Endbericht der Projektgruppe 464

CellTrack

Lehrstuhl für Graphische Systeme
Fachbereich Informatik
Universität Dortmund

5. Oktober 2005

Teilnehmer:

Marc Bachstein

Heiko Barg

Miriam Bode

Björn de Myn

Markus Friese

Heiko Jäkel

Jan-Hendrik Lochner

Hannes Olivier

Andreas Rack

Christof Rack

Jan Schimmelmann

Sarah Schlienkamp

Thomas Schramm

Betreuer:

Martin Wawro

Frank Weichert

Inhaltsverzeichnis

Abbildungsverzeichnis

Notationen

Dieser Endbericht verwendet die folgenden Notationen. Wird im Einzelfall davon abgewichen, so ist dieses an der entsprechenden Stelle vermerkt.

Symbol	**Bedeutung**	**Anmerkung**
$\mathbf{M}$	Matrix	Matrizen werden durch fettgedruckte Großbuchstaben bezeichnet.
$\mathbf{x}$	Vektor	Vektoren werden ebenfalls fettgedruckt aber klein geschrieben.
$x, \alpha, \ldots$	Variable	Variablenbezeichner sind kursiv geschriebene, lateinische oder griechische Kleinbuchstaben.
$c, G_{max}, T, \ldots$	Konstante	Konstantenbezeichner werden kursiv geschrieben.
$\mathbb{N}, \mathbb{R}$	Zahlenbereich	$\mathbb{N}$ bezeichnet die Menge der natürlichen Zahlen und $\mathbb{R}$ die Menge der reellen Zahlen (jeweils exklusive 0).
$\wp$	Potenzmenge	Die Potenzmenge einer Menge M (notiert als $\wp M$) bezeichnet die Menge aller Teilmengen von M ($X \in \wp M \Rightarrow X \subseteq M \cup \{\emptyset\}$).
$\overline{x}$	Mittelwert	Ein überstrichener Buchstabe symbolisiert den arithmetischen Mittelwert einer Messreihe.
$f', f'', \ldots$	Ableitung	Ableitungen eindimensionaler Funktionen werden durch Hochkommata gekennzeichnet.
$\phi_x, \phi_y, \phi_{xy}, \ldots$	Ableitung	Bei mehrdimensionalen Funktionen werden die Variablen, nach denen abgeleitet werden soll, als Index notiert.

Notation	Bedeutung	Anmerkung
`MasterController`, `QCanvasView`	Klasse	Klassenbezeichner werden in der Typewriter-Schriftart notiert. Bezeichner von eigenen Klassen beginnen in der Implementierung mit einem `C`, das in diesem Bericht jedoch aus Gründen der besseren Lesbarkeit weggelassen wird. Qt-Klassen werden jedoch zur besseren Unterscheidbarkeit mit einem führenden `Q` notiert.
findMatch	Methode	Methodennamen werden kursiv und ohne Angabe der Übergabeparameter notiert.
`Start`	Schaltfläche	Beschriftungen von Schaltflächen werden in der Typewriter-Schriftart notiert und mit einer Umrahmung versehen.
`File`	Menüpunkt	Menüpunkte werden wie Klassenbezeichner in der Typewriter-Schriftart notiert. Die Unterscheidung folgt jeweils aus dem Kontext.
`TBVideo`	SQL-Tabelle	Auch SQL-Tabellen werden in der Typewriter-Schriftart notiert. Zur Unterscheidbarkeit von Klassenbezeichnern und Menüpunkten beginnen Tabellenbezeichner stets mit dem Präfix `TB`.

Zeit- und Längenmessung Zeit und Länge wird in den allgemein üblichen SI-Einheiten gemessen und notiert. Für Bruchteile und Vielfache werden die üblichen SI-Präfixe verwendet. Beispielsweise steht μm für 10^{-6} Meter, also ein millionstel Meter.

Die O-Notation Zur Angabe oberer Schranken für die Laufzeit von Algorithmen wird hier die O-Notation verwendet. Umgangssprachlich bedeutet $O(f(n))$, dass die Laufzeit ab einer bestimmten Eingabelänge n_0 nicht schneller wächst als $f(n)$. Die exakte mathematische Definition lautet: $O(f(n)) = \{g : \mathbb{N} \to \mathbb{N} | \exists n_0 > 0 \wedge \exists c > 0, \text{ so dass } \forall n \geq n_0 : g(n) \leq c \cdot f(n)\}$.

Kapitel 1

Einleitung

Laut der Pressemitteilung des Bundesministeriums für Gesundheit und Soziale Sicherung vom 28.04.2004 [BMG] erkranken jährlich rund 358.000 Menschen in Deutschland an Krebs. Die Heilungschancen liegen zur Zeit bei nur etwa 30%. Krebs ist daher hierzulande als zweithäufigste Todesursache einzustufen. Die Krebsforschung hat in den letzten Jahren allerdings bereits beachtliche Fortschritte erzielt. So konnten als Therapieformen chirurgische und medikamentöse Behandlung sowie die Strahlentherapie etabliert werden. All diese Verfahren zeigen gute Erfolge, allerdings wird der menschliche Körper dabei enormen Belastungen ausgesetzt. Eines der wichtigsten Ziele der Krebsforscher ist daher die Entwicklung von Alternativen und die Weiterentwicklung von bereits vorhandenen Behandlungsformen.

