schwerpunkt der Cross Plate aber zwei Koordinaten. Wir müssen also auch die z-Komponente rekonstruieren. In Abbildung 4.16 ist eben diese eingetragen. Es handelt sich wieder um den gleichen Datensatz. Die Verschiebungen in z-Richtung waren teilweise zufällig, da die Cross Plate während der Messungen auf einer Höheneinstellung lag, auf der sie horizontal rutschen konnte. (Die Schrauben waren während der Messungen nicht angezogen.) Durch Ändern der Höhe entsteht so zum Teil eine Bewegung in z-Richtung, die aber nicht weiter bestimmt ist. Im Gegensatz dazu habe ich auch gezielte Verschiebungen in z-Richtung durchgeführt.

Die Koordinaten-Messmaschine misst in z-Richtung (konstruktionsbedingt) etwas genauer und ich nehme einen Wert von $\sigma = 1\,\mu$m an. Die Standardabweichung der Rekonstruktion liegt mit $\sigma = 0,6\,\mu$m wieder unter diesem Fehler. Man erhält so für die Höhenkoordinate ein $\chi^2 = 38,2$ mit wiederum 54 Freiheitsgraden. Das ist ein sehr kleines χ^2 und die Wahrscheinlichkeit, dieses oder ein größeres χ^2 zu erhalten, liegt bei 94,8%. Da

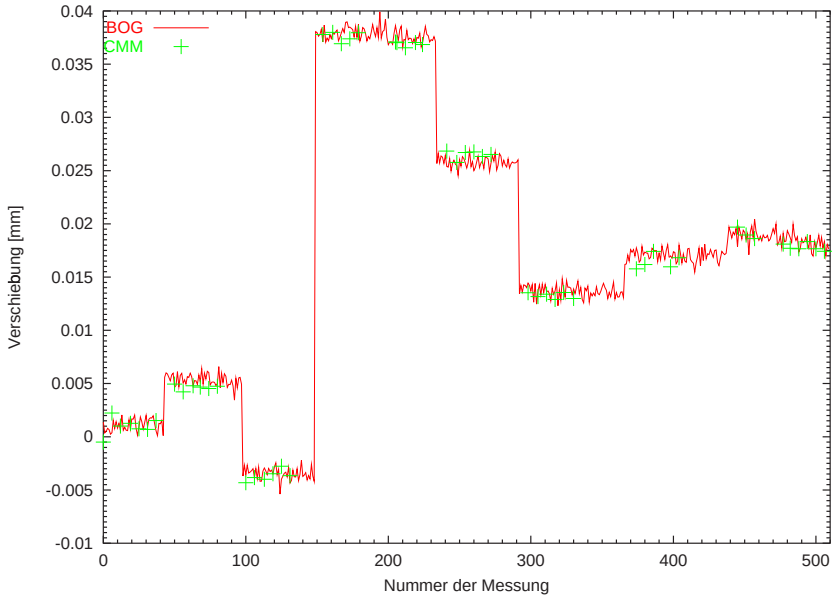


Abbildung 4.16: *In dieser Graphik wird der berechnete Wert für ΔZ_3 mit Messungen der CMM verglichen.*

der Fehler der Messmaschine jedoch nicht rein statistisch ist, wird er durch das χ^2 nicht ganz richtig bewertet.

Fassen wir die erreichbaren Messgenauigkeiten der mittleren Cross Plate zusammen:

$$\begin{aligned} \text{Winkel } (\alpha_3): &\quad \pm\, 1,8\,\mu\text{rad} \\ y\text{-Koordinate } (\Delta y_3): &\quad \pm\, 1,5\,\mu\text{m} \\ z\text{-Koordinate } (\Delta z_3): &\quad \pm\, 1\,\mu\text{m} \end{aligned}$$

Kombinieren wir die letzten beiden, so kann man sagen, dass der Schwerpunkt der mittleren Cross Plate auf mindestens $2\,\mu$m genau gemessen werden kann.

Die bis hierhin vorgestellten Ergebnisse lassen sich auch auf die meisten Myonkammern im Fassbereich des Detektors übertragen. Die zuvor definierten Winkel (α_3, α_4) und Verschiebungen (Δy_3, Δz_3) lassen sich mit jedem „normalen“ In-plane Alignment rekonstruieren. Das heißt, dass sich die „normalen“ Kammern, wie sie in Abbildung 1.7 gezeigt sind, mit diesen Parametern beschreiben lassen. Die erreichbaren Genauigkeiten hängen dabei von den zugrundeliegenden Rasnik-Strecken ab. Hierbei können sich leichte Unterschiede zu den Messungen an den BOG-Kammern ergeben. Besonders bei den längeren Kammern ist mit einer Vergrößerung der Unsicherheiten zu rechnen. Dennoch ist zu erwarten, dass auch bei diesen Kammern meine Methode eine erfolgreiche Beschreibung ergibt.

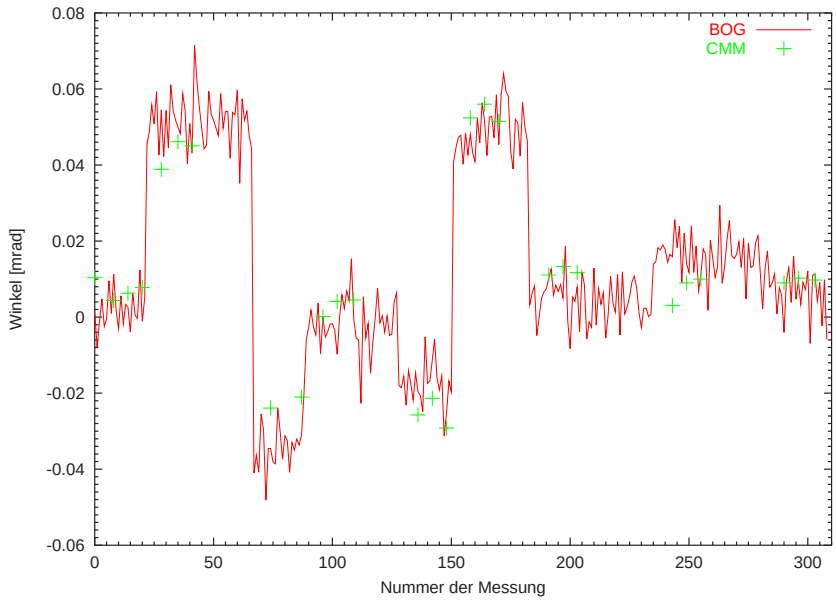


Abbildung 4.17: *Der Vergleich von α_2 aus dem Fit mit Messungen der Koordinaten-Messmaschine.*

4.3.3 Die Spezial Cross Plate

Die „Spezial Cross Plate“ ist eine Besonderheit des BOG-Kammertyps. Diese Cross Plate ist ca. 1 m von der Read Out Cross Plate entfernt und auf ihr enden einige Randrohre. Für diese Rohre ist sie die High Voltage Seite. Aus Rasnik-Sicht ist sie jedoch eine mittlere Cross Plate. Sie sitzt zwar nicht genau in der Mitte, aber sie trägt Linsen. Und Linsenpositionen werden von der Analyse berechnet. Zu beachten ist, dass die Rasniks als z-Koordinate der Linsen den Abstand von der Mitte ausgeben und nicht die Abweichung vom Sollwert. Für unsere Kammer erhalten wir also Werte von etwa $-700\,\mathrm{mm}$. Dieser Offset wird, wie bei den xy-Werten auch, von der Nullpunktkorrektur ausgeglichen.

