BACHELOR THESIS

Entwicklung eines Strahlkamerasystems zur Analyse von Ionenstrahlen

Development of a Beam Camera System for the Analysis of Ion Beams

Julian Bergmann
geboren am 25.09.1989 in Gießen, Hessen

25. Oktober 2012

Justus–Liebig–Universität Gießen
II. Physikalisches Institut

Erstgutachter: Prof. Dr. Christoph Scheidenberger
Zweitgutachter: Prof. Dr. Hans Geissel

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Hilfsmittel ausgeführt habe.

(Gießen, den 25. Oktober 2012)

Inhaltsverzeichnis

1 Motivation **1**

2 Grundlagen **2**
2.1 Flugzeit–Massenspektrometer 2
2.2 Ambiprobe: MR–TOF–MS 4
2.3 Peltierelement–Kühlung 7
2.4 Funktionsweise der Ionenstrahlkamera 7
2.4.1 Übersicht 7
2.4.2 MCP 9
2.4.3 Phosphorschirm 10
2.4.4 CCD–Kamera/Chip 11

3 Aufbau der Kamera und Softwareentwicklung **15**
3.1 Strahlkamera 15
3.2 CCD–Kamera 15
3.3 Versuchsaufbau 18
3.4 Datenaufnahme–Software 19
3.4.1 Auswahl der Programmiersprache 19
3.4.2 Grundlagen zur Programmoptimierung 20
3.4.3 Erkennen eines Signals 21
3.4.4 Bildverbesserung 23
3.4.5 Ziehen einer Linie nach Bresenheim 24

4 Messung und Ergebnisse **27**
4.1 Die Kamera im Vakuum 27
4.2 Ionenstrahl–Analyse 28

5 Zusammenfassung und Ausblick **32**

A Literaturverzeichnis **34**

B Datenblätter **36**

C Bildschirmabbilder zu Programm–Funktionen **38**

D Quelltext **42**

E Programm–Anleitung **48**

1. Motivation

Arbeitet man an Experimenten, in denen Strahlen von geladenen Teilchen vorkommen, lassen sich Fehler in der Ausrichtung des Strahls oft erst durch Abweichungen von erwarteten Ergebnissen nach der Auswertung der erhaltenen Daten erkennen.

Daher ist es Ziel dieser Arbeit, ein Strahlkamerasystem zu entwickeln, genauer einen Strahlkamera–Detektor–Aufbau (MCP, Phosphorschirm und CCD–Kamera). Dabei sollte insbesondere ein Computer–Programm entwickelt werden, um eine Echtzeitanalyse der Strahleigenschaften zu erstellen und anzuzeigen. Außerdem wird im Strahlkamera–Aufbau ein neuer, digital ansteuer– und auslesbarer CCD-Sensor verwendet.

Dieser kompakte Aufbau macht das System in beliebigen Vakuum-Aufbauten einsetzbar. Da die Sensoreinstellungen dabei ansteuerbar sind, lässt sich die Kamera–Kallibrierung auch bei bereits bestehendem Vakuum verändern. Das entwickelte Computer–Programm übernimmt dabei nicht nur die Kamerasteuerung, sondern rechnet auch zu den gemessenen Ereignissen Helligkeits-, Breiten- und Positionsangaben aus, um so die Strahlposition und -Intensität erkenntlich zu machen.

Im Rahmen dieser Bachelor–Arbeit wurde die Strahlkamera dabei in einem Test–Aufbau benutzt. Später wird sie in einem MR–TOF–MS (Multireflektions–Time–Of–Flight–Massenspektrometer) eingesetzt werden, welches auch in Medizin oder Umwelt eingesetzt wird (AmbiProbe Projekt, siehe hierzu Kapitel 2.2). Dort soll die Strahlkamera dabei helfen Flugzeitfehler minimieren, indem Strahlform und –position bestimmt wird und eine entsprechende Optimierung erfolgen kann.

Die Bachelor–Arbeit umfasst dabei Grundlagen zum verwendeten Detektoraufbau und Massenspektrometer, Erklärungen zu den verwendeten Algorithmen und schließlich den Einsatz des Programms und der neuen Kamera an einem experimentellen Aufbau. Im Anhang befindet sich zusätzlich das Programm sowie eine Anleitung zur Bedienung.

2. Grundlagen

2.1. Flugzeit–Massenspektrometer

Da das Gerät später speziell an einem Massenspektrometer eingesetzt werden soll, wird nun dessen Funktionsweise genauer behandelt werden.

Bei dem sogenannten Flugzeit–MS (Massenspektrometer), wie es schematisch in Abbildung 1 dargestellt ist, macht man sich das Prinzip der Massenträgheit zunutze. Die zu untersuchenden Teilchen werden dabei zunächst ionisiert, um dann in einem Potential beschleunigt zu werden. Die Unterschiede in der Geschwindigkeit bedeuten dabei unterschiedliche Massen, da die kin. Energie gleich ist[PDE04].

Diese Unterschiede zeichnen sich in der benötigten Zeit zur Durchquerung einer festgelegten Strecke ab. Das Start–Signal ist dabei die gepulste Extraktion, das Stopp–Signal die Messung am Detektor. Der Zusammenhang der Masse mit der Geschwindigkeit ergibt sich dann aus der Integration der Geschwindigkeit aus dem Anfangspotentials über die Flugstrecke, was die Flugzeit ergibt. Dadurch erhält man den Zusammenhang von $m = aT^2$ mit a als Proportionalitätsfaktor.

Ionisiert werden die Teilchen z.B. mittels Elektronenstoßionisation, bei der Elektronen über Glühemission mit 0,2 eV erzeugt und dann auf 70 eV beschleunigt werden, um dann weitere Elektronen aus den getroffenen Teilchen herauszuschlagen. Dabei kann zwar auch mehrfache Ionisierung oder Fragmentierung auftreten, allerdings ist dieses Verfahren mit vielen Einlass–Systemen kompatibel und mit vielen Stoffen bereits gut dokumentiert worden.

Benutzt man dabei eine Ionenquelle wie in Abbildung 2 nach Wiley und McLaren[WM55], werden die ionisierten Teilchen zwischen zwei Elektroden gleichen Potentials (U_1) produziert. Bei der Extraktion wird das Potential der 2. Elektrode genau soweit verringert, dass weiter entfernt liegende Ionen genau soviel stärker beschleunigt werden, dass die unterschiedlichen Startorte beim Eintreffen am Detektor kompensiert werden. Dabei setzt man die 2. Elektrode auf U_2 und erhält so ein elektrisches Feld E_1 zwischen den Elektroden und das Feld E_2 hinter der 2. Elektrode.

Um bei der Flugzeitmessung die Genauigkeit erhöhen zu können, wird dabei oft ein oder mehrere sogenannter Reflektoren eingesetzt (siehe dazu Abbildung 4), die die Bahnrichtung umkehren und so unter guter Ausnutzung des zur Verfügung stehenden Platzes die Länge der Flugbahn vervielfachen können. Dabei werden oft auch mehrstufige Reflektoren benutzt, bei denen das Potential in mehreren Stufen jeweils linear steigt. Dies kann dafür benutzt werden, Anfangsenergie–Unschärfen zu kompensieren. Dabei trifft das Ion, welches bereits l_{D1} zurückgelegt hat, auf den Reflektor und tritt in ein linear steigendes

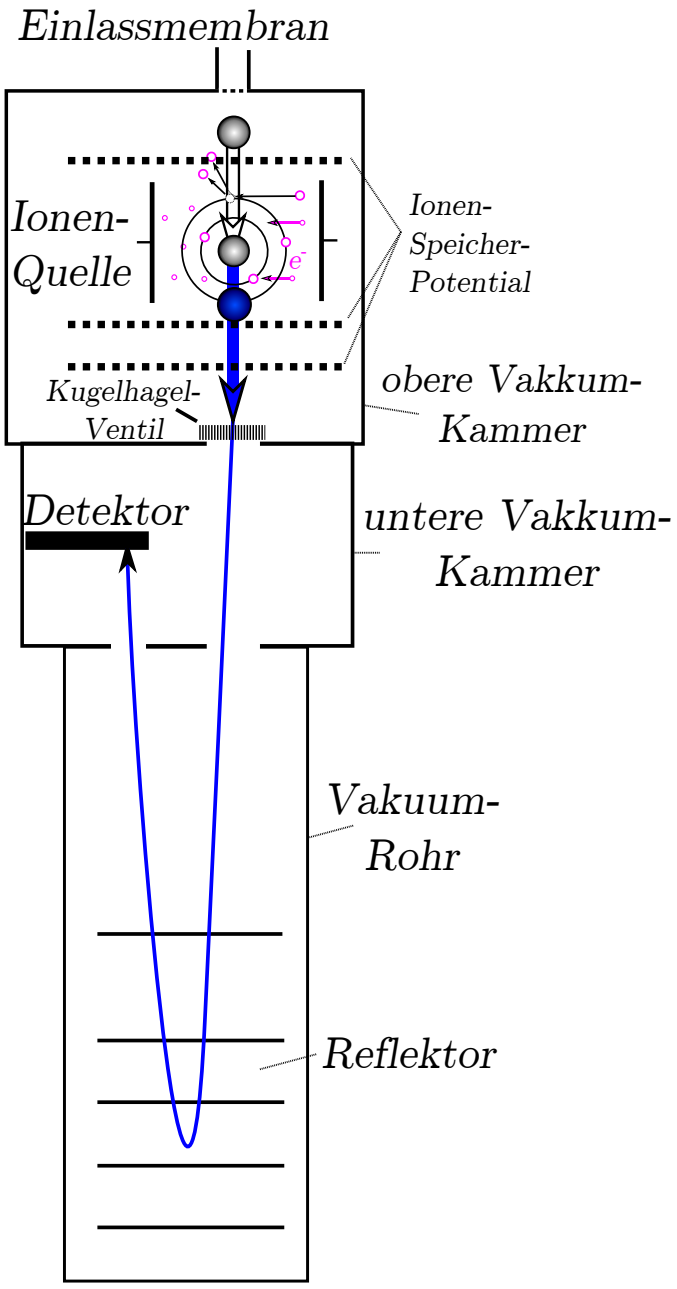


Abbildung 1
Schema eines Aufbaus des Flugzeitmassenspektrometers

Potential ein (zugehöriges Feld E_1), sodass die Flugrichtung nach einer gewissen Strecke (l_1) umgekehrt wird. Schneller fliegende Ionen durchlaufen dabei eine längere Strecke (l_2), wodurch sie idealerweise genau soviel Zeit mehr benötigen, wie sie durch ihre höhere Geschwindigkeit den anderen Ionen gegenüber früher am Detektor (nach der weiteren Strecke l_{D2}) wären[PDE04].

Messungen mit Massenspektrometern zeigen jedoch, dass Abweichungen von dieser idealen Flugbahn zu nicht unerheblichen Fehlern in der Flugzeit und damit der Massenbestimmung führen. Abbildung 3 zeigt dabei ein solches Massenspektrum. Während die Haupt–Peaks die tatsächlichen Massen angeben, sind auch kleinere Peaks bei kleineren und größeren Massen erkennbar. Die Peaks stellvertretend für höhere Massen entstehen dabei durch Kollisionen mit dem umgebenden Rest–Gas, die Peaks der leichteren Massen jedoch durch Abbildungsfehler. Diese könnten mittels Kenntnis der Strahlposition und des –durchmessers korrigiert werden, was mithilfe der Strahlkamera geschieht[Dic10].

2.2. Ambiprobe: MR–TOF–MS

Eine bereits genannte Motivation dieser Arbeit ist der LOEWE-Schwerpunkt *AmbiProbe*. Hierbei ist es Ziel, neue Massenspektrometer zu entwickeln, um In–situ–Analysen, speziell im Bereich der Umwelt–, Gesundheits– und Klimaforschung, vornehmen zu können.

Dabei wurde ein hochauflösendes MR–TOF–MS (*Multi–Reflektions–Time–of–Flight–Massenspektrometer*, siehe auch Abbildung 5) entwickelt, welches beispielsweise in der Elektrochirurgie oder der Luftanalyse Verwendung finden wird. Es ist dabei sehr robust und kompakt und daher auch mobil. Enthalten sind dabei ein Atmosphärendruck–Messgerät (*atmospheric pressure interface*, API), RFQ–Massenfilter, Ionenkühlung und –Falle und Flugzeit–Analysator sowie –Detektor. Mit Steuerungssystem ist der Aufbau dabei nur etwa 0,8 m^3 groß[P^+12, L^+12].

Da es sich hier also um einen sehr kleinen Aufbau handelt, sind die räumlichen Möglichkeiten Abbildungsfehler zu korrigieren, limitiert. Umso wichtiger ist es daher, dass die Strahlkamera (und das entwickelte Programm) Strahlposition und –durchmesser zur Flugzeitkorrektur ermitteln kann.

Damit der Analysator außerdem besonders hoch auflöst, wird er so betrieben, dass er pro Umlauf 180 Grad Phasenvorschub hat. Dies lässt sich mit Bestimmung von Strahlposition und –durchmesser gut messen und somit auch der Auflösungsmodus optimieren.

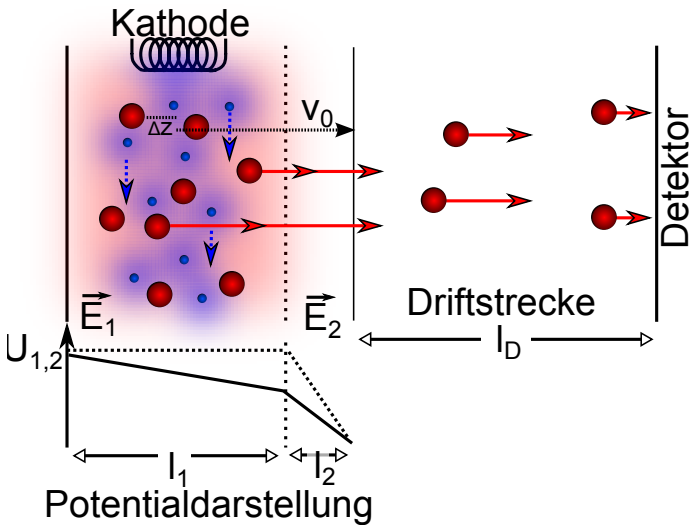


Abbildung 2
2-Stufige Ionenquelle nach Wiley und McLaren

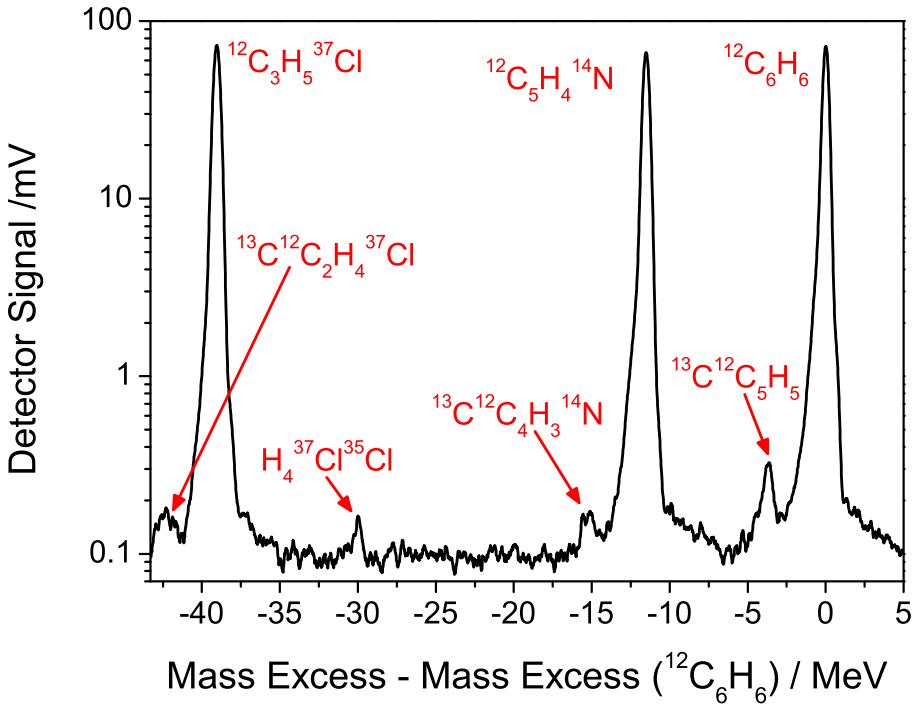


Abbildung 3
Logarithmisch aufgetragenes Massenspektrum von Cs^+–Ionen nach 210 Durchläufen, gemittelt über 20 Datenpunkte[Dic10].

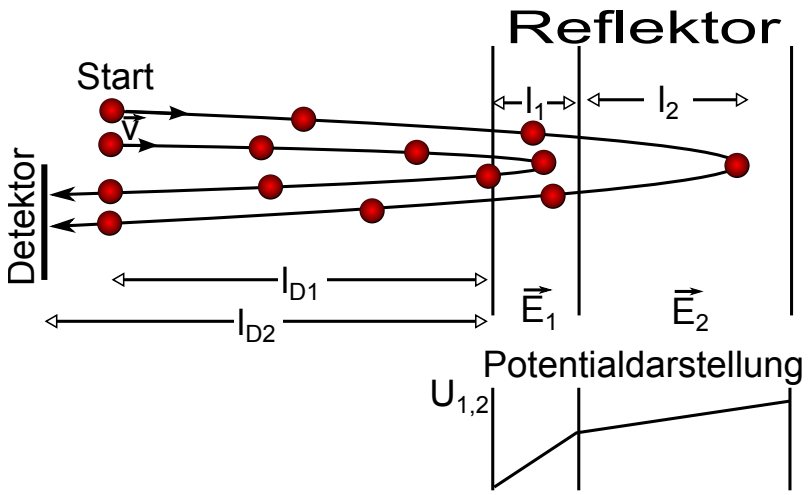


Abbildung 4
2-Stufiger Reflektor

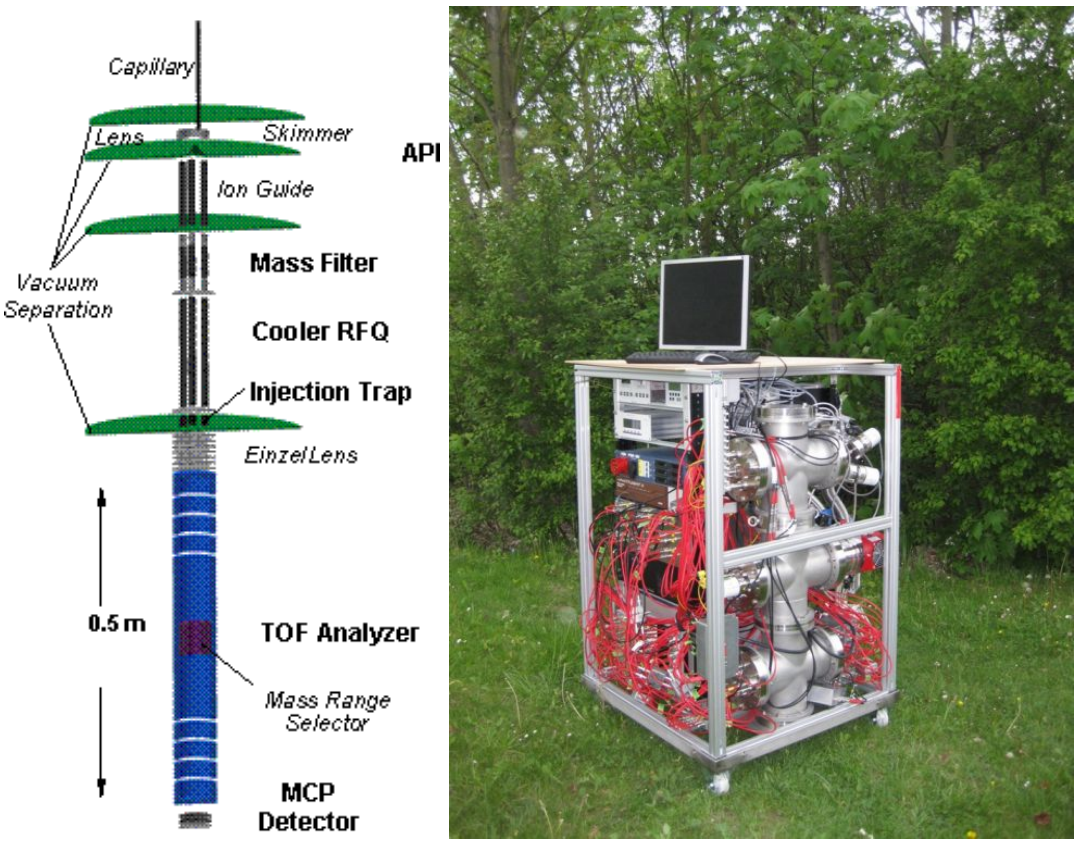


Abbildung 5
Aufbau des MR–TOF–MS im AmbiProbe-Projekt (links schematisch, rechts Photo), Bild aus [L$^+$12]

2.3. Peltierelement–Kühlung

Um den Aufbau (später das Massenspektrometer) zu optimieren, ist es notwendig, eine zusätzliche Kühlung anzubringen. Diese sorgt nämlich nicht nur für gleichbleibende Bedingungen im Vakuum und bessere Vakuum–Eigenschaften, sondern führt vor allem auch zu weniger störanfälligen Elektronik. Speziell bei Kameras erzeugen zu hohe Temperaturen oft ungewollte Bildfehler. Natürlich ist auch das Material der meisten Elektronik-Teile nicht unbegrenzt temperaturstabil.

Speziell in Bezug auf die im später beschriebenen Aufbau benutzte Kamera kann mit dieser Kühlung durch wärme verursachte Bildfehler (beispielsweise helle Streifen, siehe dazu auch Kapitel4.2) vorgebeugt werden.

Daher hat man hier eine Peltierelement–Kühlung entwickelt [Pet11], bei der außerhalb des Vakuums am Flansch Peltier–Elemente angebracht werden, die man innen mit den zu kühlenden Quadrupolen bzw. der Strahlkamera verbindet. Dabei geschieht die Durchführung über einen massiven Kupferstab, welcher innen in Kupfernetze endet. Die Wärmeabgabe erfolgt über externen Lüfter (Abbildung 6).

Während diese Kühlung sehr flexibel einsetzbar und kostengünstig ist, hat sie auch einige Probleme. So bringen Peltier-Elemente unter Volllast auch einige Eigenwärme mit ins Vakuum. Auch ist die Wärmeleitung innerhalb des Vakuums etwa bei Kupferbändern bedenklich. Diese beiden Probleme können jedoch gut mit leistungsstarken Lüftern — hier wird Crosair H100, eine autarke Wasserkühlung, verwendet — behoben werden.

2.4. Funktionsweise der Ionenstrahlkamera

2.4.1. Übersicht

Um einen Teilchenstrahl analysieren zu können, sollte dieser zunächst detektiert und in ein Signal umgewandelt werden, welches mit dem entwickelten Programm ausgewertet werden kann.
Dabei bezeichnen wir mit dem *Detektor* (nach [Ebe09]) die, wie in Abbildung 7 zu sehen, hintereinander geschalteten Elemente

- Micro–Channel Plate (MCP)
- Phosphorschirm
- Kamera/CCD–Chip

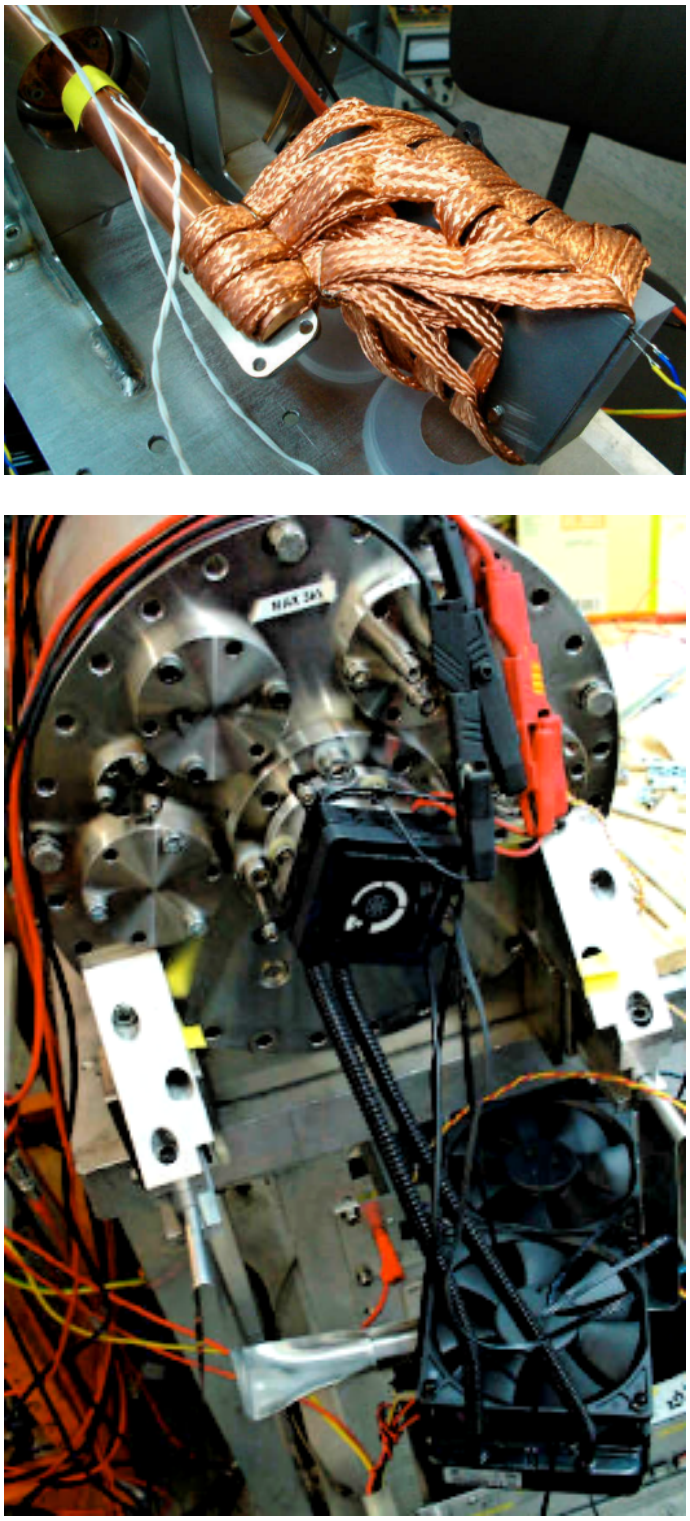


Abbildung 6
Peltiere–Element–Kühlung, hier mit autark Wassergekühltem Lüftersystem.
Oben: Aufbau innen mit Kupfernetz und Kupferstabdurchführung
Unten: Anschluss außen mit Lüftern

Die *MCP*s werden dabei benötigt, um die ankommenden Signale für die Empfindlichkeit der Kamera ausreichend zu verstärken.

Der *Phosphorschirm* wird benutzt, da der CCD–Chip darauf ausgelegt ist, Photonen zu detektieren, unser Strahl aber aus geladenen Teilchen besteht. Der Schirm kann dabei auftreffende geladene Teilchen in Lichtblitze umwandeln.

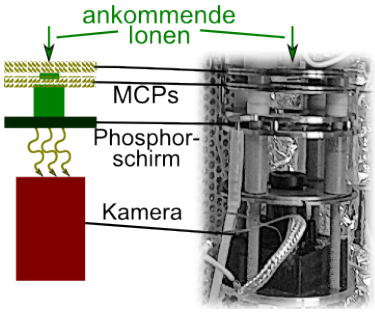


Abbildung 7
Schema des Detektor–Aufbaus

2.4.2. MCP

Eine sogenannte *Micro–Channel Plate* (*MCP*) ist ein Sekundärelektronenverfielfacher. Die ankommende Ladung schlägt Sekundärelektronen aus der Oberfläche wodurch die detektierbare Ladung vermehrt wird.

Im Falle des MCP ist dies jedoch nicht einfach eine Platte, sondern eine Anordnung von 10^6 bis 10^7 einzelnen Bleiglas–Röhrchen von etwa 5-15 µm Durchmesser und ungefähr 0,5 mm lang, wie in Abbildung 8 dargestellt. Diese sind geneigt (15°) in eine Platte eingesetzt, sodass keine geladene Teilchen ohne Kollision hindurch fliegen können. Außerdem liegt entlang dieser Röhrchen ein Spannungsabfall (ca. 1 kV) an, sodass die Sekundär–Elektronen nach dem Austritt im Potential erneut beschleunigt werden und wieder mit dem Bleiglas kollidieren. Die ankommende Ladung kann so um einen Faktor 10^3 erhöht werden.

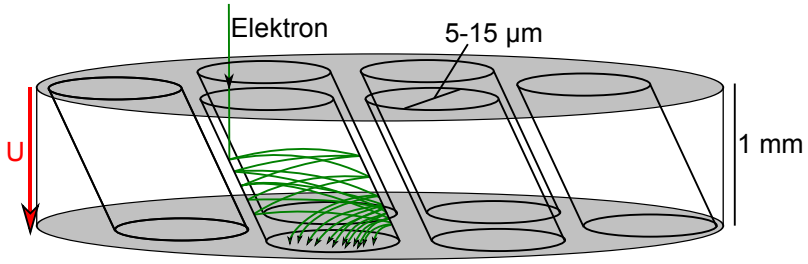


Abbildung 8
Stark vereinfachte Darstellung einer MCP

Für MCPs sind allerdings auch Sonderfälle zu berücksichtigen: die sogenannte *Ionenrückkopplung* und die *Sättigung der Kanäle*.

Gelangen positiv geladene Ionen an den Ausgang des MCPs, werden sie von dem Potential angezogen und schlagen innerhalb der Röhren Elektronen aus. Dies führt zur Detektion unerwünschten Ereignisse, da es sich nicht um Reaktionen auf das eigentliche Signal handelt.

Dieses Phänomen kann durch die sogenannte *Chevron–Anordnung* unterbunden werden. Dabei wird eine zweite MCP um 180° verdreht an die erste angefügt. Rückläufige Ionen können so nicht mehr die ganze sondern höchstens die halbe Höhe der Anordnung durchfliegen, bevor sie Elektronen ausschlagen. Dies macht die daraus resultierenden Schauer am Ausgang wesentlich schwächer als jene der erwünschten Signale.

Was nicht so einfach behoben werden kann ist die Sättigung der einzelnen Kanäle. Da sich diese im Prinzip wie Kondensatoren verhalten tritt kurz nach einer streng lokalisierten Entladung durch das Aussenden von Sekundärelektronen zunächst ein Ladungsloch auf, an dem ankommende Elektronen nur wenig bis keine Sekundärelektronen auslösen können. Da die Verstärkung kaskadenartig erfolgt ist das zwar bis zu einem bestimmten Teilchenstrom kein Problem, ab einer bestimmten Teilchenstromdichte jedoch findet durch diese Lücken keine tatsächliche Verstärkung mehr statt, weswegen man auch von einer Sättigung der Kanalplatten spricht.

Bei einer Benutzung außerhalb der Sättigungsstromdichte und Teilchen mit Energien von 2-50 keV ergibt sich eine *Detektionseffizienz* bzw. *Empfindlichkeit* von 60-85%[Wiz79].

2.4.3. Phosphorschirm

Die Aufgabe eines *Phosphorschirmes* ist es, ankommende Ladungen bzw. Teilchen zu absorbieren und der Energie entsprechend Photonen zu emittieren. Dabei besteht dieser aus einem leitenden, festen Material, welches mit einem phosphoreszierenden Stoff beschichtet wurde. Meist ist diesem Schirm auch ein Potential vorgeschaltet, um die hier ankommenden Elektronen genügend zu beschleunigen, sodass genügend Energie für Lichtblitze vorhanden ist.

Unter *phosphoreszierenden* Stoffen bzw. *Phosphoreszenz* versteht man dabei einen Sonderfall der *Szintillation*. Da wir in unserem Fall eine Kamera zur Detektion benutzen, ist der Bereich des sichtbaren Lichtes hier besonders interessant. Szintillation in diesem Bereich wird auch *Lumeneszenz* genannt. Szintillation bezeichnet allgemein die Anregung von Stoffen über ionisierende Strahlung und die Abregung über elektromagnetische Strahlung. Sie unterteilt sich dabei in die sogenannte *Fluoreszenz* und die *Phosphoreszenz*.

Bei der wesentlich häufiger vorkommenden *Fluoreszenz* fallen die angeregten Moleküle spinerhaltend unter Emission von Photonen in den Grundzustand zurück. Dabei beträgt die Halbwertszeit höchstens einige 10^{-7} s, der Effekt ist also zwar im Vergleich zum Ereignis mit $10^{-9} - 10^{-10}$ s noch relativ lang, allerdings wesentlich kürzer als die Halbwertszeit bei Phosphoreszenz.

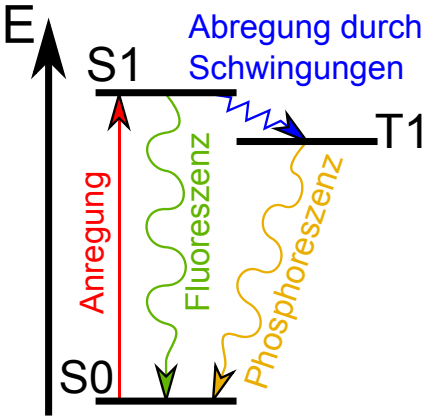


Abbildung 9
Jablonski–Termschema:
S0 als Grundzustand und S1 als erster angeregter Zustand

Die Abregung in den energetischen Grundzustand erfolgt bei der *Phosphoreszenz* im Gegensatz zur *Fluoreszenz* nicht spinerhaltend und ist damit unterdrückt, kommt also seltener vor. Es werden also Übergänge von angeregten Singulett–Zuständen in niedrigere aber dennoch angeregte Triplett–Zustände vollzogen, da hier $\Delta S \neq 0$ erlaubt ist, wie in Abbildung 9 gezeigt. Dieser Unterschied äußert sich vor allem in der Zeitdauer der Emission und der Höhe der emittierten Energie.

Die unterschiedliche Zeitdauer bis zur Emission der Anregung kommt dadurch zustande, dass der angeregte Triplett–Zustand auch wieder nur nicht–spinerhaltend in den Grundzustand fallen kann (hier muss $\Delta S \neq 0$). Dadurch kommt eine Halbwertszeit von bis zu 10^{-3} s zustande[BG94, Gue08].

2.4.4. CCD–Kamera/Chip

Bei dem sogenannten *CCD–Sensor* (*Charged Coupled Device*) handelt es sich um einzelne, in eine Zeile geschaltete oder zu einer Fläche zusammengesetzte Schaltung von *MOS–Kondensatoren* (*Metall Oxide Semiconductor*). Die Oberfläche des MOS–Kondensators ist meist eine mit SiO_2 beschichtete, p–dotierte Silizium–Platte von ca. 20 µm. Oberhalb der Beschichtung befindet sich eine positiv geladene Elektrode, die einen Potentialtopf im Silizium schafft. Unterhalb des Siliziums wird flächig eine negativ geladene Elektrode angebracht. Auftreffende Photonen können nun Elektron–Loch–Paare auf der Oberfläche auslösen, von denen die Elektronen zum Potentialtopf wandern während die „Löcher“ von der negativen Elektrode eingefangen werden. Die SiO_2-Schicht verhindert dabei das Einfangen der Elektronen durch die positive Elektrode, sodass sich hier wie in einem Kondensator eine elektrische Ladung entsprechend dem einfallenen Licht ansammelt (Abbildung 10).

Neben dem *surface–channel*–Typ, bei dem die Ladungen direkt unter der SiO_2-Schicht gesammelt werden, existiert auch ein verbesserter Aufbau, der

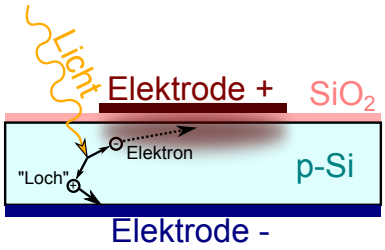


Abbildung 10
MOS–Kondensator–Prinzip

buried–channel-Typ. Bei diesem wird eine weitere Schicht gegensätzlicher Dotierung unter die SiO_2-Schicht gelegt, sodass die Ladungen sich innerhalb des Siliziums sammeln. Dies vermeidet vor allem Störungen durch Oberflächen–Kristallstörungen und verbessert das Rauschverhalten des Sensors, macht diesen aber auch teurer.

Zum Auslesen fungieren die MOS–Elemente als Schieberegister. Gewöhnlich wird eine gewisse Belichtungszeit gewartet, bis der CCD–Sensor aufgefordert wird, das Bild auszugeben. Dann wird die Ladung des ersten MOS–Elementes an den Computer gesendet während die dahinter liegenden Ladungen aufrücken. Auch ein gebräuchliches Verfahren ist das kontinuierliche Auslesen, welches jedoch nur grobe Helligkeitsinformationen liefern kann oder das sogenannte *interlaced* bzw. *Zeilensprungverfahren*, bei dem abwechselnd gerade und ungerade Zeilen eingelesen werden, um Flimmern zu vermeiden.

Um während des Schiebens dieses Registers nicht zusätzliche Ladungen aufzunehmen und somit das Bild zu verfälschen, wurden verschiedene Verfahren entwickelt, welche schematisch in Abbildung 11 dargestellt wurden. Der naheliegende Weg ist, den Sensor während des Auslesens mechanisch zu verschließen (*Full–Frame–CCD*). Diese Methode ist allerdings aufwändig und störanfällig. Vom Gedanken ähnlich ist das Verfahren, die Ladungen in einen Zwischenspeicher, einem abgedunkelten Bereich im CCD–Chip, zu schieben (*Frame–Transfer–CCD*). Hierbei wird allerdings die doppelte Zahl an Elementen benötigt und die Belichtungszeit ist durch die Dauer des Zwischenspeicherns nach unten hin stark beschränkt. Diese Methode wurde deutlich verbessert, indem der Zwischenspeicher kein zusammenhängender, vom beleuchteten Teil abgetrennter Bereich mehr ist, sondern die abgedeckten Elemente nun direkt an den zugehörigen, bildgebenden Elementen angeschlossen sind (*Interline–Transfer–CCD*). Dies ergibt zwar eine höhere Geschwindigkeit, jedoch verweilen die Ladungen weiterhin zu lange in den Speicherzellen, um beispielsweise Lichtbeugungen, die die Abdeckung umgehen können, entgehen zu können. Da dies in einem abgedunkelten Bereich des Chips nicht geschehen würde, lädt man beim *Frame–Interline–Transfer–CCD* die Ladungen aus den 1. Zwischenspeicherzellen direkt in einen solchen Bereich, wo sie dann während der nächs-

ten Messung störungsfrei einzeln ausgelesen werden können. Die nun pro Pixel benötigten 3 Zellen machen solche Chips allerdings auch äußerst teuer. Anwendung finden sie beispielsweise in Hochgeschwindigkeitskameras[LMLM12].

Natürlich lassen sich Empfindlichkeit oder Geschwindigkeit etwa durch zusätzliche, parallele Auslesekanäle oder Sammellinsen über den einzelnen bildgebenden Zellen weiterhin verbessern.