Wenden wir uns jetzt den rekonstruierten Werten zu. Da diese Cross Plate länger ist als die anderen und die Rohre nur an den Enden kleben, ist hier der Winkel besonders wichtig. Wir betrachten diese Cross Plate vollkommen unabhängig von der mittleren, und wir benutzen nur die Werte der Rasniks, deren Linsen in der Spezial Cross Plate befestigt sind. Wir können dann den Winkel dieser Cross Plate und der vierten berechnen. Schauen wir uns ersteren genauer an: In Abbildung 4.17 ist, wie gehabt, der Winkel dieser Cross Plate rekonstruiert und von der Koordinaten-Messmaschine gemessen eingetragen. Als Fehler nehme ich hier die Standardabweichung der berechneten Winkel. Wie bereits im vorigen Kapitel besprochen, sind diese Rasnik-Strecken etwas schlechter und weisen eine größere statistische

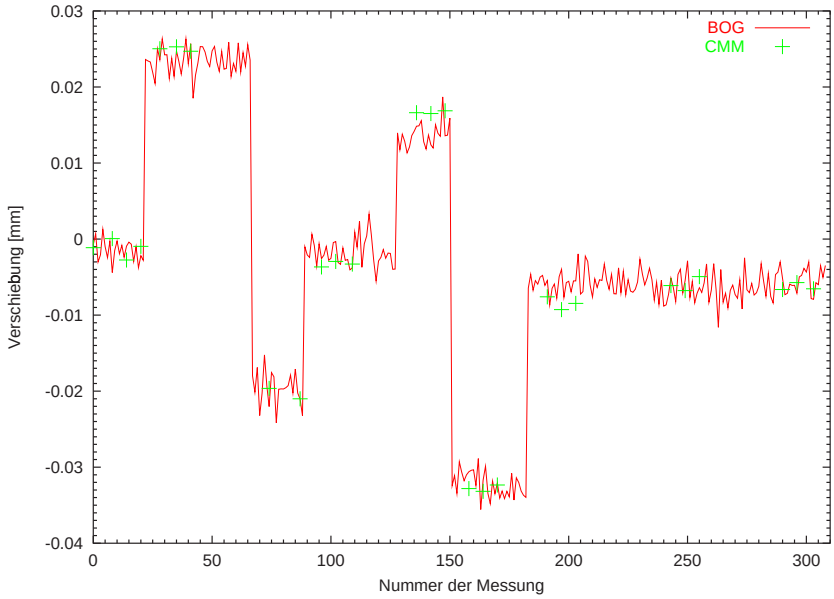


Abbildung 4.18: *Gleichzeitige Darstellung der berechneten Werte für ΔY_2 mit Messungen der CMM.*

Schwankung auf. Mit diesem Vorgehen bestimme ich $\chi^2 = 25,1$ bei 25 Freiheitsgraden.

Damit ergibt sich eine Messgenauigkeit für den Winkel von $6,6\,\mu\mathsf{rad}$. Bei einer Cross Plate Länge von $1,2\,\mathsf{m}$ entspricht das einem Fehler auf die Höhe der Enden von ca. $4\,\mu\mathsf{m}$. Für den Winkel der vierten Cross Plate ergibt sich mit diesen Rasniks berechnet ein ähnlicher Fehler. Da dieser fast doppelt so groß ist wie bei den symmetrischen Rasniks, verwende ich den so berechneten Winkel nicht weiter.

Wir haben es zwar hier mit einer „End Cross Plate“ zu tun, aber wir können trotzdem ihren Mittelpunkt bestimmen – sie ist eben ein Spezialfall. Das ist auch sehr wichtig, da an dieser Cross Plate keine Sensoren zu den Nachbarkammern vorhanden sind. Blicken wir also auf Abbildung 4.18. Das Bild zeigt wieder die gleiche Messreihe wie Abbildung 4.17, diesmal aber die Höhenwerte. Mit einem Standardfehler von $\sigma = 1,8\,\mu\mathsf{m}$ erhalte ich ein $\chi^2 = 23,3$ (25 Freiheitsgrade). Das ist auch eine hervorragende Übereinstimmung mit den statistischen Abweichungen.

Schauen wir uns zum Abschluss noch die andere Richtung an. Die Abbildung 4.19 zeigt die z-Koordinaten von Messmaschine und Rasnik-Rekonstruktion. Die Standardabweichung der Rasnik-Werte liegt recht nahe beim Fehler der Messmaschine, sodass ich hier einen kombinierten Fehler von $\sigma = 2\,\mu\mathsf{m}$ annehme. Ich bekomme damit ein $\chi^2 = 21,8$ bei wieder 25 Freiheitsgraden.

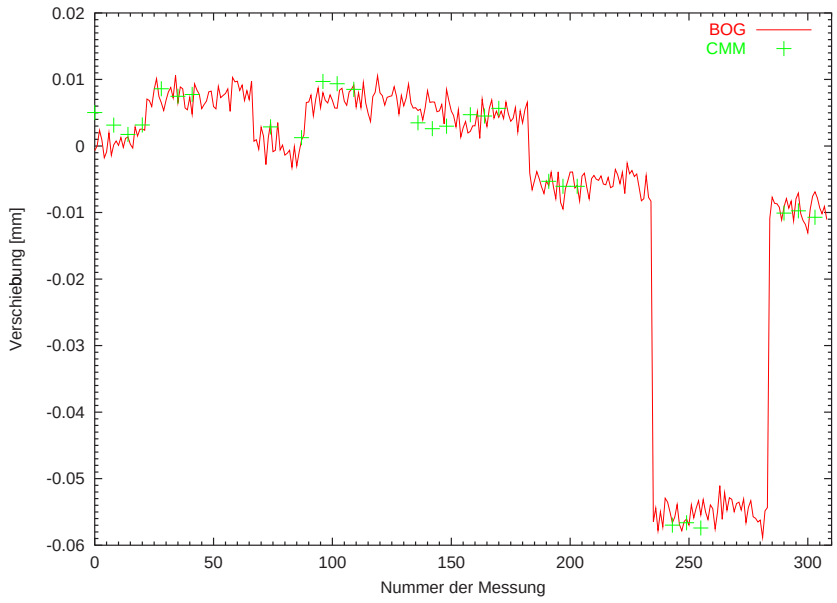


Abbildung 4.19: *Ein weiterer Vergleich von berechnetem Wert, hier ΔZ_2, mit Messungen der Messmaschine.*

Für die Unsicherheiten der Spezial Cross Plate ergeben sich folgende Werte:

$$\begin{aligned} \text{Winkel } (\alpha_2): &\quad \pm\, 6,6\,\mu\text{rad} \\ y\text{-Koordinate } (\Delta y_2): &\quad \pm\, 1,8\,\mu\text{m} \\ z\text{-Koordinate } (\Delta z_2): &\quad \pm\, 2\,\mu\text{m} \end{aligned}$$

Die Streuungen sind bis zu dreimal größer als bei den symmetrischen Rasniks, aber zusammengenommen kann man sagen, dass die Cross Plate besser als $5\,\mu$m bestimmt werden kann.

Vor allem für die Rohre, die auf dieser Cross Plate enden, ist das ein gutes Resultat: Die Bewegung der Endstopfen kann besser als $5\,\mu$m bestimmt werden, sofern die Cross Plate selbst starr bleibt.

In diesem Kapitel werden Parameter zur Beschreibung der Kammerform vorgestellt. Ich habe gezeigt, wie diese Parameter aus den Rasnik-Sensoren bestimmt werden können. Um dieses Kapitel abzuschließen, möchte ich die gefundenen Genauigkeiten für die BOG-Kammern tabellarisch zusammenfassen:

High Voltage Cross Plate	α_4:	$\pm\, 3,5\,\mu$rad
Mittel Cross Plate	α_3:	$\pm\, 1,8\,\mu$rad
	Δy_3:	$\pm\, 1,5\,\mu$m
	Δz_3:	$\pm\, 1\,\mu$m
Spezial Cross Plate	α_2:	$\pm\, 6,6\,\mu$rad
	Δy_2:	$\pm\, 1,8\,\mu$m
	Δz_2:	$\pm\, 2\,\mu$m

Vergleichen wir das mit dem „Technical Design Report“ [ATL97]: Die Abweichung der Drähte von der Sollposition („wire displacements“) soll auf $10\,\mu$m oder besser gemessen werden. Man kann daher bei den BOG-Kammern die Rasniks als „voll tauglich“ für diese Aufgabe bezeichnen.

Kapitel 5

Rahmen und Röhren

In diesem Kapitel wird überprüft, wie stark das Gewicht der Röhren den Kammerrahmen verformt. Es wird untersucht, ob das In-plane Alignment die Verformungen auflösen kann. Anschließend wird die Höheneinstellung der inneren Querträger vorgestellt. Mit ihrer Hilfe sollen die Rohre dem Durchhang des Drahtes angeglichen werden. Dabei wird diskutiert, ob das In-plane Alignment diese Einstellung unterstützen kann.

5.1 Verformungen durch Eigengewicht

Im Atlas-Detektor werden die Myonkammern an den Enden, d.h. in der Nähe der ersten und letzten Cross Plate aufgehängt. Da die Aufhängungen keine Drehmomente auf die Kammer bringen, hängt sie einfach durch, fast wie eine Stange, die nur an den Enden unterstützt ist. Das lässt die beiden äußeren Cross Plates nach innen kippen, was wir leider mit den Rasniks nicht messen können. Was man jedoch messen kann, ist die Auslenkung in der Mitte der Kammer. Somit gelangt man zu einer Abschätzung für den Kippwinkel.