Die Datenübertragung erfolgt dabei zunächst Analog jeweils als Kondensator–Entladung der in den einzelnen Elementen gespeicherten Ladung. Während die Belichtungszeit bei vielen CCD–Kameras extrem kurz gewählt werden kann, ist die Geschwindigkeit des Auslesens meist durch einen Analog–zu–digital–Wandler beschränkt. Da die Kabelverbindung hierzu meist sehr kurz gehalten ist und elektrische Felder in Kabeln mit bis zu $2{\cdot}10^8\ \frac{\mathrm{m}}{\mathrm{s}}$ übertragen werden, ist die Beschränkung des analog–führenden Kabels zu vernachlässigen. Der Digitalwandler üblicher digital–Kameras arbeitet mit dem derzeit gebräuchlichen USB2-Anschluss, welcher 480 $\frac{\mathrm{Mbit}}{\mathrm{s}}$ übertragen kann, wobei 30 fps (Bilder in der Sekunde) der Größe von ca. 640·576 px üblich sind.

Wichtig allgemein in Bezug auf Kameras ist auch immer die *Belichtungs–Empfindlichkeit*, sowie auch die Abhängigkeit vom *Farbspektrum*. Hierbei sind CCD–Sensoren auch im Vorteil gegenüber vielen anderen Sensoren, da Elektron–Loch–Bildung bereits bei niedrigen Energien (Austrittsarbeit im Trägermedium) stattfindet, oft aber auch Verstärker vorgeschaltet werden.

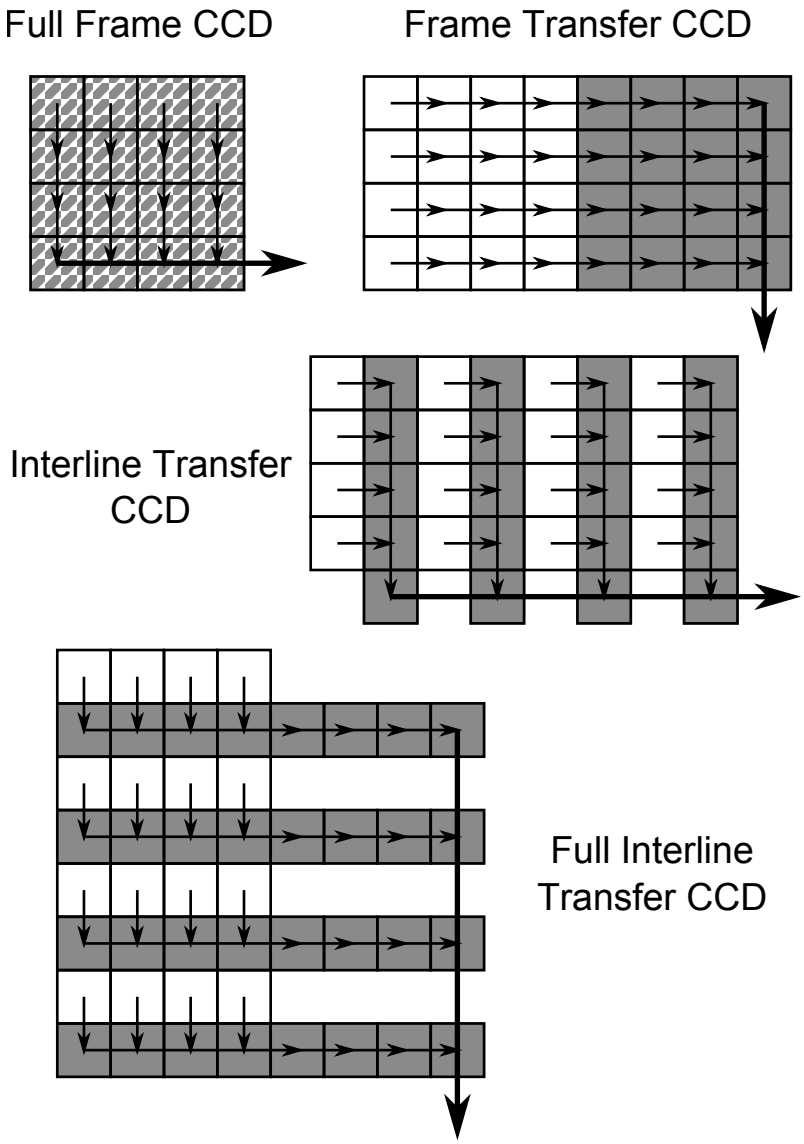


Abbildung 11
Verfahren zur Ladungsverschiebung innerhalb des CCD–Sensors; dunkle Flächen sind in blinden Bereichen der Kamera und nehmen so keine Helligkeitsinformationen auf.

3. Aufbau der Kamera und Softwareentwicklung

3.1. Strahlkamera

Das Strahlkamerasystem, wie es Abbildung 12 schematisch zeigt, soll besonders kompakt und alle detektierenden Komponenten innerhalb der Vakuum–Kammer einsetzbar sein, sodass es unabhängig von Sichtfenstern und in beliebigen Vakuumaufbauten benutzt werden kann. Die abgebildeten Kernelemente stellen dabei zwei in Chevron-Anordnung stehenden MCPs (vergleiche dazu Kapitel 2.4.2), einen Phosphorschirm und den verwendeten CCD–Sensor dar.

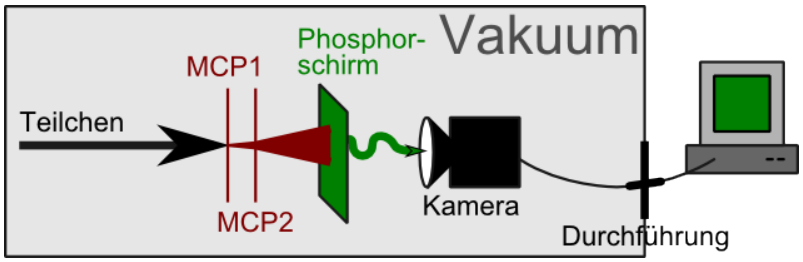


Abbildung 12
Schematischer Aufbau der Strahlkamera und des Computeranschlusses

Das Prinzip dieses Aufbaus wurde in einer vorangegangenen Bachelor–Arbeit[Ebe09] entwickelt und in einem Vertiefungsmodul[Rin12] verfeinert. Bei der dort verwendeten Strahlkamera konnten allerdings Einstellungen an dem CCD–Sensor nur außerhalb der Vakuumkammer vorgenommen werden, wobei es beim Eintritt in den Hochvakuum–Bereich häufig zu Einstellungsänderungen kam. Auch war es mit dem System nicht möglich, eine Echtzeit–Auswertung anzufertigen.

Im Unterschied zu diesem alten Aufbau verwenden wir nun einen deutlich besseren CCD–Sensor. Er lässt sich digital ansteuern, auch während des Vakuumbetriebs, besitzt eine höhere Empfindlichkeit (Faktor 2) und lässt überdies ein höheres Vakuum ($2{,}2{\cdot}10^{-8}$ mbar statt $3{\cdot}10^{-8}$ mbar) zu. Außerdem wurde nun ein Computer–Programm entwickelt, das in der Lage ist, die Sensor–Daten automatisch und während der Messung auszuwerten.

3.2. CCD–Kamera

Da wir in unserem Aufbau die CCD–Kamera in den Detektor integrieren wollen, müssen wir an diese einige zusätzliche Anforderungen stellen. Eine der wichtigsten davon ist die Vakuumtauglichkeit. Um die MCPs nutzen zu können, benötigt man min. einen Druck von $< 10^{-6}$ mbar, wohingegen im geplanten Einsatz $< 10^{-7}$ bis $< 10^{-8}$ mbar benutzt wird. Man könnte die Kamera auch auslagern und hinter ein Sichtfenster montieren. Allerdings würde dies den Einsatz auf Vakuum–Kammern mit Sichtfenstern limitieren und man müsste

die Kamera bei jeder Montage neu fokussieren. Außerdem würde man kompliziertere Linsensysteme benötigen.

Außerhalb des Laborbedarfes werden gebräuchliche CCD–Sensoren nicht auf Vakuumtauglichkeit getestet. In unserem Fall haben wir eine Kamera modifiziert, um sie vakuumtauglich zu machen (Öffnung geschlossener Kammern, Entfernung von Kunststoff–Schaum und Vergießen der Platinen). Außerdem wurde sie an eine Aluminium–Platte geschraubt, um die Wärme an ein speziell dafür entwickeltes Kühlsystem (siehe Kapitel 2.3) zu leiten.

Da der Phosphorschirm nur Signale von geringer Leuchtkraft aussendet, ist auch eine gute Empfindlichkeit notwendig. Im Vergleich zur alten Kamera (vgl. [Rin12]) wurde dies um Faktor 2 verbessert. Außerhalb des Laborbedarfes ist dies nur für starke Nachtsicht–Kameras üblich. Zusätzlich benötigt man in diesem Zusammenhang lange Shutter–Zeiten, in denen die Helligkeit integriert wird, um die Ereignisse sichtbar zu machen. Hierbei sind Shutter–Zeiten von min. $\frac{1}{100}$ s notwendig, um die Ereignisse gut zu sehen. Dabei ist natürlich auch die wellenlängenabhängige Empfindlichkeit wichtig, da der Phosphorschirm nur mit einer Wellenlänge von 450 nm abstrahlt (siehe Datenblatt in Abbildung 28). Unsere Kamera ist im sichtbaren Licht gut empfindlich, besonders jedoch bei grünem Licht bei 550 nm, hat aber auch gute Empfindlichkeit im Infrarot–Bereich (Abbildung 29).

Ein weiterer wichtiger Faktor ist die Temperaturentwicklung. Da im Vakuum Konvektion nicht vorhanden ist, ist es notwendig, die Wärme anderweitig abführen zu können (beispielsweise über Verschrauben an Metallplatten bzw. Vergießen der Platinen) oder die Stromversorgung zu drosseln. In unserem Fall benötigt die Kamera etwa 1,3 W und wird bei Normaldruck etwa 60° C bzw. nach den oben genannten Modifikationen (Platinen vergießen, an Aluminiumplatte schrauben) 50° C heiß.

Da es außerdem extrem nützlich ist, die Kameraeinstellungen ändern zu können, um beispielsweise die Shutterspeed oder den Gain an die Verhältnisse anzupassen ohne die Vakuum–Kammer öffnen zu müssen, sollte es sich um eine Digital–Kamera handeln. Auch ist dabei das digitale Übertragungssignal wesentlich Sötrungsunempfindlicher als analoge. Digitale Kameras bestehen aus einem analogen Chip und einem ADC, an welchem eine USB–Schnittstelle angebracht ist. Über die Schnittstelle lässt sich dabei Kamera– und ADC–Verhalten steuern und gleichzeitig eine 5V–Stromversorgung aufbauen.

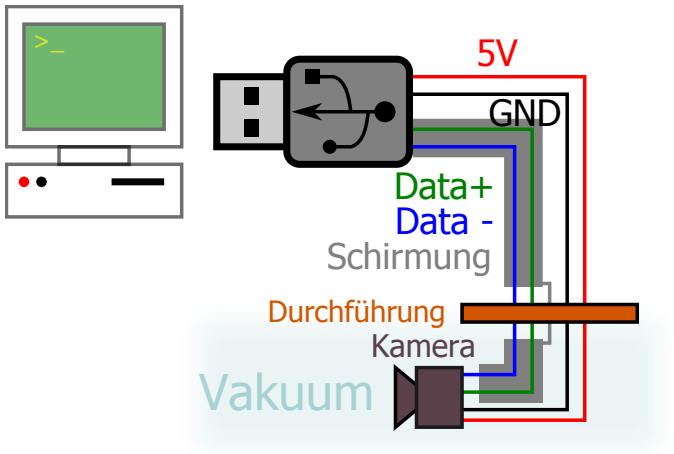


Abbildung 13
Durchführung und Schirmung eines USB2–Kabels in das Vakuum

Der verwendete Standard zur Gewährleistung einer Bildübertragungsrate von min. 30 fps (meist durch ADC beschränkt) ist dabei USB2. Da hierbei die Data+ und Data-–Leitungen extra geschirmt sind (siehe hierzu auch Abbildung 13), müssen diese Leitungen auch im Vakuum extra geschirmt und die Schirmung mit zum Anschluss durchgeführt werden.

Hinzu kommt noch, dass die Kamera ausgesprochen kompakt ist und somit ideal für den geplanten Einsatz.

Zuletzt spielt natürlich auch der Preis eine Rolle, da er sich im Gebiet der CCD–Kameras um Größenordnungen unterscheiden kann. Während man gute Kameras im Laborbedarf mit z.T. weitaus besseren Werten oft im 5–stelligen Euro–Bereich erwerben kann, konnte die eigentlich als Überwachungskamera beworbene CCD–Kamera in unserem Fall bereits im unteren 3–stelligen Bereich gekauft werden. Dabei ist natürlich weiterhin zu beachten, dass der Hersteller weniger ausführliche Tests (Vakuum, Hitzeverträglichkeit etc.) durchgeführt hat und dementsprechend selbst getestet bzw. angepasst werden muss.

3.3. Versuchsaufbau

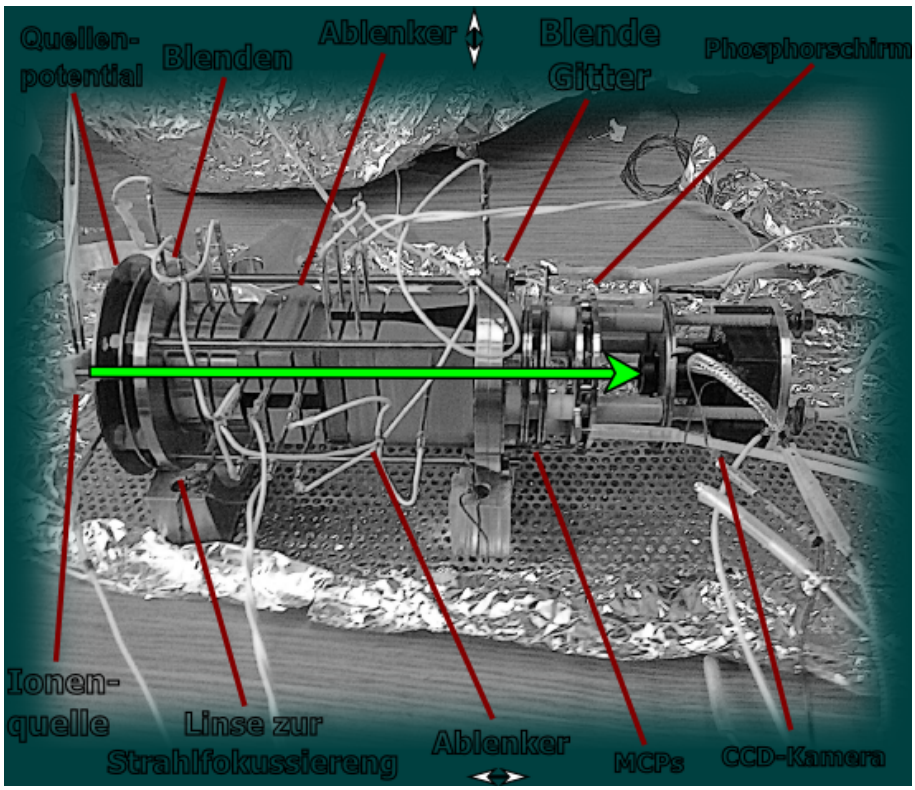


Abbildung 14
Photo des Aufbaus zum Testen von Kamera und Programm

Um Kamera und Programm testen zu können, wurde ein Probeaufbau (Abbildung 14) verwendet, in dem ein Ionenstrahl auf MCP, Phosphorschirm und Kamera gelenkt wurde[Ebe09]. Der Probeaufbau enthält dabei eine Cs–Ionenquelle, eine elektrischen Linse zur Strahlfokussierung, sowie Ablenk–Elemente in horizontaler und vertikaler Richtung. Hinter der Strahlsteuerung (in Strahlrichtung gesehen) befindet sich ein Gitter und 2 MCPs zur Signalverstärkung. Dahinter befindet sich der Phosphorschirm und die über USB2 angeschlossene CCD–Kamera.

Als Vakuum–Kammer für den Aufbau wurde ein vierer CF160 Kreuz benutzt. Als CCD-Kamera wurde das Model 21K35USB–C der Firmal Videology (Siehe Datenblatt in Abbildung 27) benutzt, wobei es sich um eine Interline–Transfer–CCD–Kamera handelt (Vergleiche dazu Erklärung in Kapitel 2.4.4).

An der Ionenquelle liegt eine Beschleunigungspotential von 300 V an, die Linse wird mit ca. +300 V betrieben, die Ablenker jeweils mit Werten zwischen -10 V und +10 V. Das erste MCP wurde mit 3,5 kV vorne und 3 kV hinten, das zweite mit 2,5 kV und 2 kV betrieben. Der Phosphorschirm benötigt etwa 5 kV. Die Kamera bezieht +5V über den USB2–Anschluss aus dem Computer.

3.4. Datenaufnahme–Software

Da die Entwicklung des Programms wesentlicher Bestandteil dieser Arbeit war und viele Algorithmen direkt für das Programm entwickelt wurden, soll hier nun näher auf die Algorithmen eingegangen werden. Die Algorithmen werden hierbei beschrieben und über ein Ablaufdiagramm veranschaulicht. Im Anhang befindet sich jeweils der zugehörige C++-Quellcode.

Dabei ist zu beachten, dass die Erklärung des Algorithmus' keine Überprüfungen, ob verwendete Variablen leer oder Pointer zulässig sind, und keine genauere Ansteuerung der Anzeigeelemente enthalten.

3.4.1. Auswahl der Programmiersprache

Zur Entwicklung des Programms, das Hauptbestandteil dieser Arbeit ist, wurde *Embarcaderos*®s *C++-Builder*® aus dem Paket XE2 (Version 16) benutzt.

Embarcaderos (ehemals *Borland*®) C++-Builder ist ein sogenanntes *RAD* (*Rapid Application Development*). Dieses bietet verschiedene Vorteile, beispielsweise die einfach zu bedienende grafische Oberfläche oder die automatische Integration der *OWL*– (*Object Window Libary*) und *VCL*–Bibliotheken (*Visual Component Libary*). Hiermit ist es schnell und komfortabel möglich, grafische Oberflächen für die eigenen Programme zu erstellen. Auch handelt es sich hierbei um einen *ANSI/ISO–Compiler*, er kann also in der Geschwindigkeit mit klassischen ANSI–C++-Programmen konkurrieren. Außerdem sind bereits viele Erweiterungen und eigene Bibliotheken enthalten, die das Suchen und Einbinden von Bibliotheken fremder Quellen überflüssig machen.

Vergleicht man den C++-Builder direkt mit anderen Entwicklungsumgebungen wie etwa das populäre *Microsoft Visual Studio C++*, so lassen sich direkt einige Vor- aber auch Nachteile erkennen: Da Microsoft Visual Studio C++ sehr bekannt ist, haben sich viele Drittanbieter entschlossen, hauptsächlich dafür zu programmieren. Dies muss aber nicht zwangsläufig den C++-Builder außen vor lassen, da dieser mit vielen Kompatibilitäts–Einstellungen, beispielsweise die automatische Uminterpretation von Microsoft–typischen Befehlen oder die standardmäßige Implementierung der *Microsoft Foundation Class* (*MFC*), aufwarten kann.

Außerdem hat Embarcadero während der Zeit, die die verschiedenen Versionen des C++-Builders auf dem Markt sind, gezeigt, dass es stabil bleibt und stetig weiterentwickelt wird, wobei Projekte veralteter Versionen auch immer in neueren compiliert werden konnten.

Natürlich gibt es noch weitere Vor- und Nachteile, beispielsweise Datenbankprogrammierung oder Portierbarkeit auf Linux und andere Entwicklungsumgebungen; in unserem Fall sind jedoch die genannten die ausschlaggebenden

Argumente zur Verwendung gewesen[HS00].

3.4.2. Grundlagen zur Programmoptimierung

Multithreading: Im Programm wird zur Erfassung der Kamerabilder die sogenannte DirectShow-API (Application Development Interface) von Microsoft verwendet. Da das Programm beim Warten auf eine Bildübertragungen dabei anhält, wird es für weitere Benutzereingaben währenddessen gesperrt. Auch wenn dabei das Programm nur im Bereich von 0,1 s gesperrt ist, ist der Effekt beim Klicken auf Schaltflächen oder sonstigem Bedienen offensichtlich.

Um dies zu verhindern, lagert man die blockierenden Teile des Codes im Multithreading in sogenannte getrennte Threads aus. Dabei handelt es sich um Programm–Teile, die noch im selben Prozess und Prozessorkern ablaufen, die allerdings durch geschickte Implementierung semi–gleich ablaufen können. Der Hintergrund–Thread kann somit zeitweise anhalten ohne den Haupt–Thread zu beeinflussen. Hohe Prozessorauslastung lässt jedoch weiterhin beide Threads temporär anhalten[Mey06].

Externe Variablen: Will man von mehreren Programmcode–Teilen, beispielsweise verschiedenen Fenstern, auf Variablen zugreifen, die in anderen Teilen ebenfalls benutzt werden, stößt man schnell auf Probleme. Eine in mehreren Dialogen definierte Variable ist in jedem Dialog neu und ohne Inhalt. Durch die Definition als externe Variable wird jedoch erklärt, dass das Programm auch fremde Dialoge und Fenster nach neuen Inhalten bzw. alten Definitionen durchsuchen soll. Ein Beispiel dafür wäre der klassische Einstellungs–Dialog, der intern in eine Einstellungsvariable speichert, die auch vom Hauptprogramm oder von anderen Unterfenstern ausgelesen werden soll. Will man dabei keinen Umweg über das Speichern in Dateien machen, benötigt man hier externe Variabeln[HS00].

Speicherformate: Da in diesem Programm Daten verschiedener Art gespeichert werden müssen, ist das richtige Dateiformat hier wichtig.

Als Bildformat ist für Kamerabilder das PNG–Format von nutzen, da gut durch andere Programme darstellbar und verlustlos komprimiert. JPG reduziert bei detaillierten Bildern zwar Dateigröße, verliert aber durch verlustreiche Komprimierung Details[Roe99].

Für das Exportieren von der Wertetabelle wurde sich in diesem Programm für eine reine Text-Datei mit Trennzeichen zwischen Spalten und Zeilen entschieden. Diese ist einfach im Texteditor darstellbar und kann ebenso einfach von anderen Programmen eingelesen werden.

Die Programm–Einstellungen hingegen müssen lediglich durch das glei-

che Programm wieder eingelesen werden. Um dabei mehr Struktur in die gespeicherten Daten zu bringen und unterschiedlichen Darstellungen von Trennzeichen auf verschiedenen Systemen aus dem Weg zu gehen, benutzt man das XML-Format. Dabei wird das Dokument in Knoten und Unterknoten mit Namen und Inhalten unterteilt, die im Programm einzeln ansteuerbar sind, statt die Datei zeilenweise auslesen zu müssen[BPSM+97].

3.4.3. Erkennen eines Signals

Bei der Erkennnung eines Signals innerhalb eines Bildes kann man in 2 Arten vorgehen: Entweder man such nach Helligkeitsspitzen über einem gewissen Schwellenwert oder man sucht nach starken Anstiegen/Abfällen (hohe Gradienten). Vorteil der ersten Methode ist das einfache Einstellen eines passenden Wertes über ein Histogramm, während man im 2. Fall Gradienten abschätzen muss.

Der größter Nachteil hingegen zeigt sich bei Signalen unterschiedlicher Helligkeiten im selben Bild (beispielsweise bei Beugung hinter Gittern). Die Gradientenmethode würde ein solches Verhalten nicht stören, allerdings reagiert diese sehr empfindlich auf Rauchen und erkennt verschmierte Signale ggf. nicht mehr. In unserem Fall haben wir uns für die Schwellenwert–Methode (Abbildung 15) entschieden. Bei den im Folgenden angegebenen Bildern wurde ein Lochgitter (1 mm) verwendet, durch das ein Ionenstrahl betrachtet wurde.

Bei der Analyse gehen wir im Bild zeilenweise von links nach rechts und oben nach unten vor. Aus dem übertragenen Bild wird der Grauwert des Pixels ermittelt und mit dem Schwellenwert verglichen. Ist der Schwellenwert niedrieger, wird eine Signal–Variable mit Koordinaten des Pixels, Helligkeit (Grauwert) und der Breite 1 gesetzt. Für jedes weitere Pixel wird x–Position und Breite um 1 erhöht und die Helligkeit aufsummiert. Unterschreitet der Grauwert den Schwellenwert wieder oder gelangt man ans Ende der Zeile, wird die x–Position um die halbe Breite verringert (Mitte des Signals). In einer Liste bisher erkannter Signale wird danach nach Signalen gesucht, in deren Breiten–Umgebung (x,y $\pm$ Breite/2) das neue Signal liegt oder umgekehrt (siehe dazu auch Abbildung 16).

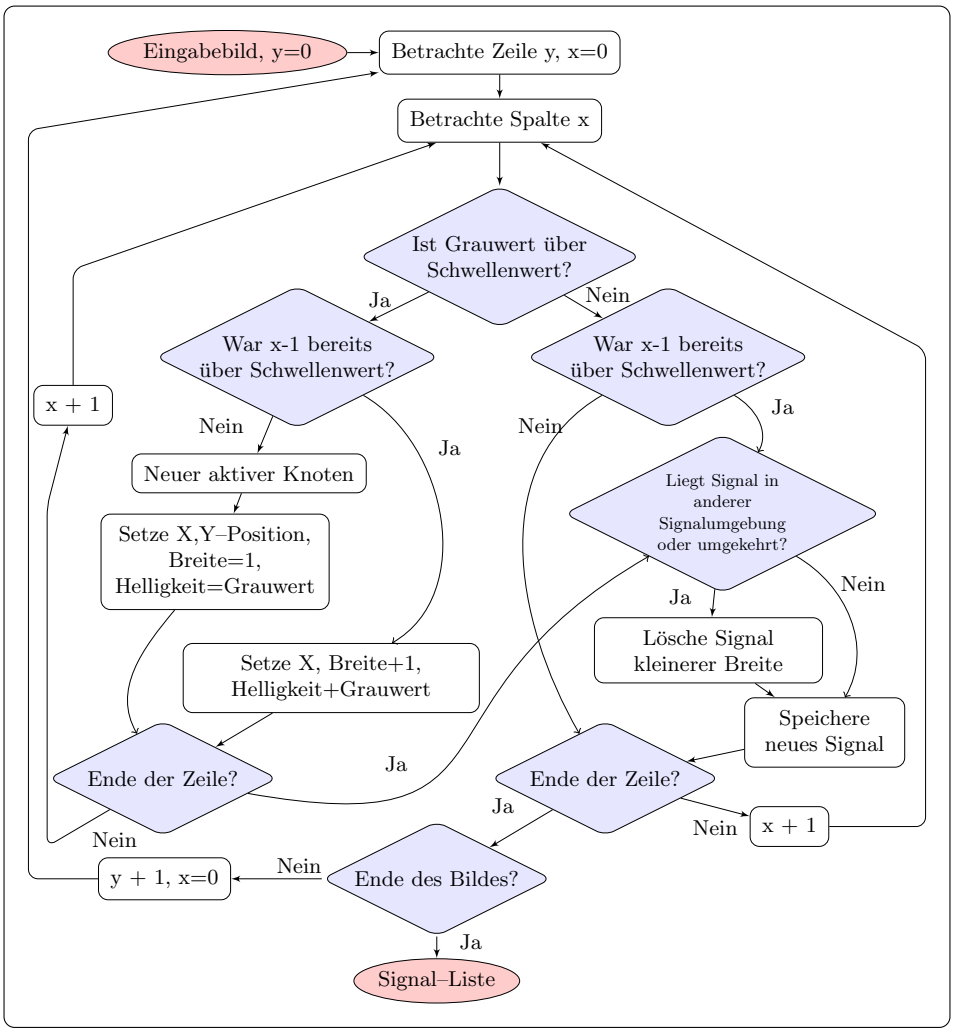


Abbildung 15
Ablaufdiagramm der Signalerkennung in einem Bild

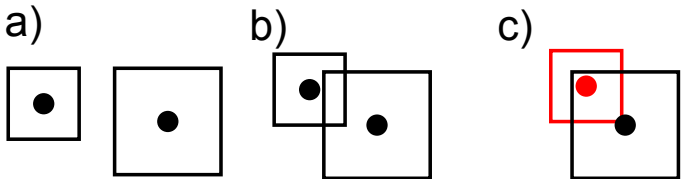


Abbildung 16
Signale und Signalumgebungen.
a) Signalumgebungen überschneiden sich nicht.
b) Signalumgebungen überschneiden sich, aber kein Signal liegt in der Umgebung des anderen.
c) Das rote Signal liegt in der Umgebung des anderen; Im Algorithmus wird daher nur das schwarze, größere als getrenntes Signal gespeichert.

Das Signal mit der kleineren Breite wird dabei gelöscht, um den Mittelpunkt des Signals zu bestimmen. Ein rundes Signal wird dabei vorausgesetzt. Anschließend wird das Signal in der Liste gespeichert und der Scan bis zum Ende des Bildes fortgesetzt.

Das hier verwendete Prinzip des Schwellenwertes wurde noch um die sogenannte *Fuzzy–Logik* erweitert: Die Helligkeit des Signals wird erst nach Überschreiten des 2. Schwellenwertes (der größer gleich dem 1. Schwellenwert zur Signalerkennung sein muss) voll gewichtet.

Liegt die die Helligkeit zwischen den beiden Schwellenwerten, wird die Gesamthelligkeit des Signals nur gewichtet erhöht. Liegt also die Helligkeit genau in der Mitte, erhält sie einen Faktor $\frac{1}{2}$. Die Fuzzy–Logik erweitert dabei die boolische Logik [Signal, kein Signal], um Unschärfen darzustellen (beispielsweise hellere Bereiche um das Signal oder durch Rauschen verursachte Helligkeitseinbrüche).

3.4.4. Bildverbesserung

Um die Signalerkennung zu verbessern, bedienen wir uns hier verschiedener Filter, die das Bild pixelweise betrachten. Diese werden benutzt, um zunächst ein Leerbild (Bild ohne Signal) abzuziehen, Pixel zusammenzufassen und das Bild weichzuzeichnen (um Rauschen zu entfernen) und zuletzt Kontrast und Helligkeit zu verändern (um Signale deutlicher zu erkennen).

Kontrast, Helligkeit und Leerbildabzug
Um den Kontrast zu erhöhen, wird der absolute Pixelwert im Bild mit einem Faktor multipliziert. Dies erhöht den Helligkeits-Abstand zwischen unterschiedlich hellen Pixeln, führt allerdings auch dazu, das Bildteile komplett weiß werden und somit Details verlieren. Um dies zu beheben wird der Pixelwert um einen einstellbaren Wert verringert, sodass das Bild insgesamt dunkler wird. Zusätzlich wird der Pixelwert aus dem Leerbild ausgelesen und hier abgezogen, sodass ein Differenzbild entsteht. Damit lassen sich auch tote Pixel und

konstante Signale entfernen. Das Ablaufdiagramm dazu ist in Abbildung 25 dargestellt.

In Abbildung 25 sind nun im Vergleich zum Ausgangsbild aus Abbildung 24 die Signale im Bild deutlich besser und das Histogramm deutlich vertikal gespreitzter zu sehen.

Weichzeichner und Pixelzusammenfassung
In einem einfachen Plus–Median–Weichzeichner (wie in Abbildung 18), gehen die Pixel an den Positionen $(x \pm 1 | y \pm 1)$ mit einfachem Gewicht und $(x|y)$ mit doppeltem ein. Die gewichtete Summe wird normiert (also bei Gewichten der Werte 1 und 2 durch $4 \cdot 1 + 2 = 6$ geteilt) und als neuer Wert gesetzt. Je stärker dabei das Gewicht des aktuellen Pixels, desto weniger wird weichgezeichnet.

Das Zuammenführen der einzelner Pixel unterdrückt einerseits Rauschen, erhöht aber auch die Geschwindigkeit des Weichzeichners, da nun weniger Pixel überprüft werden müssen (der oben genannte Weichzeichner benötigt 5 lesenzugriffe pro Pixel). Dabei wird auf die C++-Methode `Resize()` zurückgegriffen, die zunächst beim Verkleinern mehrere Pixel zusammenfasst und nach dem Weichzeichnen den Wert der zusätzlich einzufügenden Pixel durch die umgebenden mittelt.

Vergleicht man nun Abbildung 17 mit Abbildung 26 scheint die Signaltextur nun viel homogener, was sich auch im glatteren Histogramm zeigt. Außerdem sind hier auch kaum noch Störungen außerhalb der 5 hellen und 2 dunklen Ereignisse zu erkennen, der restliche Bildbereich ist rein schwarz.

3.4.5. Ziehen einer Linie nach Bresenheim

Da im Programm ein Histogramm entlang einer einstellbaren Linie dargestellt werden soll, ist ein Algorithmus notwendig, der effizient alle Pixel einer geraden und lückenlosen Linie anhand von Start– und End–Punkt ansteuern kann. Dies mag zunächst nicht schwierig klingen, lückenlose Linien und das Ansteuern von ausschließlich jenen Pixeln, die auch benötigt werden, ist bei Rastergrafiken allerdings problematisch. Auch zeigt sich die Effizienz des Algorithmus schnell bei Linien, die quer über Flächen von 100.000 px gezogen werden.

Der sogenannte Bresenheim–Algorithmus ist dabei besonders effizient und einfach zu realisieren. Man bedient sich dabei einer Zwischenvariable, in Abbildung 19 *Fehler* genannt. Man speichert zunächst die Abstände der x– (Δx) und y–Komponenten (Δy) des Anfangs– und Endpunktes der zu zeichnenden Linie. Wird beim Setzen des nächsten Pixels der Linie die x–Komponente der Position verändert, addiert man Δx zu $Fehler$. Verändert man die y–Komponente der Position, wird Δy vom $Fehler$ abgezogen. Ist dabei $-\frac{\Delta y}{2} < Fehler < \frac{\Delta x}{2}$, werden x– und y–Komponente für das Setzen des nächsten Pixels um 1 in

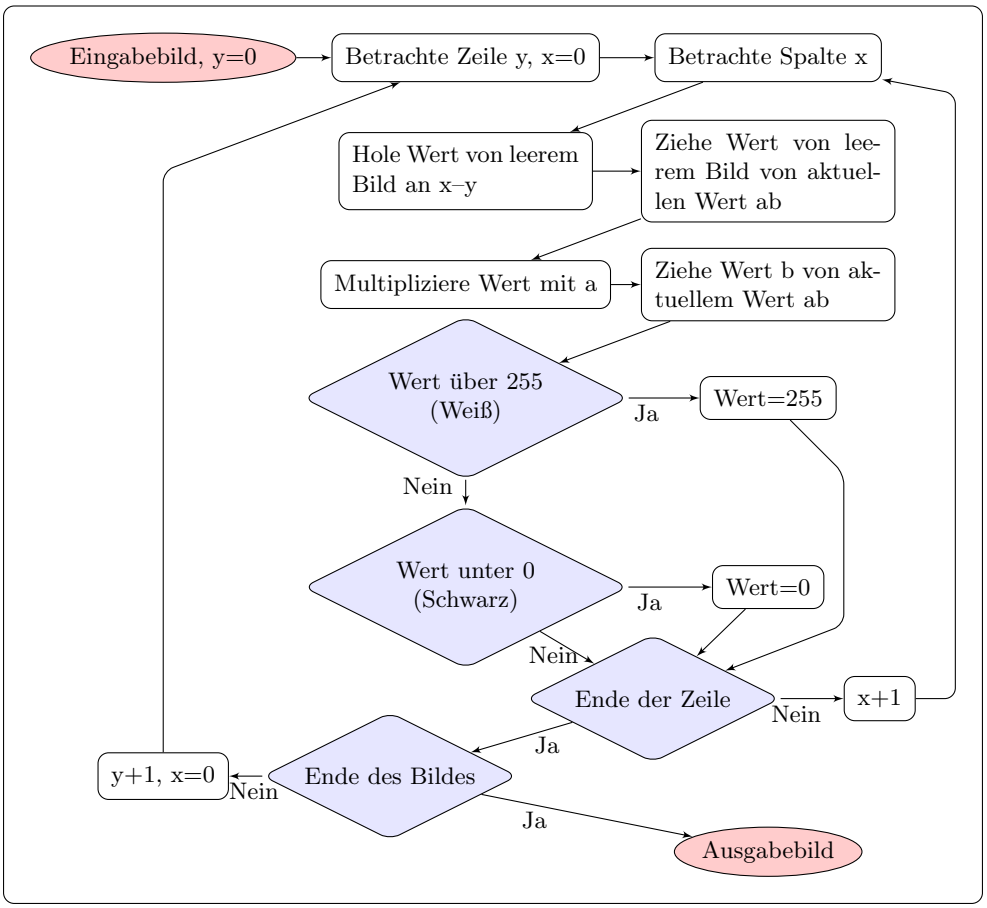


Abbildung 17
Ablaufdiagramm der Leerbildabzuges und Kontrasterhöhung in einem Bild

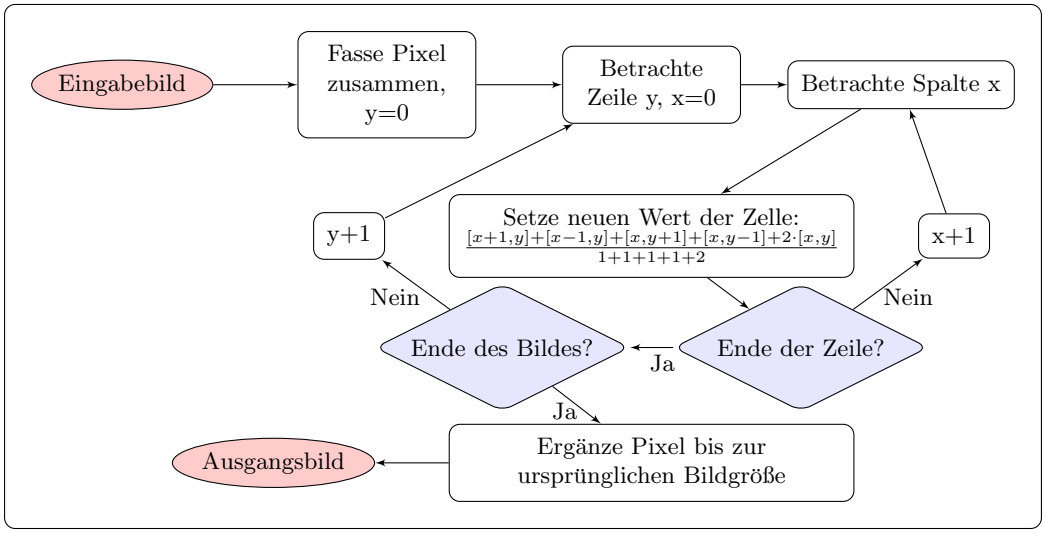


Abbildung 18
Ablaufdiagramm: Pixelzusammenfassung und Weichzeichner

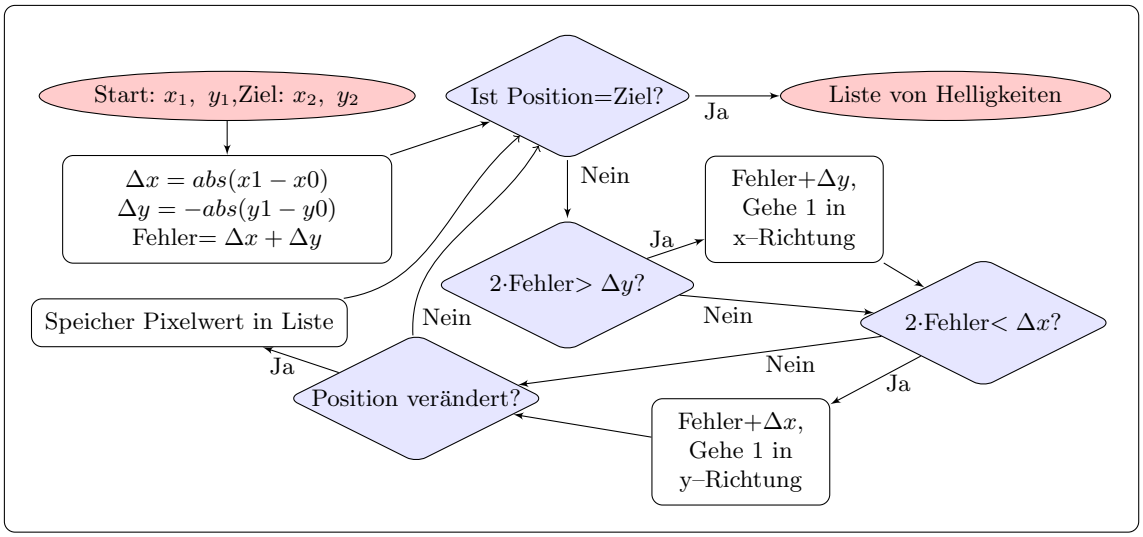


Abbildung 19
Ablaufdiagramm des Bresenheim–Algorithmus' zum Rastern einer Linie.