Nehmen wir an, die Longbeams verhalten sich wie ein Lineal, d.h. wir minimieren die Spannungsenergie und vernachlässigen das Eigengewicht. Das entspricht einer Spline-Interpolation mit drei Punkten (die Spezial Cross Plate wird hier vernachlässigt) und Mittelpunktsymmetrie. Es ist nun leicht zu sehen, dass der Longbeam dann durch ein Polynom 2. Grades beschrieben wird. Man findet dann für den Kippwinkel γ der ersten Cross Plate:

$$\gamma \approx 2\,\frac{\Delta y}{a} = 2\,\frac{\Delta y}{1853\,\mathsf{mm}} \tag{5.1}$$

Dabei gibt a die halbe Länge der Kammer an, die bei den BOG Kammern $1853\,\mathsf{mm}$ beträgt.

In Abbildung 5.1 ist gezeigt, wie weit der Spacer ohne Röhren durchhängt. Im ersten Teil der Messung liegen die mittleren Cross Plates auf und der

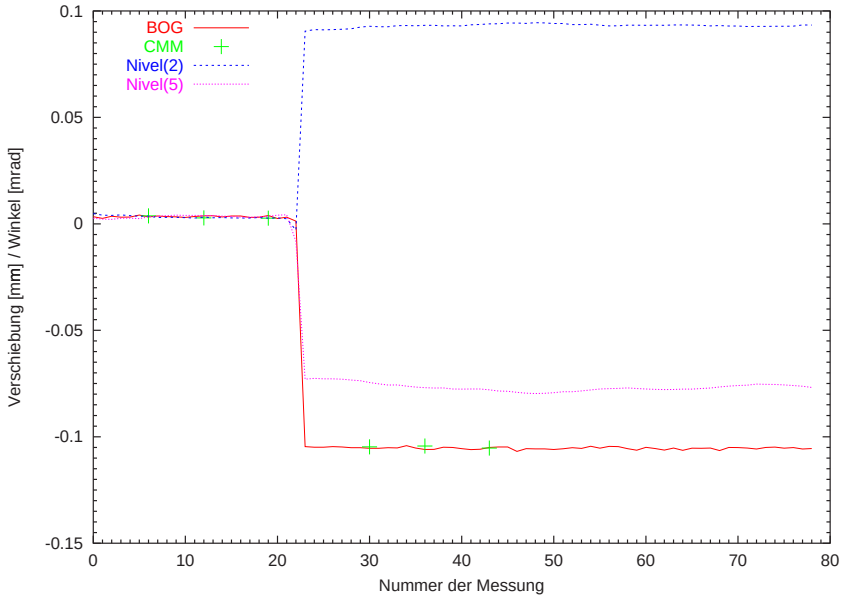


Abbildung 5.1: *Diese Messung zeigt das „Absacken“ der mittleren Cross Plate, wenn sie frei hängen gelassen wird. Bis Messung 20 liegt die mittlere Cross Plate auf, ab Messung 25 hängt sie frei.*

Rahmen ist in „Null-Position“. Dann wurden die Stützen unter den beiden inneren Cross Plates entfernt, sodass die Kammer nur noch an den Enden auflag. Die mittlere Cross Plate ist nun ca. 110 μm tiefer als zuvor. Mit dem einfachen Modell erwartet man damit einen Winkel von 120 μrad. Zum Vergleich sind in dieser Abbildung noch die Winkel der High Voltage und der Read Out Cross Plates eingetragen, wobei hier die mittlere Cross Plate als Referenz dient. Die elektronischen Wasserwaagen liefern hier Winkel von etwa 90 μrad. Die beiden Enden kippen erwartungsgemäß nach innen und wir erhalten Werte mit unterschiedlichen Vorzeichen. Für die Klebeflächen bedeutet das eine Verschiebung von 15 μm aus der Normalposition.

Das einfache Parabelmodell ist nicht sonderlich genau ($\pm 25\%$), es liefert aber dennoch eine brauchbare Abschätzung für die zu erwartenden Winkel. Eine genauere Kenntnis wird erst dann wichtig, wenn man den Einfluss dieser Winkel auf die Röhren hinreichend gut simulieren kann. Es soll daher hier bei dieser einfachen Abschätzung bleiben.

Kommen wir nun zu einer anderen Formveränderung, die durch Gewichtskräfte verursacht wird. Die aufgeklebten Rohre und vor allem die Endstopfen müssen auch von dem Rahmen getragen werden. Betrachten wir dazu unsere Spezial Cross Plate: Bei ihr sind die stärksten Effekte zu erwarten, denn sie trägt Endstopfen, ist lang und nur an den Enden belastet.

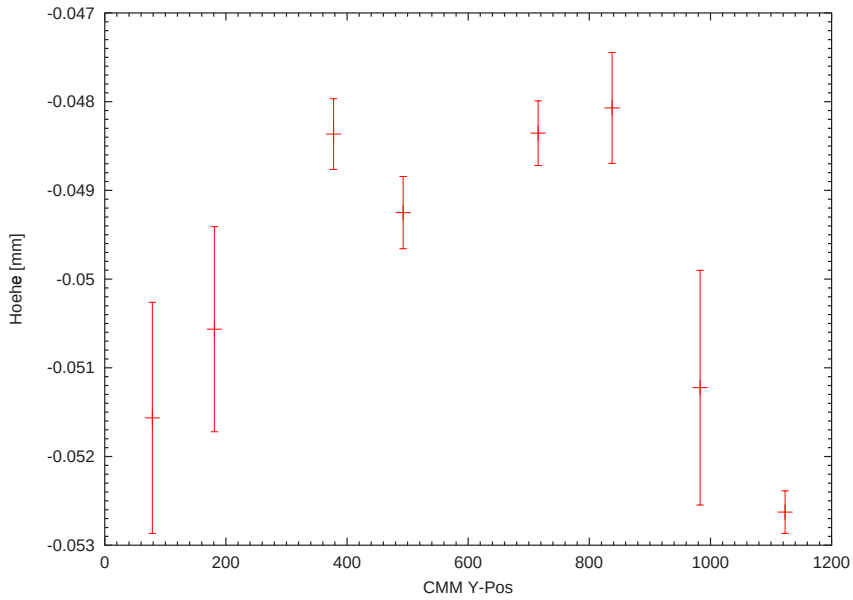


Abbildung 5.2: *Cross Plate 2 ist mit je 6 kg pro Seite belastet. Der Nullpunkt der Ordinate entspricht dem unbelasteten Zustand.*

Um die Verformungen zu sehen, wollen wir belasteten und unbelasteten Zustand vergleichen. Allerdings können wir dafür keine Rohre aufkleben. Zum einen muss der Spacer zum Kleben bewegt werden, was den Vergleich erschwert. Zum anderen will ich die Belastung auch wieder rückgängig machen können. Daher habe ich die Rohre durch Gewichte ersetzt.

Ein Endstopfen (mit Signalkappe) wiegt knapp 50 g. Mit 18 Rohren pro Seite und Lage trägt die Cross Plate etwa $3,6$ kg Endstopfen. Hinzu kommen noch Gasversorgung und Elektronik. Damit das Gewicht keinesfalls unterschätzt wird, habe ich jeweils die letzten 20 cm mit je 6 kg belastet, also 12 kg auf dem gesamten Träger aufgelegt.

Die zugehörige Messung ist in Abbildung 5.2 dargestellt. Ich habe entlang der Cross Plate 8 Kugeln angeklebt, die ich mit der Messmaschine vermessen habe. In der Abbildung ist die Differenz der Kugelpositionen vor und während der Belastung gezeigt. Zum einen sieht man, dass die Kugeln um ca. 50 μm abgesunken sind, was sich durch die aufgelegte Masse erklärt. Zum anderen erkennt man, dass die Enden „herunter hängen“. Die im Vergleich recht großen Fehler, die allein von der Messmaschine stammen, erlauben es aber nicht, verschiedene Modelle für die Verformung gegeneinander auszuschließen. Zudem liegen die Messpunkte nur 5 μm auseinander, bei dem Gewicht der Stopfen und sonstigen Versorgungsteile kann man diesen Effekt getrost vernachlässigen.