Richtung des Zielpunktes verändert. Dies entspricht ähnlichen Seitenlängen des die Linie umschließenden Rechtecks, die Linie sollte also möglichst diagonal sein. Wird $Fehler < -\frac{\Delta y}{2}$, muss die Linie mehr in x–Richtung fortgesetzt werden. Hier wird lediglich die x–Komponente verändert und Δx addiert. Ist $Fehler > \frac{\Delta x}{2}$, muss weiter in y–Richtung vorgegangen werden und Δy abgezogen werden. Diese Fälle treten periodisch auf, sodass nach Erreichen des Zielpunktes eine gleichmäßige Linie zustande kommt[GS06].

4. Messung und Ergebnisse

4.1. Die Kamera im Vakuum

Im ersten Teil unseres Versuches wurden Aufbau und Kamera auf ihre Vakuumeigenschaften überprüft. Die Kamera wurde dabei durch das Vakuum in ihrer Funktionsweise nicht beeinflusst und man konnte ein Vakuum von $2,2 \cdot 10^{-8}$ mbar erzeugen (zum Vergleich $3 \cdot 10^{-8}$ mbar bei der alten Kamera[Rin12]).

Bei der Bildübertragung aus der Vakuum–Kammer konnten zwar vereinzelte, besonders helle („tote“) Pixel beobachtet werden, ihre Position verändert sich aber von Zeit zu Zeit. Da wir nun auch die Kamera–Einstellungen direkt ändern konnten, stellten wir fest, dass die Licht–Intensität der toten Pixel proportional zu der Belichtungszeit der Kamera ist. Mit maximaler Belichtungszeit von $\frac{4}{25}$ s erscheinen die Pixel weiß, bei $\frac{1}{100}$ s sind sie kaum erkennbar, wie in Abbildung 20 dargestellt. Da auch Spiegeln des Bildes (direkt am ADC, also Umkehrung der Sensor–Ausleserichtung) die toten Pixel nicht wie erwartet spiegelte, handelt es sich dabei vermutlich um Übertragungsfehler zwischen den CCD-Sensoren.

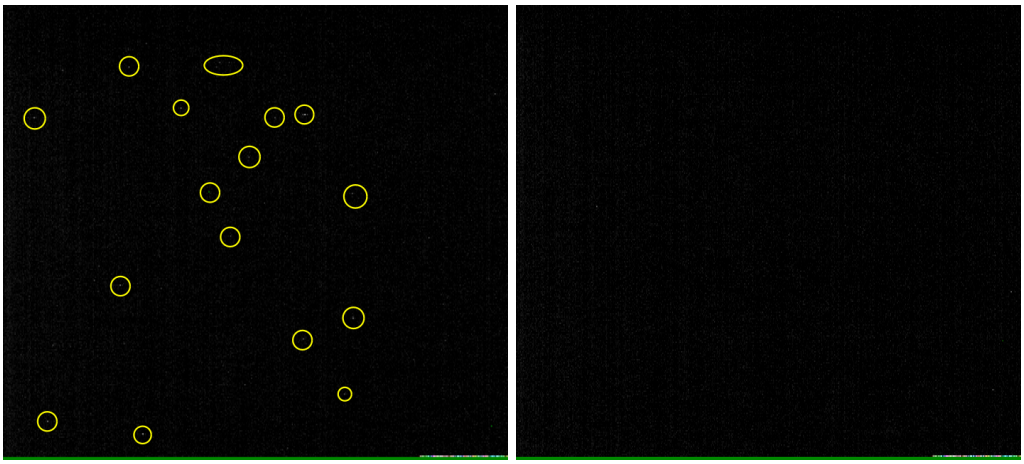


Abbildung 20
Bild der toten Pixel; links Belichtungszeit $\frac{4}{25}$ s, rechts $\frac{1}{100}$ s

Auch fiel auf, dass sich bei längerer Betriebsszeit (mehrere Stunden) der Kamera, helle, vertikale Streifen bilden (Abbildung 21). Hier beeinflussen bei zunehmender Wärme der Kamera einige Zellen ihre beiden horizontalen Nachbarzellen. Mithilfe des im späteren Einsatz eingebauten Kühlsystem wird dieser Effekt nicht mehr auftreten. Auch bereits jetzt lässt sich diesem Effekt mit der Funktion des Pixel–Zusammenfassens aus dem Programm entgegenwirken.

Im direkten Vergleich der Einzelionen–Aufnahmen der früher verwendeten Strahlkamera[Rin12] (Abbildung 23), lässt sich das Einzelionen–Signal nun deutlich besser erkennen und dessen Ausmaße bestimmen (Abbildung 22).

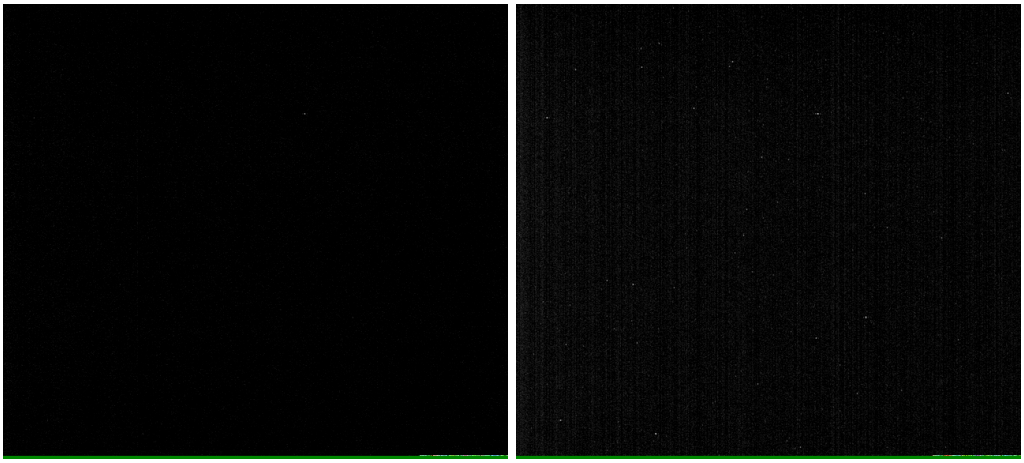


Abbildung 21
Ausbildung von Streifen nach längerer Kamerabenutzung. Links: Start der Messung; Rechts: 1,5 h nach Start

4.2. Ionenstrahl–Analyse

Nachdem der geeignete Druck von unter 10^{-6} mbar erreicht wurde, konnten die MCPs, der Phosphorschirm und die Ionenquelle in Betrieb genommen werden. Das Programm speicherte dabei automatisch Bilder des Eingangs, nach Bearbeitung und Signalerkennung, sowie ein Histogramm zu einer eingezeichneten Linie und wahlweise die Programmoberfläche. Außerdem wurde eine Zeitmessung gestartet, bei der die Signale mit Helligkeit, Position, Breite und Zeitpunkt (angegeben in übertragenen Frames) in einer Textdatei gespeichert wurden.

Das Programm ist in der Lage, mit eingeschalteten Analyse–Algorithmus und automatischer Bildspeicherung etwa mit 6 fps auszulesen, was gut mit dem Belichtungszeit–Modus $\frac{4}{25}$ zusammen arbeitet. Der Analyse–Algorithmus kann wahlweise auch ausgeschaltet werden, um Wiedergabe in Echtzeit zu ermöglichen.

Die nachfolgend bezeichneten Abbildungen sind jeweils im Bilder–Anhang zu finden. Zu den nachfolgenden Abbildungen existieren Bildschirmabbilder im Anhang, in denen Signalerkennung, Histogrammpfad und Kameraeinstellungen sichtbar sind. Die in diesem Kapitel zur Veranschaulichung eingebundenen Bilder entsprechen dem im jeweiligen Schritt korrigierten Bild und einer Histogrammaufnahme entlang der oberen Signalreihe.

Um die Einstellungen, wie unten gezeigt, korrekt vornehmen zu können, kann man die Bildaufnahme im Programm zeitweise anhalten und die Algorithmen zunächst nur auf das zuletzt empfangene Bild auszuführen.

Bei einer ersten Messung (Abbildung 24, Bildschirmabbild im Anhang Abbildung 30) wurden zunächst Kamera– und Programmeinstellungen getestet

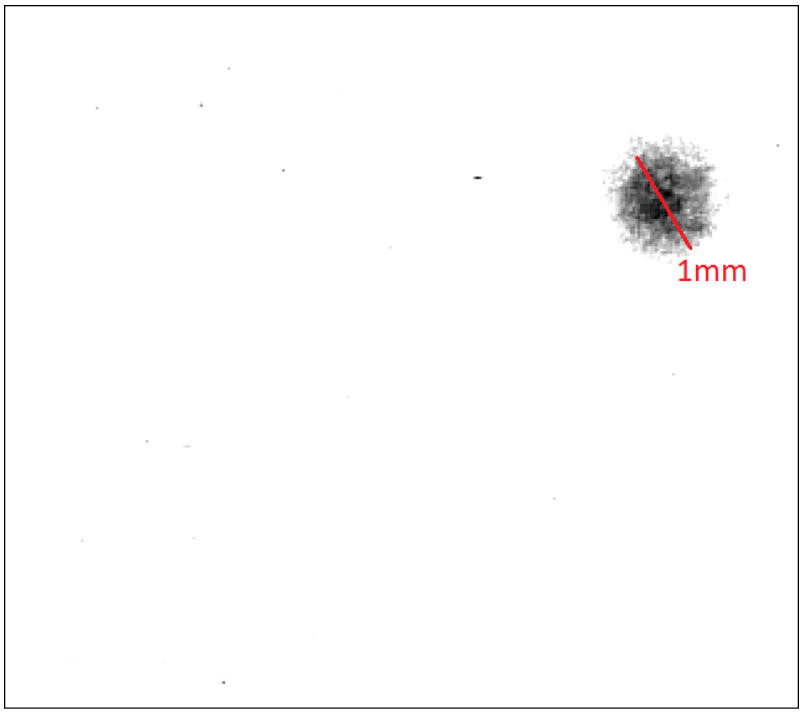


Abbildung 22
Invertierte Aufnahme der neuen Strahlkamera. Man erkennt nun klar Position und Ausmaße des einzelnen, hier gezeigten Einzelionen–Signals.

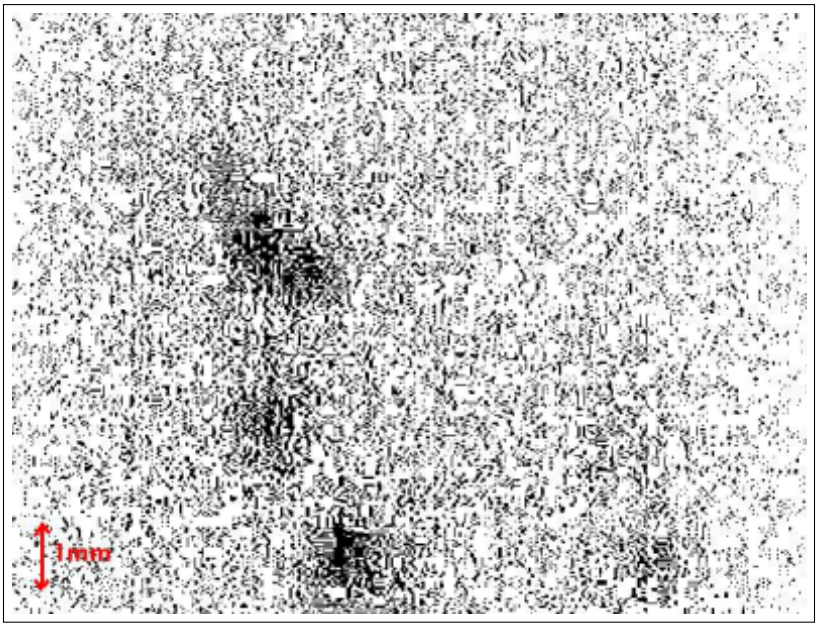


Abbildung 23
Invertierte Aufnahme mit der alten Strahlkamera im gleichen Aufbau zum Vergleich. Man kann 4 Einzelionen–Sinale erkennen.

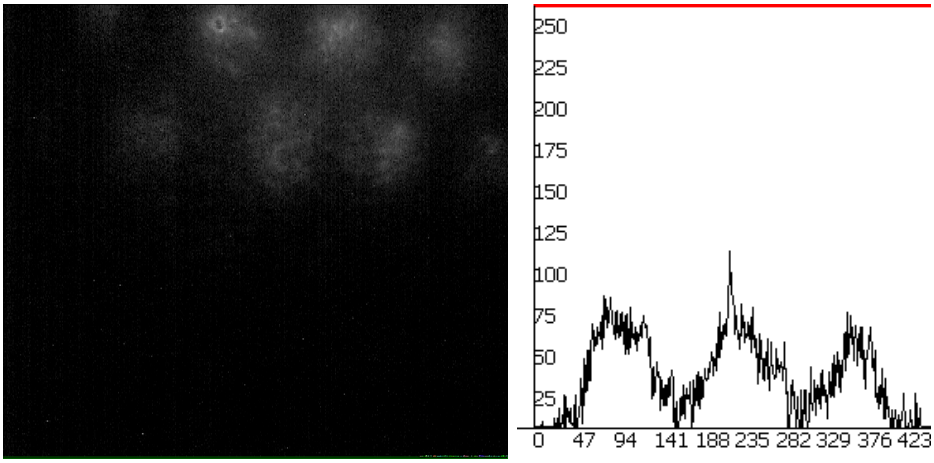


Abbildung 24
Von Kamera direkt aufgenommenes Signal (bei einer Shutterspeed von 4/25. Das Histogramm stammt von den oberen drei hellen Signalen)

und die Fähigkeit, Signale zu erkennen, erprobt. Dabei wurde ein Rauschbild (Leerbild ohne Signal) abgezogen, um statisches Rauschen zu entfernen und eine Analyse durchgeführt. Der Schwellenwert, ab der Signale erkannt werden, wurde hier nur grob eingestellt, trotzdem kann man hier gut erkennen, dass die Signale nur schwer gesehen werden können.

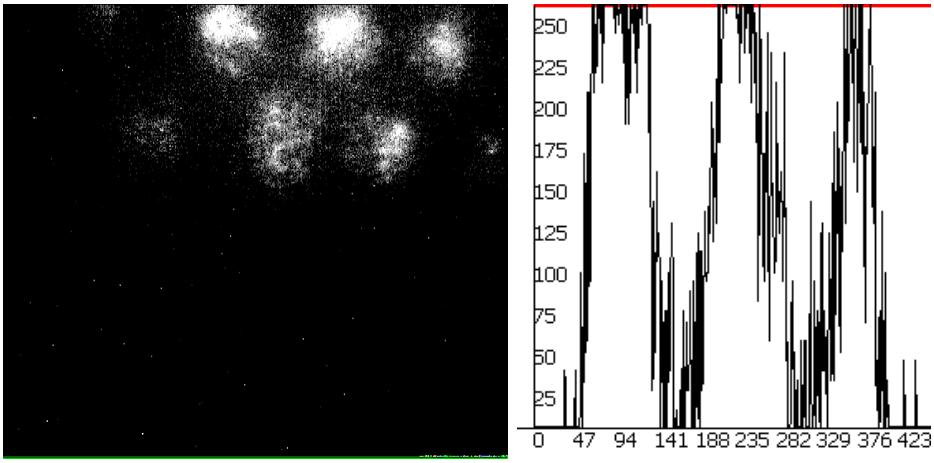


Abbildung 25
Kontrast und Helligkeitsanpassung angewandt auf Abbildung 24

Abbildung 25 (Bildschirmabbild im Anhang Abbildung 31) liefert nach der Kontrast– und Helligkeitsanpassung schon ein deutlich schöneres Bild. Das Signal ist allerdings weiterhin zerfurcht, was Probleme für die Signalerkennung bedeuten kann.

Indem nun der Weichzeichner eingesetzt und die übrigen Einstellungen entsprechend angepasst wurden, konnte eine Situation wie in Abbildung 26 (Bildschirmabbild im Anhang Abbildung 32) herbeigeführt werden, bei die Ereignisse nun gut erkannt werden können.

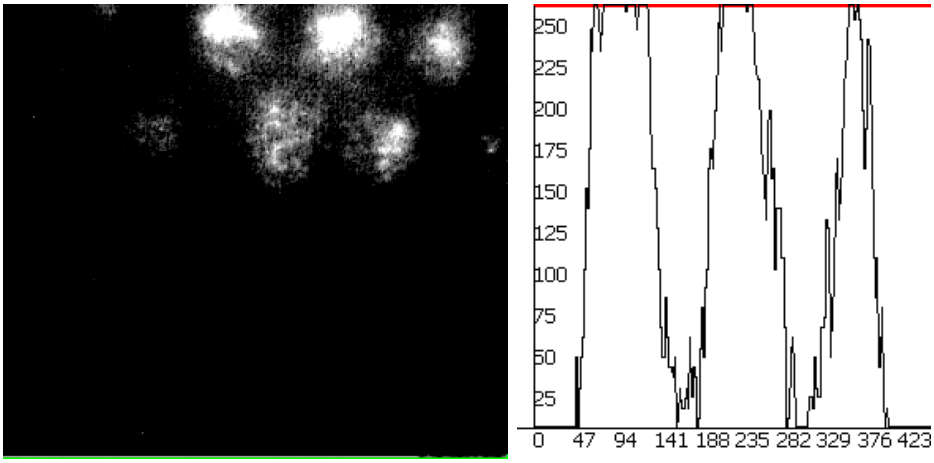


Abbildung 26
Pixelzusammenfassung und Weichzeichner angewandt auf Abbildung 25

Bildschirmabbild 33 zeigt dabei noch ein Histogramm über ein einzelnes Signal zur Darstellung dessen Topografie.

Um auch den zeitlichen Verlauf der Messung untersuchen zu können, kann eine Messungs–Aufzeichnung gestartet werden. Dabei werden Signalorte, Helligkeiten und Breiten, sowie der Frame, gezählt ab dem Start der Messung, in eine Liste geschrieben. Diese kann zusammen mit den Einstellungen in einer Text–Datei exportiert werden, wie in Abbildung 34 (Anhang) dargestellt.