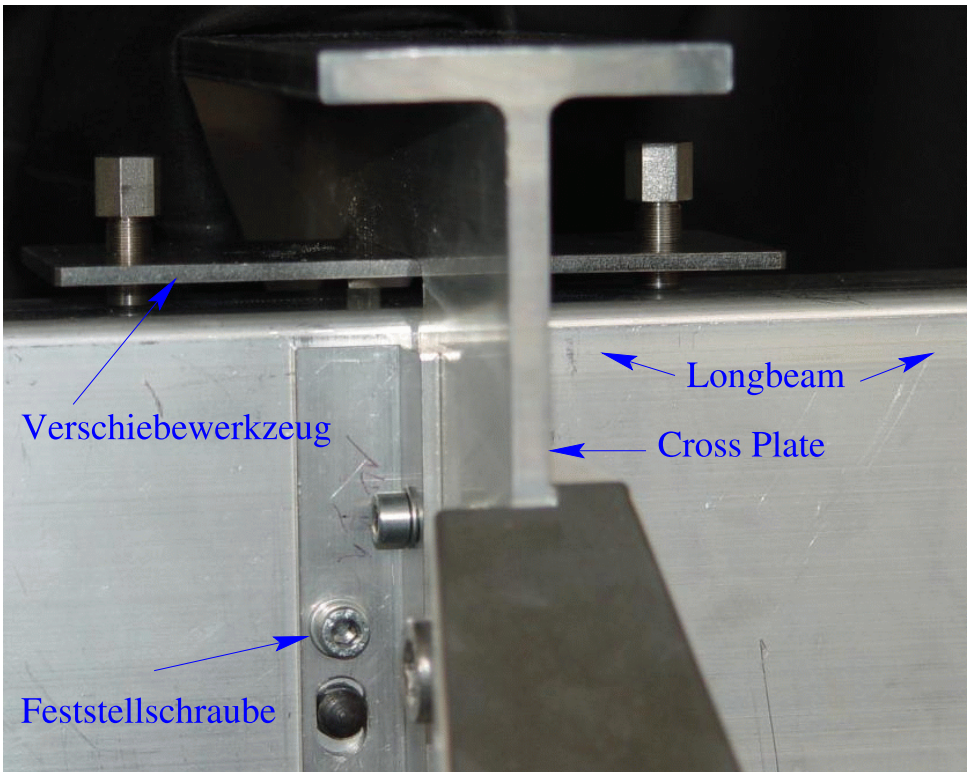


Abbildung 5.3: *Auf diesem Photo ist das Werkzeug für die Höhenverstellung gezeigt. Es ist über dem Longbeam durch die Cross Plate geschoben und ermöglicht es, an den Schrauben die Höhe zu justieren. Vor der Justage sind die Feststellschrauben zu lösen, da sie die Höhe fixieren.*

5.2 Die Höhenverstellung

Die inneren Querträger der Myonkammern sind mit einer Höhenverstellung ausgerüstet. Da der Draht nur an den Enden befestigt ist, hängt er entsprechend durch. Die Rohre und die ganze Kammer hängen natürlich auch durch, aber eben nicht im gleichen Maße. Je nach Einbaulage der Kammern unterscheidet sich der Durchhang. Die Höhenverstellung dient dazu, die Rohre dem Draht anzugleichen. Es wird versucht, den Draht möglichst in der Mitte zu halten, damit sich das elektrische Feld in den Röhren nicht zu stark ändert.

Wie ich in Abbildung 5.1 schon gezeigt hatte, sackt der Trägerrahmen um ca. 100 μm in der Mitte ab. Nach Rechnungen von U. Landgraf haben die langen Rohre eine freie Drahtlänge von 3756 mm. Das führt bei einer Drahtspannung von 350 g zu einem Durchhang in der Mitte von 190 μm (bzgl. der Endstopfen).

Es müssen somit die inneren Cross Plates um ca. 100 μm nach unten verschoben werden, damit der Draht wieder in der Mitte der Rohre sitzt. Die

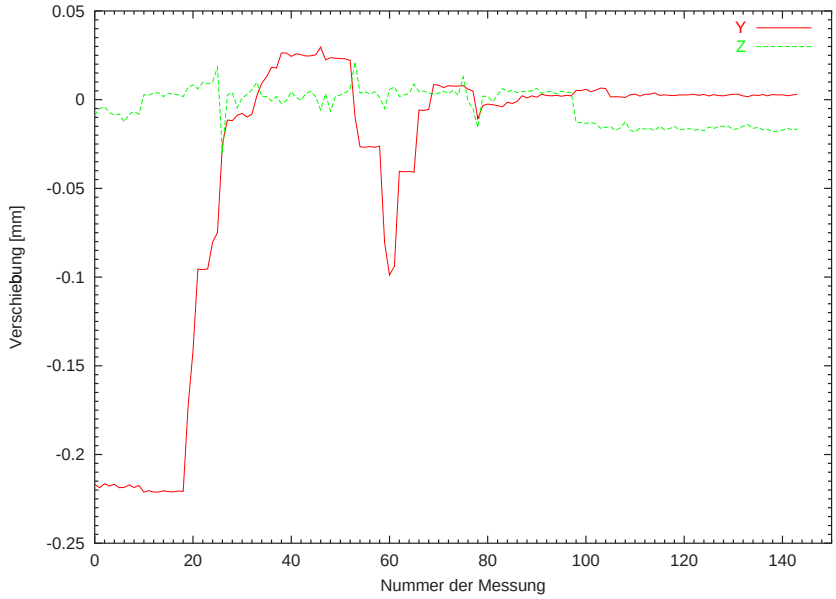


Abbildung 5.4: *Hier ist die Bewegung des Schwerpunktes der mittleren Cross Plate gezeigt, während die Höhe um ca. 200 μm verstellt wurde. Die Messwerte wurden mit den Rasnik-Sensoren aufgenommen.*

Frage ist, wie man diese Einstellung durchführt. Wichtig ist, dass während der Justage das Gewicht der Cross Plates auf den Längsträgern lastet und nicht durch die Verschiebeeinrichtung getragen wird. Die Idee ist, ein kleines Werkzeug zu bauen, das nur am Rahmen befestigt ist, und den Eigendurchhang kaum beeinflusst. Unsere (vorläufige) Lösung besteht aus einer Metallplatte, durch die zwei Schrauben mit einem Feingewinde gehen, siehe Abbildung 5.3. Diese Platte wird über den Longbeams durch die Cross Plate geschoben, sodass die Cross Plate auf der Platte aufliegt und sich die Schrauben am Längsträger abstützen. Mit diesem Werkzeug lässt sich die Höhe problemlos auf 5 μm genau einstellen.

In Abbildung 5.4 ist ein Justagevorgang protokolliert. Es sind die yz-Koordinaten des Schwerpunktes eingetragen. Die ersten 10 Punkte halten die Ausgangslage fest. Danach wurden die Schrauben gelöst und die Cross Plate lag auf der Platte. Bis Messung 97 wurde justiert und ab 105 waren die Schrauben wieder festgezogen. Es wurde in dieser Messung versucht, die Höhe auf Null zu justieren. Wie man sieht, ist das mit einer Präzision von unter 5 μm gelungen. Man sieht aber auch, dass sich die z-Koordinate ebenfalls geändert hat. Solange die Schrauben lose sind, ist die Position unbestimmt. Beim Anziehen der Schrauben verschiebt sich die Cross Plate um ca. 10 μm. Der Grund dafür ist die Form des Longbeams. Die Befestigungswinkel gleiten beim Heben an seinen Seiten entlang. Wenn diese ein wenig schräg sind, wird auch die z-Koordinate geändert – genau das ist hier geschehen.

Diese unerwünschte Bewegung kann minimiert werden, indem man den Durchhang vor dem Kleben der Rohre möglichst gut einstellt und später nur noch wenig korrigieren muss. Es sollte so möglich sein, die z-Position auf ca. $5\,\mu\mathsf{m}$ vorher zu bestimmen.

Die „Spezial Cross Plate“ ist auch verstellbar befestigt. Die langen Rohre kommen mit ihr jedoch nicht in Kontakt, es ist also nicht erforderlich, sie auf den Draht zu justieren. Sie bestimmt ausschließlich die Position der Enden der kurzen Rohre. Aus Symmetriegründen scheint es das Sinnvollste zu sein, die Höhe so einzustellen, dass sie auf einer Geraden mit der ersten und vierten Cross Plate liegt, d.h. alle Endstopfen sind in einer Ebene.

Die Untersuchungen zur Belastung der Querträger haben gezeigt, dass das Gewicht der Röhren nur eine geringe Deformation hervorruft. Die Abweichungen von einer ebenen Cross Plate sind geringer als $5\,\mu\mathsf{m}$ und können vom In-plane Alignment nicht aufgelöst werden. Der Test der Höhenverstellung zeigt, dass die Höhe der Querträger auf $\pm 5\,\mu\mathsf{m}$ genau eingestellt werden kann. Dabei kann mit Hilfe der Rasnik-Sensoren die Einstellung vermessen und überprüft werden.

Kapitel 6

Resümee

Die vorliegende Arbeit beginnt mit einer kurzen Vorstellung des ATLAS-Detektors. Bei der Entwicklung des Detektors wurde auf eine hohe Auflösung bei der Messung der Myonimpulse Wert gelegt. Daraus ergibt sich, dass die Sagitta der Myonbahn auf ca. 50 μm genau gemessen werden muss. Diese Unsicherheit setzt sich aus verschiedenen Teilen zusammen, so spielen die Produktion und Verklebung der Röhren eine Rolle. Aber auch die Überwachungssysteme (Alignment) tragen zu dieser geringen Unsicherheit bei.