Zuletzt wurden noch Einzelionen untersucht, indem die eigentliche Ionenquelle ausgeschaltet und auf zufällige Ionisierungsprozesse gewartet wurde. Diese Signale dauern dabei nur einen Frame ($< \frac{4}{25}$ s) an, konnten jedoch von der Kamera sehr gut aufgelöst und vom Programm ausgewertet werden.

5. Zusammenfassung und Ausblick

Ergebnis

Im Rahmen dieser Arbeit wurde eine Strahlkamerasystems für die Analyse von Ionenstrahlen im Vakuum entwickelt und in Betrieb genommen. Der Aufbau basiert dabei auf einer früheren Bachelor–Thesis[Ebe09, Rin12], wobei nun ein neuer, besserer CCD–Sensor und ein Analyse–Computer–Programm eingesetzt wurden.

Die im Experiment eingesetzte Kamera ließ im Vergleich zur vorherigen[Rin12] ein höheres Vakuum ($2{,}2 \cdot 10^{-8}$ mbar statt $3 \cdot 10^{-8}$ mbar) zu, ist um Faktor 2 empfindlicher (0,005 lux) und hat als Digitalkamera eine wesentlich weniger störanfällige Signalübertragung. Außerdem kann sie nun flexibel konfiguriert werden, sodass die Vakuum–Kammer bei Änderungen der Einstellungen nicht geöffnet werden muss.

Dem Analyse–Programm ist es dabei möglich, die von dem CCD–Sensor gelieferten Daten direkt während der Messung zu analysieren und darzustellen. Es beherrscht dabei Grundlegende Bildverbesserungs–Möglichkeiten (Einstellungen von Kontrast und Helligkeit) und kann Signalbreiten, –helligkeiten und –positionen zuverlässig und festhalten. Angezeigt werden können unter anderem die Eingabedaten, die Daten nach Durchlaufen der Bildverbesserungs– und Analysealgorithmen, sowie zugehörige Histogramme. Bei letzteren lässt sich auch ein Pfad über das Bild legen, dessen Verlaufshistogramm dann angezeigt wird (etwa um einzelne Signalprofile besser betrachten zu können). Zusätzlich lassen sich diese Bilder automatisch zwecks Dokumentation abspeichern. Das Programm selbst benötigt dabei nur sehr geringe Systemressourcen.

Im Experiment zeigte sich, dass das Programm zuverlässig arbeitet und die ankommenden Signale gut erkannt und vermessen hat. Damit ist der Aufbau nun bereit für den Einsatz am Massenspektrometer.

Verbesserung des Aufbaus

Auch innerhalb dieses Strahlkamerasystems sind noch Verbesserungen denkbar.

Um die Bildaufnahme weiter zu optimieren, könnte man etwa einen Phosphorschirm verwenden, der nicht mit 450 nm (Datenblatt in Abbildung 28), sondern mit 550 nm abstrahlt, da hier die Empfindlichkeit der Kamera höher ist (Abbildung 29).

Das Programm kann noch um einen sogenannten *Filesink–Filter* erweitert werden, um Videos direkt abspeichern zu können oder die Analysegeschwindigkeit durch interne Umstrukturierung verbessert werden. Um auch gleichzeitig auf-

tretende Signale unterschiedlicher Helligkeit zu erkennen, kann der Gradienten–Algorithmus noch mitbenutzt werden. Auch wäre es hier sinnvoll später Signale mit Beispiel–Ionendaten fitten zu können.

Ausblick

An den Sensor selbst wird noch das in Kapitel 2.3 erwähnte Peltier–Kühlsystem angeschlossen. Damit wird es möglich sein, Temperatureffekte, die sich negativ auf das durch die Kamera erkanntes Bild auswirken (Kapitel 4.1), entgegenwirken zu können. Auch wird es dann möglich sein, die Strahlkamera in Aufbauten einzusetzen, die den CCD-Sensor sonst zu hohen Temperaturen aussetzen würden.

Wurden die Elemente eingebaut ist die Verwendung im AmbiProbe–MR–TOF–MS (Kapitel 2.2) geplant. Da dort die Vakuumkammer räumlich stark eingeschränkt ist, wird mithilfe der Kenntnis von Lage und Durchmesser der Ionenstrahlen die Ionenstrahlführung optimiert. Auch lässt sich mit diesen Daten der Analysator gut auf einen Phasenvorschub von 180 Grad einstellen.

A. Literaturverzeichnis

[BG94] Blasse, G. ; Grabmaier, B.C.: *Luminescent Materials*. Springer Verlag, 1994

[BPSM+97] Bray, T. ; Paoli, J. ; Sperberg-McQueen, C.M. ; Maler, E. ; Yergeau, F.: Extensible markup language (XML). In: *World Wide Web Journal* 2 (1997), Nr. 4, S. 27–66

[Dic10] Dickel, Timo: *Design and Commissioning of an Ultra-High-Resolution Time-of-Flight Based Isobar Separator and Mass Spectrometer*, JLU Gießen, Dissertationen, 2010

[Ebe09] Ebert, Jens: *Aufbau und Inbetriebnahme einer Strahlkamera für die Massenspektrometrie*, JLU Gießen, Bachelorarbeit, 2009

[GS06] Gumm, H.P. ; Sommer, M.: *Einführung in die Informatik*. Oldenbourg Wissenschaftsverlag, 2006

[Gue08] Guetlich, Eiko: *Untersuchungen zu Leuchtschirmen für die Hochstrom–Strahlprofilmessung am UNILAC der GSI*. Diplomarbeit. Fachhochschule Wiesbaden, 2008

[HS00] Hollingworth, Jarrod ; Swart, Bob o.: *C++ Builder 5 developer's guide*. Pap/Cdr. Sams, 2000

[L+12] Lang u. a.: Developement and Commissioning of Multiple-Reflection Time-of-Flight Mass Spectrometer for in-situ Measurements. In: *poster on DESORPTION 2012 conference* (2012)

[LMLM12] Löffler-Mang, Martin ; Löffler-Mang, Martin: *CCD-Chip*. Vieweg+Teubner Verlag, 2012

[Mey06] Meyers, S.: *Effektiv C++ programmieren*. Pearson Deutschland GmbH, 2006

[P+12] Plass u. a.: Jahresbericht des II. Phys. Inst. In: *JLU Gießen* (2012)

[PDE04] Plass, Wolfgang ; Dickel, Timo ; Ebert, Jens: *Massenspektrometrie und Spurenanalytik*, JLU Gießen, Versuchsanleitung, 2004

[Pet11] Petrick, Martin: Kühlen und Heizen mit Peltierelementen für den Einsatz in Vakuumumgebung. In: *Universität Giessen* (2011)

[Rin12] Rink, Ann-Kathrin: *Ionenstrahl-Kamera*, JLU Gießen, Dokumentation zum Vertiefungsmodul, 2012

[Roe99] Roelofs, G.: *PNG: The Definitive Guide*. 1999

[Wiz79] Wiza, Joseph L.: *Microchannel Plate Detectors.* Vol. 162. Nucl. Instrum. Methods, 1979

[WM55] Wiley, W. C. ; McLaren, I. H.: Time-of-Flight Mass Spectrometer with Improved Resolution. In: *Rev. Sic. Instrum.* 26 (1955), Nr. 12, S. 1150–1157

Alle Bilder ohne ausdrückliche Quellenangabe in der Bildbeschreibung wurden von mir eigenständig angefertigt (Als Bildschirmabbild, Photo oder mithilfe von Inkscape®).

B. Datenblätter

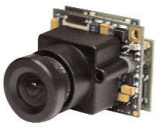
VIDEOLOGY®
IMAGING SOLUTIONS INC.
www.videologyinc.com

High Resolution USB 2.0 B&W Board Camera

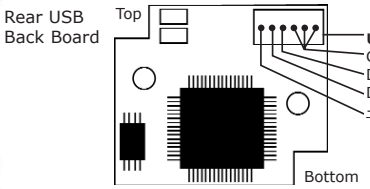
20K13XUSB / 21K13XUSB

Camera Features

- **All-Digital Design Uses Camera Board's Digital (D1) Output For Input To USB Board**
- **Sony Ex-View Sensor® Provides Highest Sensitivity & Performance**
- **USB 2.0 Bus Provides Power To Camera**
- **Camera Control Over USB**
- **Software**
  - WDM device drivers Digitally signed by Microsoft® under Windows XP®
  - DirectX compliant
  - TWAIN compliant



20K135USB Shown
22mm x 26mm x 32mm w/M-12 Board Mount



Videology's WDM drivers are digitally signed by Microsoft® certifying the software has been tested with Windows XP® and has not been altered since testing, enabling compatibility with your applications.

Technical Specifications

Electrical

	20K13XUSB EIA	21K13XUSB CCIR
CCD Sensor	1/4" Sony Ex-View® CCD	
Active pixels (HxV)	768 x 494	752 x 582
Viewer Display **(USB 2.0, 30fps)**	720 x 480 pixels max	720 x 576 pixels max
(USB 1.1, 30fps)	320 x 240 pixels max	
Sensitivity	< 0.005 lux, near IR sensitive	
Signal to noise ratio	> 48 dB (AGC off)	
Gamma	0.45 default (1.0 via Software)	
Gain Control	Automatic 36 dB (AGC default) or Fixed options via software	
Scan Mode	Interlaced / Non Interlaced (selectable via Software)	
Mirror Mode	Selectable via software	
Synchronization	Internal	
Back light compensation	Default on (selectable via software)	
Contour enhancement	Default on	
Iris Control	CCD Iris default	
Shutter Speeds	Automatic from 1/60 to 1/100,000	Automatic from 1/50 to 1/100,000
	14 fixed speeds via software	
Control Communication	Camera control via USB bus	
Power supply	5VDC via USB bus	
Power consumption	< 1.3 W	

Mechanical

Dimensions (WxHxD)	26mm x 22mm x 16mm (without lens mount)
Lens mounts	Replace "X" in model number with desired lens mount option:
Example: Change 20K13XUSB to 20K135USB to select an M-12 Board Mount.	2 = Metal M-12 Pinhole; 5 = Metal M-12 Board; 8 = Metal CS Board
Connectors	Camera to USB board: Board to board connector
	USB board connector: (6 pin) power & data

Software

Software for entire product family: • WDM device drivers *Digitally signed by Microsoft® under Windows XP®* • DirectX compliant • TWAIN compliant	
Full camera DSP control available for OEMs	
Data rate	480 Mb/sec. (maximum) Distance 5m W/repeater 30m
Installation software SFT-04040	Camera Device Driver Viewer (GUI) User's Manual
SDK SFT-04040-SDK	Complete software development kit and support for OEMs

Environmental

Ambient operating temp.	-15° C to + 55° C
Ambient operating Humidity	20 up to 93%RH
Storage temp.	-25° C to + 70° C
Storage Humidity	Up to 98%RH
Lifetime	MTBF > 150000
Packaging	ESD safe package

Complementary Models

20K14XUSB	Standard resolution color USB 2.0 board camera
20K14XUSB-DN	USB 2.0 Day / night color board camera
	Note: Day/Night optical filter (or no filter) options for maximum sensitivity in B&W mode
20K15XUSB	High resolution color USB 2.0 board camera

06/27/07 PDS-20K135USB Rev D

All data subject to change without notice.

Videology® Imaging Solutions, Inc.
37M Lark Industrial Parkway
Greenville, Rhode Island 02828 USA
Tel: (401) 949 - 5332 Fax: (401) 949 - 5276
North/South American Sales: Sales@videologyinc.com

Videology® Imaging Solutions, Europe B.V.
Neutronenlaan 4
NL-5405 NH Uden, Netherlands
Tel: +31 (0) 413 256261 Fax: +31 (0) 413 251712
European Sales: Info@videology.nl

Abbildung 27

Datenblatt zur verwendeten Kamera. Bezeichnung des Kameramodells ist irreführend, das aktuelle Modell heißt 21K35USB–C. Bis auf einen geringeren Energieverbrauch und mehr Einstellungsmöglichkeiten ändert sich hier allerdings nichts.

KPI

PHOSPHOR SCREENS
RUGGED and UHV P22 PHOSPHOR

FOR USE IN:

BEAM DETECTION
BEAM ALIGNMENT
LENS TESTING
UNIFORMITY TESTING
SURFACE PHYSICS
UHV EXPERIMENTS

FEATURES / OPTIONS:

HIGH LUMINOSITY P22 PHOSPHOR
SENSITIVE TO ELECTRONS,
STARTING AT 500 eV
EASY HANDLING / HIGH ABRASION
RESISTANCE (RUGGED SCREENS)
UHV COMPATIBLE, BAKEABLE
METAL or GLASS BACKING
STANDARD SIZES MATCH eV Parts®
RAPID AVAILABILITY, LOW COST

Kimball Physics Phosphor Screens are made of high luminosity P22 phosphor (ZnS:Ag). The screens are sensitive to electrons starting at approximately 500 eV with a threshold of 1 x 10^{-7} A/cm^2 at 500 eV. The maximum recommended input beam power density is 1 Watt/cm^2. Two general types of phosphor screens are made by Kimball Physics: UHV screens and Rugged screens.

Rugged Phosphor Screens (designated by RP22) are easy to handle and unusually resistant to mechanical damage and rough handling. Ruggedized Screens are bonded to their metal or glass backings using a bonding agent which has a low but non-zero vapor pressure. Due to the binder, the rugged phosphor is only suitable for vacuum pressures down to 10^{-8} torr (at the lower end of this operating range, some outgassing may be observed). The rugged screens are particularly suited for use in experimental vacuum systems. Rugged screens have a phosphor thickness of approximately 75 μm.

UHV Phosphor Screens (designated by UP22), with no binder, are compatible with vacuums better than 1 x 10^{-8} torr. Although these screens are better for ultra high vacuum, they are more fragile and require greater care when handling; since the phosphor coating is delicate and can be easily destroyed. Damage will result from touching a UHV Phosphor coating; shock from knocking or dropping the screen may cause the phosphor coating to flake off. Standard UHV screens are shipped with a stainless steel protective cover. The thickness of the UHV phosphor ranges from 50 μm to 70 μm.

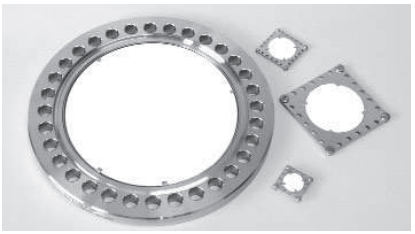
The phosphor can be deposited either on a 0.012 or 0.025 inch thick 304 stainless steel plate (SS) or on 0.030 inch thick conductive glass (GL) fitted into a stainless plate for mounting. The screens deposited on metal must be viewed from the electron impact side (since the metal is opaque). The screens deposited on glass and held in a metal frame, can be viewed from either side (since the glass is transparent).

Custom Phosphor Screens can be deposited on almost any metal, glass, or ceramic surface on a special order basis. Round screen diameters can range from 10 mm diameters, up to more than 200 mm in diameter. Rugged phosphor screens can be made in a variety of custom sizes and shapes; rectangles, strips, and patterned shapes are possible; edge definition is better than 100 micrometers. Custom screen can be made up even in quantities of one. Contact Kimball Physics Engineering for more information.

Toll Free: 1 888 KIM-PHYS (1 888 546-7497) e-mail: info@kimphys.com
Tel. 1 603 878-1616 Fax: 1 603 878-3700

PHOSPHOR:

PHOSPHOR TYPE:	ZnS:Ag Type 1330 (P-22 Blue)
SATURATION THRESHOLD:	3 x 10^{-2} Amps/cm^2
PEAK EMITTED WAVELENGTH:	450 nanometers
MAXIMUM INPUT POWER DENSITY:	1 Watt/cm^2
MINIMUM POWER DENSITY:	5 x 10^{-5} Watts/cm^2



6 inch CF thin flange-mounted UHV Phosphor Screen and three standard Phosphor Screens on 0.7, 1.4, and 2.0 inch squares

NOTES:

1) Using the phosphor screens at electron energies below the first unity-secondary-emission crossover point may cause the screen to charge up to electron cathode potential, at which point the screen temporarily goes out.

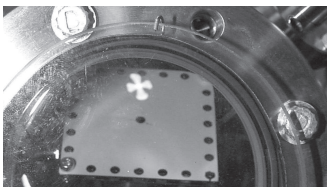
2) When using the phosphor screen, input power density should not exceed 1 Watt/cm^2, or the phosphor coating may be damaged To preserve screen brightness, it is advisable to use the lowest beam power density that provides a clear spot. Normal usage will result in gradual browning of the screen.

3) Ruggedized screens are bakeable up to 200°C; UHV screens are bakeable up to 350°C.

4) Larger screens, mounted in either six or eight inch viewports are also available. The diameter of a screen for a six inch viewport would be 4.2 inch and the diameter of a screen for a eight inch viewport would be 6.2 inch. These larger phosphor screens can be deposited on leaded glass if required.

5) Rugged screens on stainless steel are deposited directly on the square plate. UHV screens on stainless steel are deposited on a round stainless steel plate that is affixed to the square mounting plate by four equally-spaced tabs spot-welded to the square mounting plate and to the underside of the round (phosphor-coated) plate.

6) Standard phosphor screens deposited on conductive glass (both Rugged and UHV) are held between two stainless steel plates, within a center hole, by four equally-spaced tabs on both sides of the screen. The tabs, which are spot-welded to the stainless steel plates, also serve to bleed off charge from the screens.



Maltese-cross shaped spot from an unfocused LaB_6 cathode seen on a custom phosphor screen inside the vacuum chamber

KIMBALL PHYSICS INC. KPI

Abbildung 28
Datenblatt zum verwendeten Phosphorschirm.

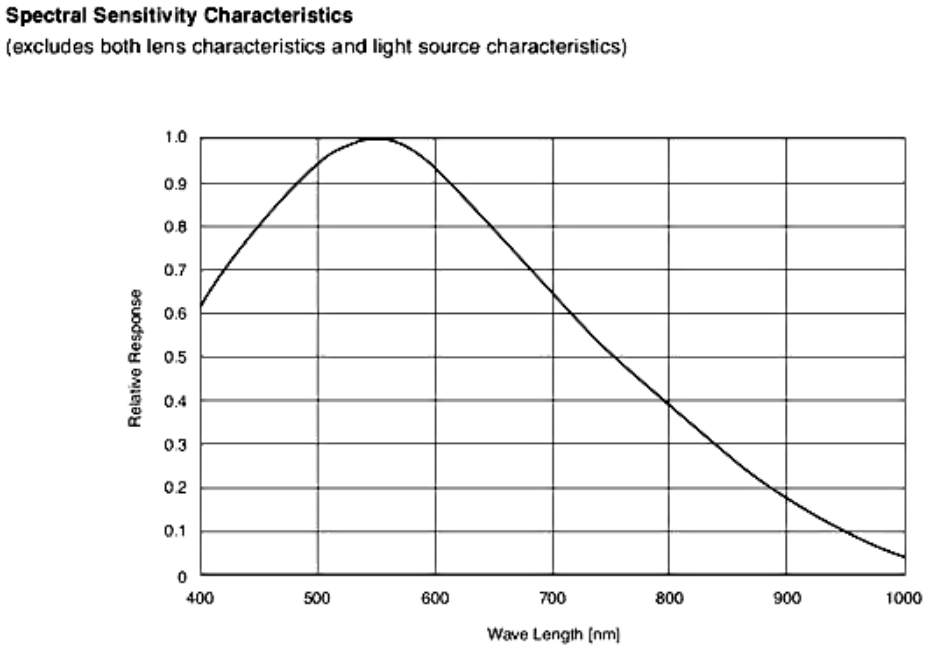


Abbildung 29
Farbraum-Empfindlichkeit der verwendeten Kamera.