Das zweite Kapitel stellt eines dieser Systeme, das In-plane Alignment, detailliert vor. Dabei zeige ich, dass dieses System mit Hilfe meines Treibers, meiner Anpassung der RasMux-Steuerung und der NIKHEF-Analyse auch unter Linux eingesetzt werden kann. Dadurch steht dem Einsatz von Linux für dieses System bei ATLAS nichts mehr im Wege. Durch die Modularität meiner Programme ist es leicht möglich, das Rasnik-System für die Messung der Linsenbrennweite einzusetzen.

Ebenso leicht kann das System mit der „Myon MDT Produktionsdatenbank“ (*MMPDB*) verbunden werden, sodass alle Informationen an einer zentralen Stelle gespeichert und der Zusammenbau der Kammern unterstützt werden kann.

Nachdem die Tauglichkeit des Rasnik-Systems unter Linux gezeigt ist, werden die speziellen Rasnik-Strecken in den BOG-Kammern untersucht. Dazu habe ich zuerst die Linsen vermessen, die in die Kammer eingebaut werden. Bei den Messungen an der Kammer stellte sich heraus, dass die erreichbaren Genauigkeiten von Strecke zu Strecke schwanken, dass aber alle die Spezifikationen [ATL97] erfüllen.

Als nächstes wende ich mich der Frage zu, wie die Rasnik-Daten zu interpretieren sind, d.h. welche Aussage sie über die aktuelle Kammerform machen. Nimmt man die Randbedingungen hinzu, die durch die Aufhängung der Kammern gegeben sind, so lassen sich die Positionen aller Querträger (Cross Plates) rekonstruieren. Dazu zerlege ich die Verformungen in elemen-

tare Bewegungen (Drehungen und Verschiebungen) und kann so alle Bewegungen bestimmen, die in der Ebene senkrecht zu den Röhren bzw. Drähten stattfinden. Das sind gerade die Bewegungen, welche die Messungen der Sagitta der Myonbahnen beeinflussen. Durch die Rasniks können die Lagen der Querträger auf besser als $5\,\mu$m (siehe Kapitel 4) genau bestimmt werden. Das In-plane Alignment kann also dazu beitragen, die Messgenauigkeit zu verbessern.

Nimmt man an, dass die Cross Plates starr sind, so sind ihre Positionen durch die Rasnik-Messungen überbestimmt. Die Überbestimmung lässt sich auflösen, indem man dynamische Parameter einführt. Zwei dieser Parameter lassen sich als die Durchbiegungen der inneren Cross Plates interpretieren. Die anderen sind mit der thermischen Ausdehnung korreliert. Die Temperatur kann allerdings besser mit speziellen Temperatursensoren des In-plane Alignments gemessen werden. Daher werden diese freien Parameter nicht weiter ausgewertet.

Zum Abschluss wird noch überprüft, ob sich die Querträger unter dem Gewicht der Röhren bzw. Endstopfen merklich verformen. Es stellt sich dabei heraus, dass die Verformungen selbst bei doppelter Belastung vom In-plane Alignment nicht aufgelöst werden können. Allerdings sind diese Abweichungen so gering, dass alle Positionen innerhalb der Spezifikationen bleiben, und man braucht sich um das Gewicht keine weiteren Sorgen zu machen. Ebenfalls überprüft wurde, ob der Durchhang der Längsträger (ca. 0,2 mm ohne Rohre) durch Verschieben der Cross Plates ausgeglichen werden kann. Das von uns entwickelte Werkzeug hat diesen Test gut bestanden.

Anhang A

Die *Rasnik for Linux* Umgebung

Dieser Anhang wird die Software aus technischer Sicht beleuchten, die ich im Laufe der vorliegenden Arbeit erstellt habe. Es sollen dabei die Möglichkeiten dieser Paketsammlung dargestellt werden. Zudem sollen wertvolle Hinweise für alle gegeben werden, die mit meinen Programmen arbeiten oder sie erweitern.

A.1 Treiber für die DT 3152

Die Karte für Bildaufnahme *DT 3152* ist für Messaufgaben entwickelt worden. Der Hersteller selbst liefert nur Treiber für unterschiedliche Windows-Versionen.

Der hier vorgestellte Treiber wurde von mir mit allen Fähigkeiten ausgestattet, die in Verbindung mit dem Rasnik benötigt werden. Wie zahlreiche Rückmeldungen über das Internet belegen, fand dieser Treiber auch bei anderen Benutzern Verwendung.

Wie schon in Kapitel 2 erwähnt, klinkt sich der Treiber in die dafür vorgesehene Videoschnittstelle (V4L) des Linux-Kerns ein. Dieses Kernel-Subsystem stellt dem Anwender für jede registrierte Karte eine Gerätedatei (/dev/video*) bereit. Wird diese geöffnet, so kann die Karte über `ioctl(fd,cmd,data)`-Aufrufe konfiguriert werden. Dabei ist `fd` der Filedescriptor der offenen Videoschnittstelle und `cmd` ein Befehl an den Treiber. `data` ist ein Zeiger auf einen Datenbereich, der vom Befehl abhängt. Die zulässigen Befehle für meinen Treiber sind in Tabelle 2.1 zusammengefasst. Die meisten dieser Befehle sind in der V4L-Dokumentation ([Cox00]) ausführlich beschrieben.

Ich habe jedoch einige Befehle hinzugefügt, die die speziellen Fähigkeiten der *DT 3152* berücksichtigen. Sie sind alle in der Datei dt3152.h definiert und sollen im Folgenden beschrieben werden. Die Namensgebung der Befeh-

le ist analog zu denen in *Video for Linux*. Sie beginnen alle mit `VIDIOC` für „Video i/o control“, dann kommt ein `G` für „get“ oder `S` für „set“, d.h. ob aus dem Kernel gelesen oder hinein geschrieben wird. Die folgenden Buchstaben beschreiben dann die eigentliche Aktion. Die zusätzlichen Befehle der *DT 3152* sind:

`VIDIOCGINCREMENT/VIDIOCSINCREMENT`: Die Karte besitzt die Fähigkeit, bei jedem Bild eine Anzahl von Zeilen oder Spalten zu überspringen und somit das Bild zu verkleinern. Mit diesem Befehl wird diese Anzahl gesetzt. Die übergebene Struktur besteht aus zwei Integerwerten für die Spalten bzw. Zeilen.

`VIDIOCGLUT/VIDIOCSLUT`: Die *LUT* (Look up table) ist ein genau 256 Bytes großer Speicherbereich. Jeder vom A/D-Wandler gelieferte Wert wird als Index für diesen Speicher verwendet, und der Wert an dieser Stelle wird dann zurückgeliefert. Möchte man von diesen Fähigkeiten keinen Gebrauch machen (wie beim Rasnik), so ist die *LUT* mit den Werten $0 \ldots 255$ zu laden. Um beispielsweise das Bild zu invertieren, lädt man die Werte $255 \ldots 0$. Bei diesem Befehl verweist der `data`-Zeiger auf einen 256 Bytes großes Char-Array.

`VIDIOCGSYNC/VIDIOCSSYNC`: Dieser Befehl ermöglicht den Zugriff auf die vielfältigen Synchronisationsmöglichkeiten der Karte. Ihm wird die Struktur `struct video_sync` übergeben, in der z. B. der Eingangskanal (Pins 8–5) oder die Synchronisationsschwelle (in `mV`) gesetzt werden. Ebenso kann auf ein externes Signal umgeschaltet und die aktive Flanke gewählt werden.

`VIDIOCGPCLK/VIDIOCSPCLK`: Der programmierbare Taktgenerator der Karte (*DP8534* von National Semiconductor) kann hiermit kontrolliert werden. In der übergebenen Struktur `struct video_PixelClock` wird der gewünschte Pixeltakt angegeben oder der externe Takteingang (wie beim Rasnik) aktiviert.

`VIDIOCGDIGOUT/VIDIOCSDIGOUT`: Die *DT 3152* hat acht Digitalausgänge. Diese (*TTL* kompatiblen) Ausgänge können mit diesem Befehl ein- und ausgeschaltet werden. Ihm wird ein Integerwert übergeben, bei dem die unteren acht Bit den Ausgängen entsprechen.

`VIDIOCSFORMAT`: Ermöglicht den Zugriff auf das „Format RAM“, in dem das genaue Video-Format definiert ist. Hier werden die Positionen und Größen des aktiven Videobereiches, sowie der Austastlücken definiert. In der Struktur `struct video_format` sind Felder für alle diese Werte enthalten. Es können so auch nicht standardisierte Formate programmiert werden.