C. Bildschirmabbilder zu Programm–Funktionen

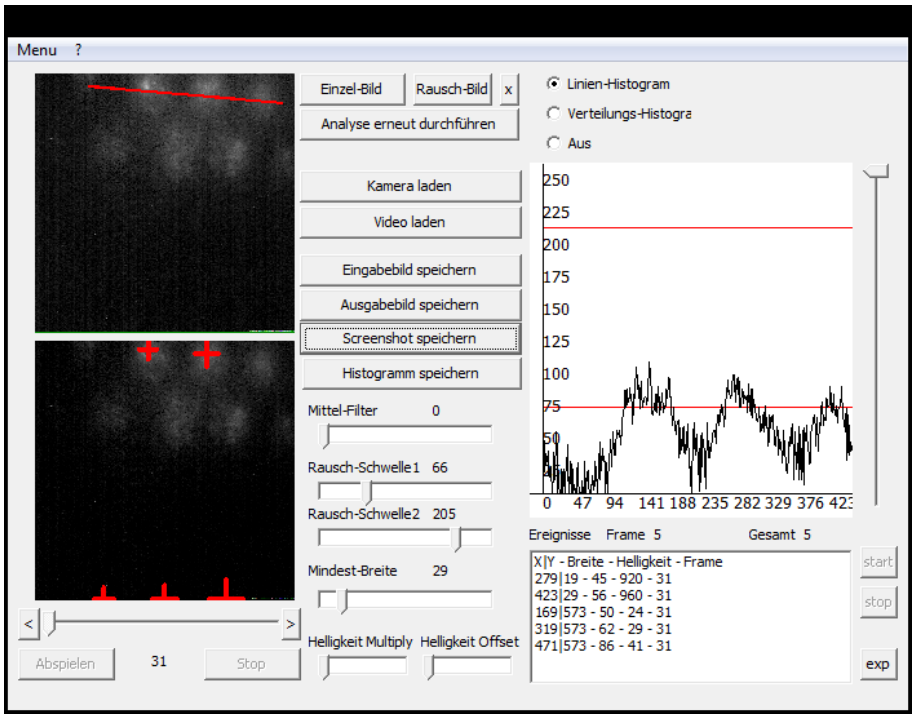


Abbildung 30
Screenshot: Signalanalyse nur mit Rauschabzug und gleichzeitige Histogrammanzeige des rot eingezeichneten Bereiches

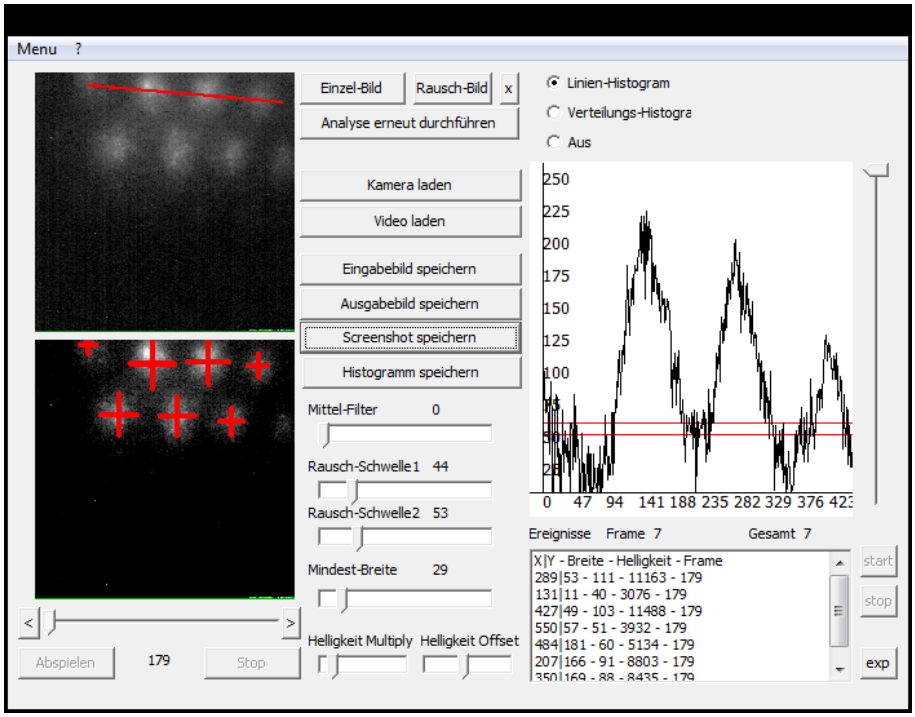


Abbildung 31

Screenshot: Signalanalyse mit Kontrast, Helligkeit (ohne Übersteuerung) und Rauschabzug und gleichzeitige Histogrammanzeige des rot eingezeichneten Bereiches

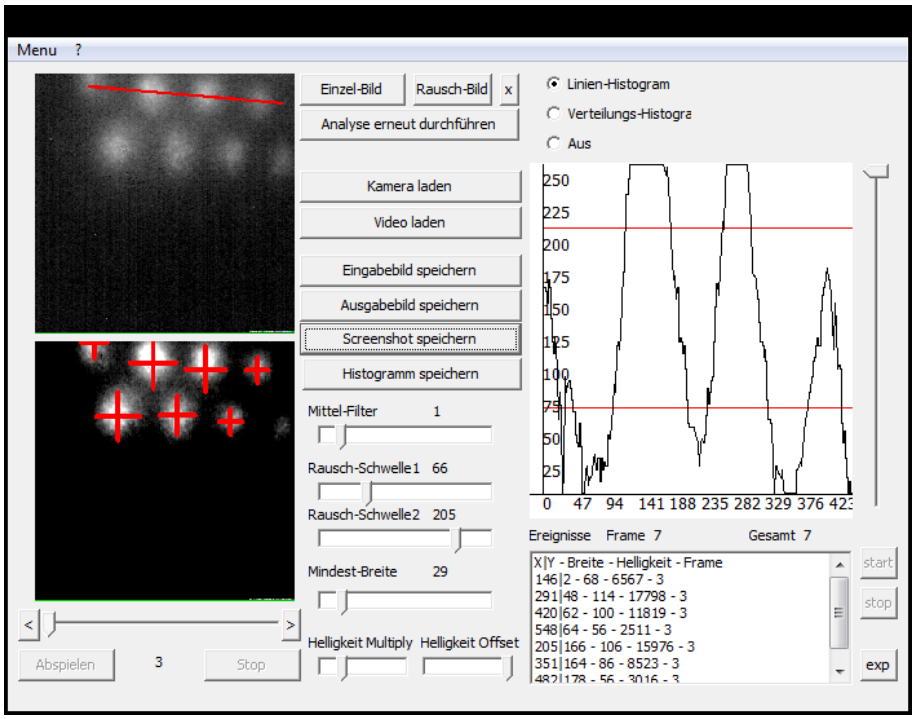


Abbildung 32

Screenshot: Signalanalyse mit Kontrast, Helligkeit, Weichzeichner und Rauschabzug und gleichzeitige Histogrammanzeige des rot eingezeichneten Bereiches

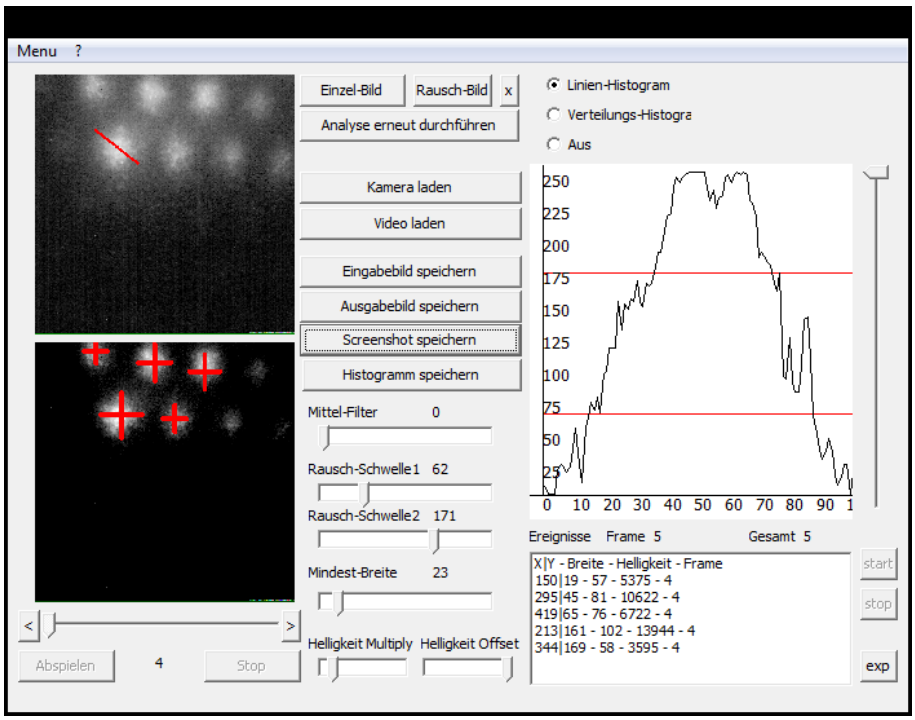


Abbildung 33
Screenshot: Signalanalyse und Histogramm für einzelnes Signal

```
Videomodus:
Fri Aug 31 14:23:18 2012


Mittelwertfilterstärke    0
Schwelle1    90
Schwelle2    205
Mindesbreite    33
Helligkeits    Multiply    3
Helligkeits    Offset    6
Anzahl    Events    6

X    Y    Breite    Helligkeit    Frame
288    35    91    15867    36
427    39    76    10360    36
196    155    67    6581    35
352    167    71    4191    35
489    183    35    3184    33
565    571    40    8423    33
```

Abbildung 34
Beispiel einer Messaufzeichnung in einer Textdatei

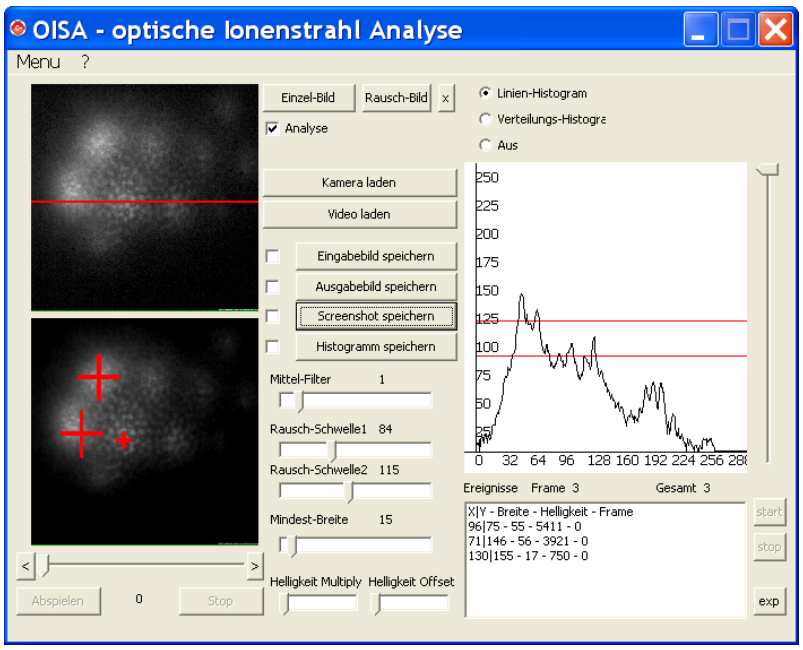


Abbildung 35

Aufnahme mit Signalen stark unterschiedlicher Intensität (Probleme für Schwellenwert–Algorithmus)

D. Quelltext

Im folgenden wird der C++–Quellcode zu den in Kapitel 3.4.3f genannten Algorithmen angegeben. Es handelt sich dabei nicht um den kompletten Programm–Quellcode, sondern lediglich um die 5 wichtigsten Funktionen.

Quellcode 1

Signalerkennung

```
void TForm1::analyse(){
   try{
   if(!img2)return;//Ohne Eingabebild keine Analyse

   delete img; //neuzuweisung pointer
   img=new TPngImage();
   img->Assign(img2); //kopiert img2 in img. img2 ist Quelle.

   if(!Timer1->Enabled)Label14->Caption=0;
   //Wenn keine Langzeitmessung, dann gesamtzahl Events immer neu gesetzt

   rauschabzug(img,rausch); //ziehe pixelwerte rausch von img

   plusmedian(img,TrackBar1->Position); //weichzeichnen
   erhellung(TrackBar3->Position,TrackBar6->Position*20,img); //Helligkeit/Kontrast
        digital

   if(!messsummation)ev.clear(); //Leere die aufgezeichneten Events.
   //Bei Messungen über mehrere Bilder sollte dies nur bedingt ausgeführt werden.

   for (int j=0;j<255;++j){histoverteil[j]=0;} //setze die Hüufigkeit der grauwerte
        auf 0 zurück

   Byte *prt;//Pointer auf Bytearray entsprechend BGR-Code in einer Zeile.
   int schwelle1=TrackBar2->Position,schwelle2=TrackBar8->Position; //Grauwert, ab
       dem ein Event erkannt wird:0-255
   int forschritt=1; //Idee der Beschleunigung: Scanne nur jede 2. Zelle,
       allerdings stark Fehleranfüllig
   float gewicht; //Fuzzy-Logik, Helligkeit wird zwischen den beiden schwellen
       gewichtet
   list<coord>::iterator zwiakt;
   coord *pkt=new coord(0,0,0,0,aktdatei); //aktuell erkanntes Ereignis
   double hellmax=0; //speichert maximale Helligkeit in Zeile, um Breite bei
       Randevents zu rekonstruieren
   int hellmaxx=0; //x-koordinate der max. Helligkeit in Zeile in RandEvents
   for(int y=0;y<img->Height-1;y+=forschritt){
      prt=(Byte *)img->Scanline[y]; //lese Bytearray der y-Zeile ein.
      hellmax=0;hellmaxx=0; //helligkeit und x-Position
      for(int x=1;x<img->Width-1;x+=forschritt){
         histoverteil[(int)grau(prt,x)]+=1; //erhühe Zühler für aktuellen grauwert.
         int mittelgrau=0;
         mittelgrau=grau(prt,x);//+grau(prt,x-1)+grau(prt,x-2);
         if(mittelgrau>schwelle2){
            //Event detektiert, breite und integral wird gespeichert
            pkt->breite+=1;
            pkt->helligkeit+=mittelgrau;
            if(mittelgrau>hellmax){hellmax=mittelgrau;hellmaxx=x;}
         }else if((mittelgrau<=schwelle2)&&(mittelgrau>=schwelle1)){
            //Fuzzy-Logik für weiche Signalkanten, Helligkeit wird linear gewichtet
```

```
    gewicht=float(mittelgrau-schwelle1)/float(schwelle2-schwelle1);
    pkt->breite+=1;
    pkt->helligkeit+=mittelgrau*gewicht; //helligkeit in Fuzzy-logik
        zwischen den schwellen linear gewichtet
    if(mittelgrau*gewicht>hellmax){hellmax=mittelgrau*gewicht;hellmaxx=x;}
}
if((mittelgrau<schwelle1||x==img->Width-2)&&pkt->breite>0){
    if(x==img->Width-2&&hellmax>x-pkt->breite){pkt->breite=(hellmaxx-(img->
        Width-1-pkt->breite))*2;}
    if(x-pkt->breite<0){pkt->breite=(x-hellmaxx)*2;}
    if(pkt->helligkeit>0&&pkt->breite>TrackBar7->Position){
        //rechte Eventgrenze wurde erreicht, Event wird ggf. zur Liste
            hinzugefügt
        pkt->x=x-(pkt->breite/2);pkt->y=y; //Eventursprung in Zeile in Mitte
             der Breite

        //Falls Rechnung Bildüberschreitungen ergibt, werden Koordinaten
            abgeschnitten:
        //Auskommentiert, da Evnts auch im Randbereich gut entdeckt werden
        //if (pkt->x>=img->Width)pkt->x=img->Width-1;
        //if (pkt->y>=img->Height)pkt->y=img->Height-1;
        //if (pkt->x<0)pkt->x=0;
        //if (pkt->y<0)pkt->y=0;

        //Abbruch bei zu vielen detektierten Events.1-10 plausibel, 100
            Spielraum nach oben.
        if(ev.size()>500)goto eventabbruch;
        //find ist hier auch richtig, wenn der Punkt um b/2 daneben liegt,
            um naheliegende
        //Event-Trigger (durch Rauschen etc. verzerrt) einem gemeinsamen
            Event zuzuordnen.
        zwiakt=find(ev.begin(),ev.end(),*pkt);

        if(zwiakt==ev.end()){ //neues Signal
           ev.push_back(*pkt);
        }else{
        //Es gibt bereits Signale im Signalbreiten-quadrat oder dies ist
            Signal in existierenden Breitenquadrat

           if((pkt->frame-zwiakt->frame==0&&zwiakt->breite<pkt->breite)){
              //neues Event im selben Frame ist breiter
              zwiakt->x=pkt->x;
              zwiakt->y=pkt->y;
              zwiakt->breite=pkt->breite;
              zwiakt->helligkeit=pkt->helligkeit;
           }else if((pkt->frame-zwiakt->frame>=1&&pkt->frame-zwiakt->frame<=
               settings.eventframes)&&!listalt){
              //neues Frame hat Event an gleicher Position wie letztes
              //Funktioniert effektiv nur bei normalem Abspielen bzw. Kamera
              //Bei Navigation und Sprung wird neu detektiert
              ListBox1->Items->Delete(ListBox1->Items->IndexOf((String)
                  zwiakt->x+"|"+zwiakt->y+" - "+int(zwiakt->breite)+" - "+
                  int(zwiakt->helligkeit)+" - "+zwiakt->frame));
              ev.remove(*zwiakt);
              ev.push_back(*pkt);
           }
        }
    }
```

```
            delete pkt; //Speicherfreigabe und detektion des nüchsten Ereignisses
            pkt=new coord(0,0,0,0,aktdatei);
         }
      }
      delete pkt; //Ende der Zeile erreicht.
      pkt=new coord(0,0,0,0,aktdatei);
   }
   delete pkt;
   eventabbruch: //Bei 100 detektierten Events wird detektion abgebrochen und
       analysiert.
   TrackBar4Change(TrackBar4); //Histogramanalyse vom aktuell ausgewühlten anzeigen
       .
   TPngImage *img3=new TPngImage(); //neues Image aus verwertetem Bild um Treffer
       rot zu markieren.
   img3->Assign(img);
   list<coord>::iterator aktev = ev.begin();
   if(!Timer1->Enabled&&!listalt&&!messsummation){
      //Liste wird auüerhalb der summation immer neu erstellt
      ListBox1->Items->Clear();
      ListBox1->Items->Add("X|Y - Breite - Helligkeit - Frame");
   }
   int aktframecount=0;
   for(aktev=--ev.end();aktev!=--ev.begin();--aktev) {
      if(aktev->frame==aktdatei){
         ++aktframecount;
         img3->Canvas->Pen->Color=clRed;
         img3->Canvas->Pen->Width=img3->Width/50;
         img3->Canvas->MoveTo(aktev->x-aktev->breite/2,aktev->y);
         img3->Canvas->LineTo(aktev->x+aktev->breite/2,aktev->y);
         img3->Canvas->MoveTo(aktev->x,aktev->y-aktev->breite/2);
         img3->Canvas->LineTo(aktev->x,aktev->y+aktev->breite/2);
         //innerhalb der Messung werden neue Events an erster Stelle geschrieben,
             wenn sie im aktuellen Frame ankamen
         if(messsummation)ListBox1->Items->Insert(1,(String) aktev->x+"|"+aktev->y+
             " - "+int(aktev->breite)+" - "+int(aktev->helligkeit)+" - "+aktev->
             frame);
      }
      //listalt im videomodus lüsst zu Frame springen, daher wird hier das
          Neuzeichnen unterdrückt.
      //auüerhalb listalt und messummation wird die liste mit allen Events normal
          erstellt
      if(!messsummation&&!listalt)ListBox1->Items->Insert(1,(String) aktev->x+"|"+
          aktev->y+" - "+int(aktev->breite)+" - "+int(aktev->helligkeit)+" - "+
          aktev->frame);
   }
   Image2->Picture->Graphic=img3; //Anzeige gefiltert, rot markiert
   Label2->Caption=aktframecount; //Anzahl Events
   Label16->Caption=aktdatei;
   Label14->Caption=(ListBox1->Items->Count-1); //Anzahl Events
   delete img3;
   }catch(...){}
}
```