Zum Abschluss sei noch eine Erweiterung des *Video for Linux* Befehls `VIDIOCSCHAN` erwähnt. Es wurde die Norm `VIDEO_MODE_VV5430` hinzugefügt, die das Format für die RasCam einstellt.

Die Benutzung der Karte von Tcl/Tk aus gestaltet sich recht einfach. Das Paket `CaptureTcl` stellt den Befehl `Capture` bereit, der ein Einzelbild (Frame) in einem Tcl/Tk-Image ablegt. Dieses Kommando kann durch einige Optionen beeinflusst werden. Ich werde sie kurz beschreiben:

`-image`	Name des Images für das Bild.
`-device`	Gerätedatei der Karte (std: /dev/video).
`-channel`	Kanalnummer des Videoeingangs.
`-x/-y`	Legt die linke obere Ecke des Bildausschnittes fest.
`-width/-height`	Bestimmt die Breite und die Höhe des Bildausschnittes.
`-brightness`	Stellt die Helligkeit ein.
`-contrast`	Regelt den Kontrast.

Die Routine versucht bis zu drei Mal ein Bild aufzunehmen, bevor ein Fehler zurückgegeben wird. Dadurch werden kurzzeitige Störungen ignoriert.

A.2 RasMux-Steuerung

In Kapitel 2 wurde schon beschrieben, dass der RasMux über den Befehl `const char *RasCmd(const char *cmd);` gesteuert wird. (In Tcl/Tk mit `RasCmdV1` erreichbar.)

Die Steuerung läuft über Zeichenketten, um den RasMux V2 zu emulieren. Die Befehle des RasMux sind auf Seite 23 aufgelistet. (Alle Zahlen werden mit Hexadezimalen geschrieben.) Über die Datei rascmd.h wird die Steuerung eingebunden. Sie muss durch einen Aufruf von `int RasInit(const char *dev);` initialisiert werden.

An dieser Stelle möchte ich auf die Unterschiede zum RasMux V2 genauer eingehen. Am augenscheinlichsten ist die unterschiedliche Anzahl der RasLed-Anschlüsse. Der RasMux V1 besitzt 12, der neuere 16, wobei jeweils zwei gleichzeitig geschaltet werden. Der V2 schaltet mit Led-Nr. 0 bis 7 jeweils ein Paar ein, und mit Nummer 8 alle RasLeds aus. Ich habe mich dazu entschieden, das gleiche Verhalten auch beim RasMux V1 zu implementieren. Led 8 wurde unter Nummer 15 (`$f`) zugänglich gemacht, die weiteren sind mit 9 bis 11 (`$a`) erreichbar. Dadurch sind alle Ausgänge des V1 benutzbar.

Ein weiterer Unterschied besteht in der Statusabfrage (Befehl **S**). Der RasMux V2 ist in der Lage, den Stromverbrauch zu messen. Weiter kann er feststellen, ob ein Pixeltakt ansteht. Diese Möglichkeiten hat der V1 nicht. Er bietet lediglich ein 8-Bit Statusregister (*JTAG* Instruction Register Nr. `%0100`), das mit diesem Befehl ausgelesen wird. Die Bedeutung der einzelnen Bits kann in [GR99b] nachgelesen werden.

```
struct RasNIN
{
  int
    NIN_SIZE,
    FIN_SIZE,
    VERSION,
    MX,                 /* #Pixel / line */
    MY,                 /* #Lines */
    NCHESS,             /* chess field size */
    LEVEL,              /* debug level */
    IXOK,               /* inv. x */
    IYOK,               /* inv. y */
    TO_CENTER,          /* origin on ccd */
    TO_LENS,            /* 1= fit lens; 0=fit mask */
    ANG_XY,             /* fit angles */
    DEBUG,              /* debug printout */
    BANDS,              /* locate side bands */
    PLOTS,
    FIX_BW_WB,          /* B/W bias */
    _fill[16];          /* set size to 32 */
};
```

Abbildung A.1: *Die Struktur für die Ganzzahl-Parameter.*

Etwas abgewandelt wurde auch die Versionsabfrage (**V**). Hier soll der Unterschied offensichtlich zu Tage treten. Über das *JTAG*-Register IR `%0101` wird das Datum der Firmware ausgelesen. Dieses wird zusammen mit einem Text und der Versionsnummer *meiner* Software ausgegeben, beispielsweise „`99/11/10 - MiniMasterMux Interpreter V1.3`“.

Der Lesebefehl (**R**) für die I^2C-Register liefert, auf Grund eines technischen Problems, das höchstwertigste Bit nicht zurück, doch das wurde schon in Kapitel 2 diskutiert. Wegen dieses Fehlers und der eingeschränkten Statusabfrage macht es auch keinen Sinn, den erweiterten Lesebefehl (**R1**) einzubauen. Der Befehl **U** (total deselect) verhält sich genau wie **O**, da ein weiteres Abschalten beim RasMux V1 nicht möglich ist. Darüber hinaus wurde der Befehl **D** (delay to trigger) nicht implementiert. Dieses Kommando ist dem V1 unbekannt und für den normalen Betrieb überflüssig.

A.3 Einbindung der Nikhef-Analyse

Die Analyse ist in einer Bibliothek zusammengefasst und wird über drei Funktionen aufgerufen (siehe Abbildung 2.4). Mit der Routine `initanalysis_ (int *ver);` wird die Bibliothek initialisiert und die Versionsnummer der Analyse zurückgegeben. Die Funktion `doanalysis_ (struct RasNIN *n_in, struct RasFIN *f_in, char image[], int *err);` führt die eigentliche Analyse durch. In den Strukturen `struct RasNIN` und `struct RasFIN` wer-

```
struct RasFIN
{
  float
    XPIC,                /* x pixel size [mm]*/
    YPIC,                /* y pixel size [mm]*/
    XCCD2,               /* ccd size [mm] */
    YCCD2,               /* ccd size [mm] */
    BAND0,               /* Life size [mm] Band */
    XMSK,                /* Mask Pixel size x */
    YMSK,                /* Mask Pixel size y */
    FOCUS,               /* focal length */
    DIAM,
    WAVE,                /* light wave length */
    DMASK,               /* distance Lens-Mask */
    DCCD,                /* distance Lens-CCD */
    DSYS,                /* distance Mask-CCD */
    DCCD_MASK,           /* DCCD / DMASK */
    _fill[18];           /* set size to 32 */
};
```

Abbildung A.2: *Aufbau der Struktur für die Rationalzahl-Parameter.*

den die Ganz- und Rationalzahlparameter an die Fit-Routine übergeben. Ihre Definitionen sind in Abbildung A.1 und Abbildung A.2 angegeben. Das Feld `char image[]` enthält das Bild, mit einem Byte pro Pixel. Über den letzten Zeiger wird eine Fehlernummer zurückgeliefert.

Die Ergebnisse des Fits werden an zwei Funktionen übergeben, die der Aufrufer exportieren muss. Über `winout_ (char *msg, int length);` werden Informationen für den Benutzer ausgegeben, wie die berechneten Positionen oder Warnungen, z. B. über schlechte Bilder.

Die Funktion `fileout_ (char *msg, int length);` bekommt die Daten für die Log-Datei. Die Positionen weisen hier mehr Stellen auf und es sind weitere Daten wie Unsicherheiten und Zwischenergebnisse für die Fehlersuche enthalten. Leider werden Warnungen nicht an diese Funktion weitergereicht. Icaras schreibt diese Daten in eine Datei, mein Tcl/Tk-Paket kann darüber hinaus einzelne Zeilen in Variablen ablegen.

Die Datenübergabe verläuft, wie bei Fortran üblich ([Bur98]), über einen Zeiger auf den Text und eine Variable für die Länge. Außerdem füllt Fortran die Zeilen mit Leerzeichen auf, weshalb ich alle Leerzeichen am Ende lösche.

Ein Aufruf von `exitanalysis_ ()` gibt schließlich alle Ressourcen der Bibliothek wieder frei. Erst nachdem `initanalysis_ (int *ver);` erneut aufgerufen wurde, kann sie wieder benutzt werden.

Die Verbindung zu Tcl/Tk wird über das Paket `NikAnaTcl` erreicht. Die Analyse wird mit dem Kommando `NikAna` aufgerufen. Die Optionen, die die Fit-Parameter einstellen, sind in Tabelle 2.2 aufgelistet. Zusätzlich können noch die einzelnen Zeilen, die über `fileout_ (char *msg, int length);`

ausgegeben werden, in Variablen abgelegt werden, um sie der Anwendung zu übergeben. Die Optionen heißen **-udvar** für die ud-Zeile, **-uevar** für die ue-Zeile, und so weiter. Die Zeile der Log-Datei, die den Namen und die Zeit enthält, wird nicht von der Analyse geschrieben. Sie muss von der Anwendung erzeugt werden. Das ist auch der Grund, warum an `NikAna` ein Tcl/Tk-Kanal (`-file`) und kein Dateiname übergeben wird. Bei mir wird die Zeile von der Prozedur `IcaStamp` geschrieben, die sich im Paket `NikFile` befindet.

A.4 Rasnik-Betriebsprogramm

Für die Auslese der Rasniks einer Kammer wurde ein Programm entwickelt, welches sich auf einfache Weise erweitern lässt. Dieses Programm (RasApp.tcl) enthält nur die Grundfunktionen zum Analysieren eines Rasniks oder einer Sequenz, zum Schreiben der Log-Datei und zum Definieren und Speichern der Sensoren. Weiter ermöglicht dieses Grundsystem, die Bilder anzuschauen und die Positionen gegen die Zeit zu plotten.

Die Definition einer Rasnik-Strecke wird in einer Tcl/Tk-Liste gespeichert, die den Namen, die Optionen für den RasMux, die Bildaufnahme und die Analyse sowie optional die Null-Position enthält. Mit Hilfe dieser Liste kann ein vollständiger Analysezyklus (Einschalten, Bildnehmen, Analysieren) durchgeführt werden. Wurde zuvor eine Log-Datei bestimmt, so werden alle Daten, wie bei Icaras, darin gespeichert. Optional können weitere Zeilen hinzugefügt werden, z. B. die Definitionsliste, Warnungen über `winout_`, Umgebungsparameter oder der Systemzustand.

Das Grundsystem bietet zahlreiche Ansatzpunkte für Erweiterungen. Es können Menüs eingehängt oder Prozeduren nach jeder Messung oder einem Durchlauf aufgerufen werden.

Im Laufe der Zeit habe ich einige Module geschrieben, die das Grundsystem erweitern. So wurde die Anbindung an die Produktionsdatenbank (siehe Anhang B) realisiert. Damit können die Rasnik-Definitionen direkt aus der Datenbank generiert werden. Dieses Modul unterstützt auch den Einbau der Rasniks. Es werden alle noch nicht eingebauten Linsen aus der Datenbank angezeigt und nach Auswahl einer Linse die nötige Maskenverschiebung berechnet.

Ein weiteres Modul dient der Bestimmung der Null-Positionen. Alle Strecken werden mehrmals gemessen und die Streuung berechnet. Wird sie vom Benutzer akzeptiert, so werden Mittelwert und Standardabweichung zusammen mit den Bildern in der Datenbank gespeichert.

Ebenfalls als Erweiterung programmiert ist die Formanalyse nach Kapitel 4. Es werden automatisch nach jeder vollständigen Sequenz die Winkel und Verschiebungen berechnet. Sie werden in einer gesonderten Log-Datei gespeichert, die sequenz-orientiert ist und somit auch Daten aufnehmen

kann, die mehrere Rasniks betreffen. In diese Datei werden auch die Kugelpositionen gespeichert, die durch die Koordinaten-Messmaschine bestimmt werden.

A.5 Linsenvermessung

Die Messung der Linsenbrennweite erforderte ein spezielles Programm, da neben dem Rasnik auch der Motor für die Maske bedient werden musste. Darüber hinaus waren Routinen erforderlich, die die Schärfe des Bildes messen und anschließend die Position der größten Schärfe fitten.

Dieses Programm wurde dann noch um die Datenbankanbindung erweitert, sodass die vermessenen Linsen in die Datenbank eingetragen werden können. Man gibt dem Programm die Identifikation (Id) und die Sollbrennweite der Linse vor. Dann wird die Messung durchgeführt und es werden an der besten Position die Rasnik-Daten gefittet. Ist der Benutzer mit den Daten zufrieden, so wird die Linse mit allen Messergebnissen in die Datenbank eingetragen.

Außerdem bietet das Programm die Möglichkeit, eine neue Linsenlieferung (Batch) in der Datenbank anzulegen und die Linsen dieser „Batch“ zuzuordnen.

Anhang B

Anpassungen der MMPDB

Während meiner Arbeit mit den BOG-Kammern stellte sich immer wieder heraus, dass die „Myon MDT Produktions-Datenbank“, kurz *MMPDB*, die Besonderheiten unserer Kammern nicht darstellen kann. Daher wurden von mir einige Tabellen hinzugefügt, die in diesem Anhang beschrieben werden. Die ursprüngliche *MMPDB* bleibt dabei als Untermenge enthalten, da keine Tabellen verändert wurden. So ist weiterhin der problemlose Export dieser Daten gewährleistet.

In den folgenden Darstellungen der Tabellen ist in der ersten Zeile der Name und eine kurze Beschreibung der Tabelle angegeben. Die weiteren Zeilen beschreiben jeweils ein Feld. Die erste Spalte gibt den Namen des Feldes wieder, die zweite den Datentyp. Die dritte Spalte gibt an, ob das Feld leer sein darf (Y) oder nicht (N). In der letzten Spalte sind Besonderheiten eines Feldes vermerkt, wie Referenzen oder ob es sich um den Hauptschlüssel (Primary Key) handelt.

In der ursprünglichen Datenbank werden jeder Kammer genau vier Rasniks zugeordnet. Damit lässt sich die Situation der BOG-Kammern nicht abbilden. Ich habe daher eine Tabelle definiert, die eine einzelne Rasnik-Strecke beschreibt. Sie enthält eine Menge Informationen über dieses Rasnik und einen Verweis auf die Kammer, in die es eingebaut ist. Die Struktur der Tabelle ist in Tabelle B.1 wiedergegeben.

Bei dem ersten Feld (RASID) handelt es sich um eine numerische Identifikation. Das nächste beschreibt den Namen des Rasniks, dieser sollte pro Kammer eindeutig sein. Dann kommt der Verweis auf die Kammer. Es folgen dann die Parameter für den Fit, die verwendeten RasMux-Anschlüsse sowie ein Verweis auf die Linse (s.u.). Als letztes können noch die Null-Positionen gespeichert werden.

Ich hatte bereits darauf hingewiesen, dass es wichtig ist, die Bilder der Null-Position zu speichern. In der Tabelle B.2 kann jeweils ein Bild abgelegt

RASNIKS	**TABLE**		**Describes a single Rasnik system on a chamber**
RASID	NUMBER(6,0)	N	-= PK =-
NAME	VARCHAR2(20)	N	
IDCHAM	VARCHAR2(20)	Y	->CHAMBER
DCCD	FLOAT(126)	N	
DMSK	FLOAT(126)	N	
LENSID	NUMBER(6,3)	Y	->LENSES
MSKPIT	FLOAT(126)	N	
XPIX	FLOAT(126)	N	
YPIX	FLOAT(126)	N	
MSKX	FLOAT(126)	Y	
MSKY	FLOAT(126)	Y	
LED	NUMBER(4,0)	N	
CAM	NUMBER(4,0)	N	
OPTIONS	VARCHAR2(2048)	Y	
LEDLOC	NUMBER(2,0)	N	
CAMLOC	NUMBER(2,0)	N	
ANGLE	FLOAT(126)	Y	
X0	FLOAT(126)	Y	
Y0	FLOAT(126)	Y	
Z0	FLOAT(126)	Y	
AX0	FLOAT(126)	Y	
AY0	FLOAT(126)	Y	
AZ0	FLOAT(126)	Y	

Tabelle B.1: Beschreibt ein einzelnes Rasnik.

RASIMAGE	**TABLE**		**Rasnik Image and/or fit result**
IMGID	NUMBER(6,0)	N	-= PK =-
NAME	VARCHAR2(20)	N	
RASID	NUMBER(6,0)	Y	->RASNIKS
X	FLOAT(126)	Y	
Y	FLOAT(126)	Y	
Z	FLOAT(126)	Y	
AX	FLOAT(126)	Y	
AY	FLOAT(126)	Y	
AZ	FLOAT(126)	Y	
LOG	VARCHAR2(2048)	Y	
NOTE	VARCHAR2(1024)	Y	
IMGDATE	DATE(7)	Y	
FORMAT	VARCHAR2(20)	Y	
IMAGE	BLOB(4000)	Y	

Tabelle B.2: Diese Tabelle speichert die Rasnik-Bilder.