Quellcode 2

Abziehen eines Raschbildes

```
void rauschabzug(TPngImage *bild, TPngImage *raus){
   try{
   if (!raus)return ; //Ohne Rausch kein Rauschabzug
   TColor neup=0; //neuer Farbwert für jedes Pixel
   //Scanline-pointer zeigen auch Bytearray entsprechend den Pixelfarben einer
       Zeile des Bildes
   Byte *lin, *rlin;
   for(int y=0;y<bild->Height-1;y+=1){
      lin=(Byte*)bild->Scanline[y];
      rlin=(Byte*)raus->Scanline[y];
      for(int x=0;x<bild->Width*3-4;x+=1){ //3 Byte-Elemente pro Pixel BGR
         neup=lin[x]-rlin[x]; //einfacher Rauschabzug
         if(neup<0)lin[x]=0;else lin[x]=neup;
         /*unterschreitet Farbwert 0, wird das negativ-byte im bytearray
             verschluckt, also
         wird das Bild an diesen stelen weiü. Um dies zu verhindern, wird die
             untere
         Farbwertgrenze auf 0 gesetzt*/
      }
   }
   }catch(...){}
}
```

Quellcode 3

Kanalzusammenfassung und Weichzeichner

```
void plusmedian(TPngImage *eingabe,int intens){
   try{
   if(intens==0)return; //keine Mittlung
   TPngImage *img3= new TPngImage; //Dieses Bild wird verkleinert und dann
       weichgezeichnet.
   img3->Assign(eingabe);//Kopiert eingabe in img3
   TPngImage *img4= new TPngImage; //Zwischenablage um das Bild wieder auf ursprü
       ngliche Grüüe zu skalieren.
   img4->Assign(eingabe);
   float ratio=1+(intens-1)/4 + intens*eingabe->Width/500;
   /*Das Bilddimension wird durch ratio geteilt zum verkleinern.
   Daher min. 1. intens-1/4 ist die abhüngigeit von der eingestellten
       Weichzeichnerstürke.
   Bei groüen Bildern ist direkte Pixelweichzeichnung kaum erkennbar, daher noch
       abhüngig von der Bildbreite*/
   int orw=img3->Width; //urprüngliche Grüüe
   int orh=img3->Height;
   TRect *rec=new TRect(0,0,orw/ratio,orh/ratio); //Grüüe auf die weichgezeichnet
       wird.
   TRect *orec=new TRect(0,0,orw,orh); //Originalgrüüe auf die spüter wieder
       skaliert werden soll
   img3->Draw(img3->Canvas,*rec); //Bild wird skaliert.
   //Gründe: 1. Performance, da Laufzeit breite*hühe ist. Dies führt zu Laufzeiten
       von 1-2s bei 500*700px.
   // Der aktuelle Algoritmus ist grüütenteils unabhüngig von Grüüe und benütigt
       ~0.05.
   img3->SetSize(orw/ratio,orh/ratio);
// img3->Draw(img3->Canvas,*rec);
   Byte *prt1=0,*prt2=0,*prt3=0; //3 pointer für jeweils 3 Zeilen des Bildes.
   for(int y=0;y<img3->Height-1;++y){
```

```
      prt1=(Byte *)img3->Scanline[y];//aktuelle Zeile
      if(y>0)prt2=(Byte *)img3->Scanline[y-1]; //Zeile vorher
      if(y<eingabe->Height-1)prt3=(Byte *)img3->Scanline[y+1]; //Zeile nachher
      for (int x=0;x<img3->Width-1; ++x) {
         int wert =grau(prt1,x)*2; //Zellenmittengewichtung von 2.
         //Matrix entspricht: {{010}{121}{010}}
         int n=2; //n ist gesamtgewicht, anzupassen fals am Bildrang, sonst 4*1+2
         if(x<img3->Width){wert +=grau(prt1,x+1);++n;}
         if(x>0){wert +=grau(prt1,x-1);++n;}
         if(y<img3->Height-1){wert +=grau(prt3,x);++n;}
         if(y>0){wert +=grau(prt2,x);++n;}
         wert/=n; //wert wird gemittelt
         prt1[x*3]=wert; //selber grauwert für alle Farbanteile
         prt1[x*3+1]=wert;
         prt1[x*3+2]=wert;
      }
   }

   img3->Draw(img4->Canvas,*orec);//skaliere Bild wieder auf Orignalgrüüe
   img4->AssignTo(eingabe); //schreibe zwischenspeicher wieder auf eingabe.
       direktes Draw funktioniert nicht.

   delete rec; delete orec;
   delete img3;
   delete img4;
   }catch(...){}
}
```

Quellcode 4

Kontrast– und Helligkeitsanpassung

```
void erhellung(int multiply,int offset, TPngImage *zimg){
   try{
   Byte *ptr1=0,*ptr2=0;
   TPngImage *ruck=new TPngImage;//Rückgabebild, wird am ende in zimg geschoben
   if(!zimg)return;
   ruck->Assign(zimg);
   for(int y=0;y<zimg->Height;++y){
      ptr1=(Byte*)zimg->Scanline[y]; //hole y-Zeile
      ptr2=(Byte*)ruck->Scanline[y];
      for(int x=0;x<zimg->Width*3;++x){ //gehe alle pixel und dort 3 Farbwerte in
          Bytes durch
         int s=ptr1[x]*multiply-offset; //Kontrast und Helligkeit
         if(s>255)s=255; //Abschnitt bei Weiü
         if(s<0)s=0; //Schwarz
         ptr2[x]=s;
      }
   }
   ruck->AssignTo(zimg);//Rückgabe
   delete ruck;
   }catch(...){}
}
```

Quellcode 5

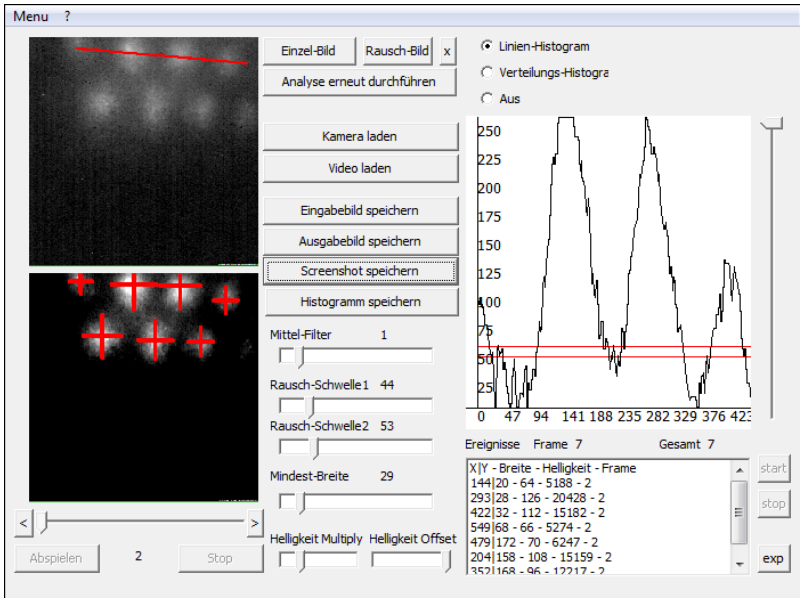
Ziehen einer Linie: Bresenheim Algorithmus

```
void line(TPngImage *im, int x0, int y0, int x1, int y1, TColor col)
{
  histopixel.clear();
  int dx =  abs(x1-x0), sx = x0<x1 ? 1 : -1;
  int dy = -abs(y1-y0), sy = y0<y1 ? 1 : -1;
  int err = dx+dy, e2;
  coord *pkt=new coord(0,0,0,0,0);
  for(;;){
   if (x0==x1 && y0==y1) break;
   e2 = 2*err;
   if (e2 > dy) {err+=dy;x0+=sx;}
   if (e2 < dx) {err+=dx;y0+=sy;}
   if(e2>dy || e2<dx){
      pkt->x=x0;pkt->y=y0;pkt->helligkeit=dgrau(img->Pixels[x0][y0]);
      histopixel.push_back(*pkt);
   }
  }
}
```

Anleitung zum Programm

OISA

optische Ionenstrahl Analyse



Inhaltsverzeichnis

1 Installation **49**
1.1 Mindestanforderungen 49
1.2 Einrichtung 49

2 Programm–Oberfläche **50**

3 Hinweise zur Verwendung **52**

1 Installation

1.1 Mindestanforderungen

Das Programm wurde erfolgreich unter den Betriebssystemen Windows XP, Windows Vista und Windows 7 getestet. Eine Emulierung in einem Unix-System, etwa mit Wine, ist möglich. Es benötigt etwa 20 MB RAM und 3 MB Festplattenspeicher[1]. Das Anschließen der Kamera über einen USB-Anschluss wird vorausgesetzt.

1.2 Einrichtung

OISA benötigt keine Zusatzbibliotheken oder Registry-Zugriffe, ist also als sogenannte Standalone–Exe ohne Installation einsatzfähig.

Die Einstellungen des Programms werden beim Schließen automatisch in eine xml-Datei desselben Pfades gespeichert. Hat das Programm keinen Schreibzugriff dort (Systempfad oder gebrannte CD), werden bei jedem Start die Standard–Einstellungen geladen.

[1] Benötigter Speicherplatz zur Abspeicherung von Bildern nicht einbezogen

2 Programm–Oberfläche

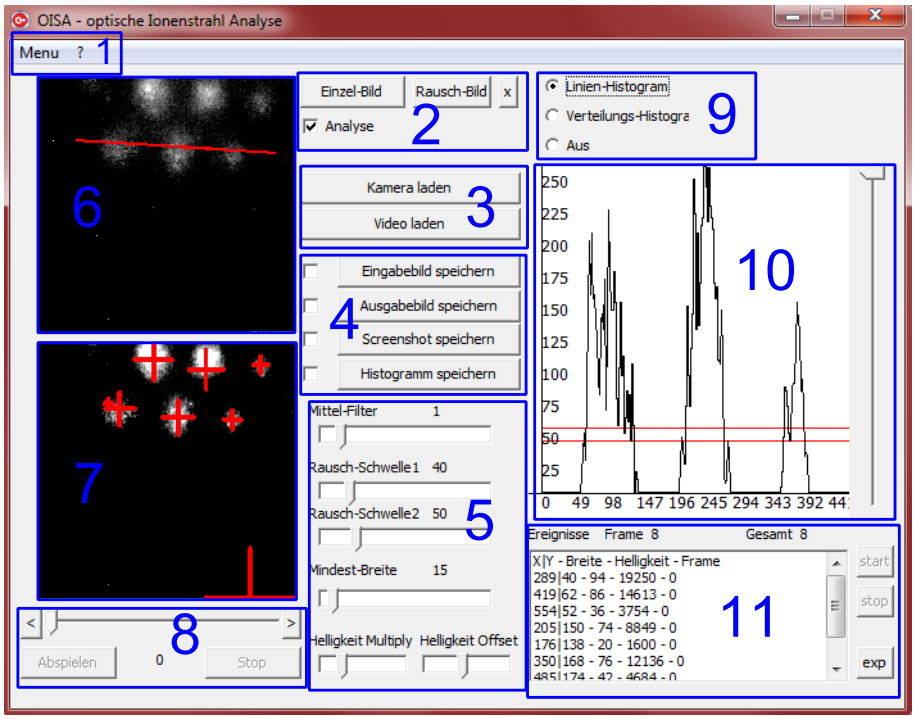


Abbildung 37
Haupt–Oberfläche des Programms

Beim Start des Programms öffnet sich die in Abbildung 37 zu sehende Haupt–Oberfläche. Zur leichteren Erklärung wurden bestimmte Bereiche in dieser Anleitung zusammengefasst und mit Nummern bezeichnet.

1. *Menü.* Hier können Einstellungen vorgenommen und diese Anleitung aufgerufen werden. Einstellungen werden beim nächsten Auslese–Start wirksam und bei Beendigung des Programms abgespeichert. Dazu gehören Programm–Einstellungen, Kamera–Einstellungen und der Pfad, in dem Bilder automatisch abgespeichert werden sollen.
2. *Einzelbild–Operationen.* Mit der ersten Schaltfläche können einzelne Bilder eingelesen werden, um sie zu analysieren.
   Die 2. lässt ein Leerbild auswählen, das von allen analysierten Bildern abgezogen wird.
   Die mit „X“ gekennzeichnete Schaltfläche entfernt dieses Leerbild wieder.
   Ist die Kontrollbox „Analyse“ deaktiviert, werden Bilder lediglich eingelesen, jedoch nicht verbessert oder analysiert.
3. *Mehrbild–Operationen.* Der obere Button sucht nach der Kamera „*Videology USB-C Camera*“ und startet Wiedergabe und Analyse.
   Der untere Button lässt MPEG1–Videos einlesen und startet Wiedergabe und Analyse des Videos.
4. *Bild–Export.* Mithilfe der Schaltflächen kann das aktuell im bezeichneten Bereich dargestellte Bild in eine PNG-Datei exportiert werden. Das Eingabebild wird dabei ohne die rote Linie exportiert und kann beispielsweise als Leerbild für den Rauschabzug benutzt werden. Aktiviert man die

daneben angezeigten Kontrollboxen, wird die Operation in jedem Frame ausgeführt. Dabei wird das Bild mit aktueller Uhrzeit und einer Laufvariable im *Auto–Screenshot–Pfad* abgespeichert. Dieser kann im Menü vorher festgelegt werden.

5. *Parameter. Mittel–Filter* steht für die Pixelzusammenfassung und Weichzeichnung.
   Rausch–Schwelle1 gibt die 1. Schwelle der Signalerkennung an. Ab hier wird die Breite gezählt und das Signal als solches erkennt.
   Rausch–Schwelle2 gibt die Fuzzy-Schwelle an, ab der Helligkeitswerte mit vollem Gewicht in das erkannte Signal eingerechnet werden. Zwischen Schwelle 1 und 2 wird die Helligkeit linear gewichtet.
   Mindest–Breite gibt die Pixelbreite an, die Signale mindestens haben müssen, um als solche erkannt zu werden.
   Helligkeits–Multiply multipliziert die Farbwerte der einzelnen Pixel mit dem angegebenen Faktor (Kontrasterhöhung).
   Helligkeits–Offset zieht das 20–Fache des angegebenen Wertes von den einzelnen Pixeln ab. Multiply und Offset werden gleichzeitig ausgeführt (ohne dazwischen Helligkeit abzuschneiden), sodass jeder Helligkeitsbereich gespreizt dargestellt werden kann.
6. *Eingabebild.* Hier erscheint das ursprüngliche Bild vor der Bildverbesserung. Innerhalb des Bildes lässt sich eine rote Linie aufziehen, entlang der im rechten Teil ein Histogramm angezeigt wird. Die Werte des Histogramms beziehen sich dabei allerdings auf das Ausgabebild. Die Linie wird zur Übersichtlichkeit in diesem Bild angezeigt.
7. *Ausgabebild.* Das hier angezeigte Bild entsteht durch Bildverbesserung entsprechend der eingestellten Parameter. Werden Signale erkannt, wird zusätzlich ein Kreuz an der Position der Signalmitte eingezeichnet. Die Größe des Kreuzes entspricht der erkannten Breite des Signals. Auch hier lässt sich mit der Maus eine rote Linie aufziehen, die jedoch im Eingabebild angezeigt wird.
8. *Video–Navigation.* Wurde ein Video eingelesen, kann hier die Wiedergabe gestoppt und in der Zeitleiste navigiert werden. Angezeigt wird dabei der jeweils aktuelle Frame. Halten der SHIFT–Taste beim Klicken auf die Pfeil–Schaltflächen am Rand verändert die Frameposition um 10 statt 1.
9. *Histogramm–Auswahl.* Hier kann eingestellt werden, ob das Histogramm entlang der eingezeichneten Linie angezeigt wird oder ob das gesamte Bild ausgewertet werden soll. Auch lässt sich die Histogramm–Funktion hier ausschalten.
10. *Histogramm.* Im Linien–Modus wird auf der x–Achse die Position in Pixeln angegeben, wobei 0 den Ursprung der aufgezogenen Linie bezeichnet. Die y–Achse steht dabei für den Helligkeitswert, normiert auf eine Skala von 0 (Schwarz) bis 255 (Weiß).
    Im Verteilungsmodus entspricht die x–Achse den Helligkeitswerten und die y–Achse der Anzahl der mit diesem Wert gefundenen Pixel im Bild. Der Regler an der Seite lässt den unteren Bereich des Bildes vergrößern.
11. *Messdaten.* In der oberen Leiste dieses Bereiches wird angezeigt, wie viele Signale innerhalb dieses Frames und wie viele innerhalb einer Messauf-

zeichnung detektiert wurden. Im Einzelbild–Modus entspricht die Anzahl im aktuellen Frame immer der Gesamtzahl. „Ergebnisse“ erscheint dabei rot, wenn eine Messaufzeichnung stattfindet.
In der Liste werden die eigentlichen Signaldaten aufgeführt. Position, Breite und Helligkeit, sowie der Frame, in dem sie detektiert wurden, sind hier aufgeführt. Die Liste leert sich dabei außerhalb einer Messaufzeichnung bei jeder neuen Analyse. Im Videomodus kann mit einem Klick in die Liste zum zugehörigen Frame gesprungen werden.
Im rechten Teil des Bereiches kann die Messaufzeichnung gestartet und gestoppt werden. Dabei werden alle neuen Signale in die Liste vorne angefügt, anstatt diese zu überschreiben. Spielt man dabei ein Video ab oder wird die Kamera angezeigt, werden Signale an gleicher Position als einzelnes Signal behandelt, also der Listeneintrag wird überschrieben.
Außerdem befindet sich im rechten Teil noch die „exp“–Schaltfläche, mit der man die Liste in eine Textdatei exportieren kann. Diese enthält die vorgenommenen Programm–Einstellungen und, mit Tabulatoren und Absätzen getrennt, die Daten aus der Liste.

3 Hinweise zur Verwendung

- Um die Einstellungen flüssig vornehmen zu können, sollte dies zunächst an einem einzelnen Bild geschehen. Wird der Video– oder Kamera–Modus gestoppt, kann hier das zuletzt übertragene Bild betrachtet und in Ruhe analysiert werden.
- Der Rauschabzug funktioniert am besten, indem das Bild zu einem Zeitpunkt ohne Signale angehalten und exportiert wird. Das exportierte Eingabebild kann dann als Leerbild neu geladen werden.
- Die Messaufzeichnung startet sofort nach dem Klicken der Schaltfläche. Die Stopp-Schaltfläche in Teil 8 stoppt allerdings lediglich die Wiedergabe, nicht die Aufzeichnung. Zum Stoppen wird die Schaltfläche in Teil 11 benötigt.
- Das Stoppen der Messaufzeichnung erfolgt leicht verzögert (in diesem Zeitraum wird „Ergebnisse“ grün) um Überschreiben der Signalliste zu vermeiden. Die dazu benötigte Zeit kann für langsamere Rechner in den Optionen angepasst werden.
- die Messaufzeichnung ist besonders beim Abspielen eines Videos von Vorteil, um zu dem Frame der erkannten Signale zu springen.
- Veränderungen der Kameraeinstellungen werden erst beim Stoppen und erneuten Starten des Kameramodus umgesetzt.
- Auf langsamen oder beschäftigten Rechnern kann die Oberfläche einzelne Mausklicks ignorieren.
- Im Kameramodus wird die Framezahl bei Start auf 0 gesetzt und bei jedem übertragenen Frame um 1 erhöht.