RASORIGIN	**TABLE**		**Images used for calc. of origin**
RASID	NUMBER(6,0)	N	->RASNIKS
IMGID	NUMBER(6,0)	N	->RASIMAGE
ORGDATE	DATE(7)	Y	
NOTE	VARCHAR2(1024)	Y	

Tabelle B.3: Bestimmt, welche Bilder die Null-Position definieren.

werden. Neben den Bilddaten (IMAGE) kann sie die ermittelten Positionen und einen Auszug aus der Log-Datei (LOG) aufnehmen.

Es kam irgendwann der Wunsch auf, mehr Bilder in der Datenbank zu speichern als für die Null-Position nötig sind. Um die richtigen Bilder wiederzufinden, wurde eine weitere Tabelle eingeführt, die nur auf die Bilder der aktuellen Kalibration verweist. Wie in Tabelle B.3 zu sehen ist, ist diese Tabelle recht klein und enthält im Wesentlichen nur die Verweise auf das Rasnik und die Bilder.

Die Datenbank bietet sich auch dafür an, die Daten der Linsen aufzunehmen (siehe Tabelle B.4). Die wichtigste Größe der Linse ist die Brennweite, sie wird zum einen über die Abstände und zum anderen über die Vergrößerung bestimmt. Dann werden noch das Datum und die Temperatur während der Messung gespeichert. Dazu kommen die Fit-Ergebnisse und die Schleifradien der Linse. Das ergibt einen vollständigen Datensatz, der, auch nach dem Einbau, Hinweise auf die Qualität der Linse gibt.

LENSES	**TABLE**		**Description of a single lens for Rasnik**
LENSID	NUMBER(6,3)	N	-= PK =-
LENSBAT	NUMBER(3,0)	N	->LENSBAT
FNOM	FLOAT(126)	N	
FSHRP	FLOAT(126)	Y	
FMAG	FLOAT(126)	Y	
OFFSET	FLOAT(126)	Y	
MEASDATE	DATE(7)	Y	
TEMP	FLOAT(126)	Y	
XRAS	FLOAT(126)	Y	
YRAS	FLOAT(126)	Y	
MRAS	FLOAT(126)	Y	
DMSK	FLOAT(126)	Y	
DCCD	FLOAT(126)	Y	
GWIDTH	FLOAT(126)	Y	
FITRMS	FLOAT(126)	Y	
ANGLE	FLOAT(126)	Y	
RAD1	FLOAT(126)	Y	
RAD2	FLOAT(126)	Y	

Tabelle B.4: In dieser Tabelle werden alle Daten der Linsen gespeichert.

Wir bekamen die Linsen in mehreren Lieferungen. Ich habe daher, analog zu den Röhren, eine Tabelle für die Linsen-Lieferung (siehe Tabelle B.5) definiert. Sie enthält Informationen über die Produktionsstätte und über Datum und Umfang der Lieferung.

Damit sind nun alle Tabellen vorgestellt, die ich zur Beschreibung der Rasniks in den BOG-Kammern hinzugefügt habe. Diese Tabellen sind universell gehalten und lassen sich auch bei anderen Kammertypen verwenden.

LENSBAT	**TABLE**		**A Batch of Lenses**
IDLENSBAT	NUMBER(3,0)	N	-= PK =-
LENSBATDT	DATE(7)	Y	
LENSBATSITE	VARCHAR2(3)	Y	->SITECODE
LENSBATPROD	VARCHAR2(20)	Y	
LENSBATNP	NUMBER(5,0)	Y	
LENSBATREJNP	NUMBER(5,0)	Y	

Tabelle B.5: Diese Tabelle nimmt Informationen über die Lieferung der Linsen auf.

Literaturverzeichnis

[ASS00] ASSET InterTech, Inc: *Boundary-Scan Tutorial*. http://www.asset-intertech.com/login.html?doc=/PDFs/boundaryscan_tutorial.pdf, 2000.

[ATL94] ATLAS Collaboration: *ATLAS: Technical Proposal*. Dezember 1994. CERN-LHCC-94-43.

[ATL97] ATLAS Muon Collaboration: *ATLAS Muon Spectrometer - Technical Design Report*. http://atlasinfo.cern.ch/Atlas/GROUPS/MUON/TDR/pdf_final/mTDR.pdf, Juni 1997.

[ATL99a] ATLAS Collaboration: *ATLAS: Technical Design Report*. Vol. 1, Mai 1999. CERN-LHCC-99-14.

[ATL99b] ATLAS Collaboration: *ATLAS: Technical Design Report*. Vol. 2, Mai 1999. CERN-LHCC-99-15.

[Bar01] *The Barrel Alignment System: Principle, Requirements and Implementation of the Subsystems*. http://atlasinfo.cern.ch/Atlas/GROUPS/MUON/alignment/documents/Barrel_description_140301.pdf, März 2001.

[Bur98] Burley, James Craig: *Using GNU Fortran*. http://gcc.gnu.org, 1998.

[Cli03] Cline, David B.: *Die Suche nach Dunkler Materie*. Spektrum der Wissenschaft, (10/03):44–51, Oktober 2003.

[Cox00] Cox, Alan: *Video4Linux Programming*. http://kernelbook.sourceforge.net/videobook.pdf, 2000.

[Dat95] Data Translation, Inc.: *Engineering Specification for the DT3152*, August 1995.

[GR99a] Groenstege, H. und P.A.M. Rewiersma: *RasCaM, The sensor for the RasNik alignment system.*

http://www.nikhef.nl/pub/departments/et/experiments/atlas/rasnik/atlas_rasnik/rascam/rascam.pdf, Amsterdam, NL, September 1999.

[GR99b] Groenstege, H. und P.A.M. Rewiersma: *RasMux*. http://www.nikhef.nl/pub/departments/et/experiments/atlas/rasnik/proto_rasnik/rasmux_v1.pdf, Amsterdam, NL, November 1999.

[GR02a] Groenstege, H. und P.A.M. Rewiersma: *MiniMasterMux*. http://www.nikhef.nl/pub/departments/et/experiments/atlas/rasnik/atlas_rasnik/minimastermux/minimastermux.pdf, Amsterdam, NL, Mai 2002.

[GR02b] Groenstege, H. und P.A.M. Rewiersma: *RasLed*. http:// www.nikhef.nl/pub/departments/et/experiments/atlas/rasnik/atlas_rasnik/rasled/rasled.pdf, Amsterdam, NL, August 2002.

[Gro94] Groenstege, H.: *The coding of the mask for CCD_Rasnik*. http://www.nikhef.nl/pub/departments/et/experiments/atlas/rasnik/atlas_rasnik/rasled/code/code.pdf, Amsterdam, NL, 1994.

[Hig64] Higgs, Peter W.: *BROKEN SYMMETRIES AND THE MASSES OF GAUGE BOSONS*. Phys. Rev. Lett., 13:508–509, 1964.

[K. 03] K. Hagiwara et al.: *Physical Review D – Particles and Fields*, Band 66. Particle Data Group, 2003. partial update for edition 2004 (URL: http://pdg.lbl.gov).

[Kam95] Kammer, Dr. Catrin: *Aluminium – Taschenbuch*. Aluminium-Verlag, Düsseldorf, 15. Auflage, 1995.

[Leo94] Leo, William R.: *Techniques for Nuclear and Particle Physics Experiments*. Springer-Verlag, New York, USA, 2. Auflage, 1994.

[LoI92] *ATLAS: Letter of intent for a general purpose p p experiment at the large hadron collider at CERN*. Oktober 1992. CERN-LHCC-92-4.

[Phi00] Philips Semiconductors: *The I^2C-Bus Specification*, Januar 2000. Document order number: 9398 393 40011.

[Sch97] Schott: *RG 830*, Juni 1997. Product Datasheet.

[TS93] Tietze, Ulrich und Christoph Schenk: *Halbleiter-Schaltungstechnik*. Springer-Verlag, Berlin, 10. Auflage, Januar 1993.

[vdGGLR00] Graaf, H. van der, H. Groenstege, F. Linde und P. Rewiersma: *RasNiK, an Alignment System for the ATLAS MDT Barrel Muon Chambers*. http://www.nikhef.nl/pub/departments/et/experiments/atlas/rasnik/atlas_rasnik/rasnik_rev2.pdf, Amsterdam, NL, April 2000.

[vE99] Es, J.T. van: *NTC-Temp.sensor Scanner and Conditioner 990203 circuit.* http://www.nikhef.nl/~hansve/ATLAS/Temperatuurmeter990203.pdf, Amsterdam, NL, Oktober 1999.

[VLS98] VLSI VISION Limited, Edinburgh, UK: *VISION VV5430 Monolithic Sensor*, März 1998. Product Datasheet.

[Wau00] Waugh, Tim: *The Linux 2.4 Parallel Port Subsystem*. http://kernelbook.sourceforge.net/parportbook.pdf, 2000.