Krebs ist eine Zellkrankheit. Gesunde Körperzellen mutieren zu Krebszellen, vermehren sich durch die Bildung von Tochtergeschwüren (den sogenannten Metastasen) und befallen dadurch im Endeffekt den gesamten Körper. Das Immunsystem des Körpers versucht, diese Metastasierung zu verhindern und bildet zu diesem Zweck Abwehrzellen aus (vgl. Kapitel 2.1.2).
An dieser Stelle setzt die Krebsforschung an: Um eine effektive Bekämpfung von Krebszellen gewährleisten zu können, benötigen die Wissenschaftler Informationen über das Verhalten von Krebs- aber auch von Abwehrzellen. Insbesondere wird untersucht, ob und in welcher Art und Weise Zellen auf bestimmte Stoffe reagieren. Diese Informationen werden in der Regel experimentell gewonnen.

Am Institut für Immunologie der Universität Witten/Herdecke (http://www.uni-wh.de) werden solche Versuche durchgeführt: Menschliche (Krebs- oder Abwehr-)Zellkulturen werden von einer Kamera durch ein Lichtmikroskop aufgenommen. Die Bilddaten werden dann analog mit Hilfe eines Zeitraffer-Videorekorders gespeichert. Um möglichst ideale Bedingungen zu schaffen, werden die Zellkulturen durch eine Wärmelampe stets auf Körpertemperatur (37°C) gehalten. Der Versuchsaufbau ist in Abbildung 1.1 zu sehen. Die Wissenschaftler beobachten die Zellen nun über einen längeren Zeitraum und untersuchen, inwiefern sich die Zugabe verschiedener Substanzen auf das Zellverhalten auswirkt. Als wichtigster Parameter wird dabei die Migrationsaktivität betrachtet. Sie bezeichnet den Prozentsatz der Zellen, der sich innerhalb des Beobachtungszeitraums bewegt. Um die Migrationsaktivität festzustellen, zeichnen die Biologen am Bildschirm die Zellbewegungen per Hand nach. Dieses Vorgehen erfolgt für jede beobachtete Zelle einzeln, wodurch sich offensichtlich ein erheblicher Zeitbedarf ergibt.

Im Wintersemester 2004/2005 wurde nun am Lehrstuhl VII des Fachbereichs Informatik an der Universität Dortmund das Projekt CellTrack (Projektgruppe 464) ins Leben gerufen. Im Rahmen dieses Projekts sollte ein Software-System erstellt werden, das den Biologen bei der Auswertung ihrer Versuchsreihen behilflich ist. Die Hauptkomponente dieses Software-Systems soll den Biologen die mühselige Arbeit der Zellenbeobachtung (Cell-Tracking) abnehmen oder zumindest

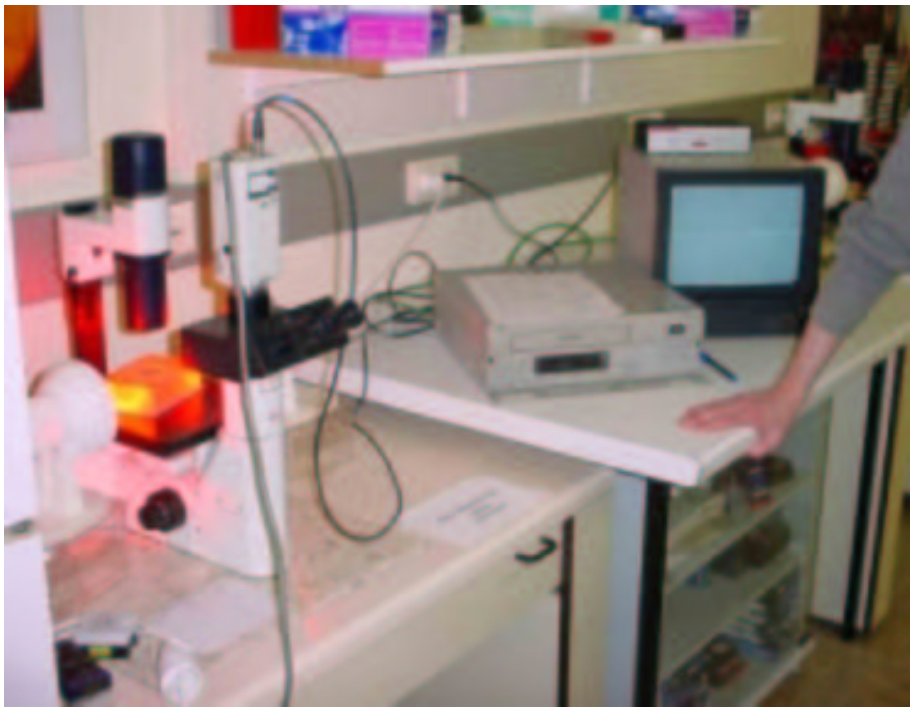


Abbildung 1.1: Versuchsaufbau für das Zell-Tracking

erleichtern. Darüber hinaus soll eine statistische Auswertung der Beobachtungsdaten möglich sein. Das Software-System sollte – grob formuliert – die folgenden Aufgaben vereinen:

- **Akquisition:** Das vorhandene analoge Bildmaterial muss in geeigneter Form digitalisiert werden.
- **Datenaufbereitung:** Die ursprünglich analoge Bilddatenerfassung erzeugt gegebenenfalls unerwünschte Störeffekte in den Aufzeichnungen. Diese sollen vor der weiteren Verarbeitung beseitigt oder zumindest reduziert werden.
- **Segmentierung:** Zellen sollen erkannt und insbesondere vom Hintergrund unterschieden (= segmentiert) werden können.
- **Tracking:** Durch Zellbewegungen hervorgerufene Änderungen sollen erfasst werden. Neben der Fortbewegung der Zellen muss hier auch die Zellverformung berücksichtigt werden. Tracking kann dabei evtl. durch iteriertes Segmentieren realisiert werden.
- **Visualisierung:** Nach der Erfassung und Berechnung von Daten, müssen diese geeignet ausgegeben werden. Es bietet sich z.B. eine grafische Ausgabe an, die den bisherigen Aufzeichnungen der Zellmigrationspfade durch die Biologen entspricht.

Darüber hinaus ist die Implementierung statistischer Verfahren notwendig, um eine Auswertung der Daten durchführen zu können. Außerdem ist beim Entwurf der Benutzungsoberfläche auf eine intuitive Bedienbarkeit zu achten.

Die Projektgruppe 464 hat sich vom Wintersemester 2004/2005 bis zum Sommersemester 2005 mit dem Entwurf und der prototypischen Implementierung eines Software-Systems auseinandergesetzt, das den oben genannten Anforderungen gerecht wird. Begonnen hat die PG-Arbeit mit

einer Seminarphase zur Schaffung eines gemeinsamen Kontextes, der die medizinisch-biologischen Grundlagen, Verfahren zur Bildverarbeitung und -verbesserung, Methoden der statistischen Datenanalyse, Prinzipien der objektorientierten Softwareentwicklung sowie Grundlagen der zu verwendenden Entwicklungsumgebung beinhaltete. Im Folgenden haben sich Teilgruppen mit verschiedenen Aspekten des Projekts befasst, wobei die Fortschritte und Teilergebnisse in zweimal wöchentlich abgehaltenen PG-Sitzungen besprochen und in das Gesamtprojekt integriert wurden.

Dieser Endbericht dokumentiert die Arbeit der Projektgruppe von der Erarbeitung der Grundlagen über den Systementwurf und die Implementierung des Prototyps bis hin zur Evaluierung des abgeschlossenen Produkts.

Kapitel 2

Grundlagen

Inhaltsangabe

2.1 Medizinisch-biologischer Hintergrund

Um besser zu verstehen, warum in der Projektgruppe CellTrack die Migrationsbewegungen von Krebszellen mit digitalen Bildverarbeitungsverfahren beobachtet werden sollen, ist es sinnvoll, den medizinischen und biologischen Hintergrund etwas zu betrachten.

In diesem Kapitel werden zunächst kurz der Aufbau und die Funktionen von Zellen beschrieben. Danach wird erläutert, was Krebs ist, wie er entstehen kann und welchen Einfluss er auf die Zellen und Organe im Körper hat. Zum Schluss wird auf Zellmigration und auf die Forschung, die momentan am Institut für Immunologie an der Universität Witten-Herdecke betrieben wird, eingegangen. Diese Forschung liefert die Motivation für die Projektgruppe CellTrack.

2.1.1 Was ist eine Zelle?

Eine Zelle ist die Struktur, aus der jedes uns bekannte Lebewesen besteht. Man kann das Leben auf der Erde grob in zwei Typen von Zellen unterscheiden: Die Prokaryoten und die Eukaryoten.

Die Prokaryoten stellen eher den „primitiven“ Zelltyp dar. Ihr Zellkern ist nicht von einer Membran umgeben, weshalb man auch von keinem „echten“ Zellkern spricht. Die DNA befindet sich frei im Cytoplasma (lebende Masse der Zelle), die Zelle ist nicht unterteilt und es sind keine Zellorganellen enthalten. Der Aufbau einer prokaryotischen Zelle ist in Abbildung 2.1 dargestellt.

Im Gegensatz dazu sind die Eukaryoten Lebewesen, bei denen die Zellen einen echten abgegrenzten Zellkern und ein Cytoskelett enthalten. In Abbildung 2.2 ist eine eukaryotische Zelle mit ihren wichtigsten Zellorganellen dargestellt. Sowohl einzellige als auch mehrzellige Tiere, zu denen wir Menschen gehören, zählen zu den Eukaryoten. Der größte Teil der genetischen Information ist im Zellkern enthalten und verschiedene Zellorganellen innerhalb der Zelle übernehmen unterschiedliche Aufgaben. Eine Besonderheit bei den Eukaryoten ist, dass sie zur Protein-Biosynthese

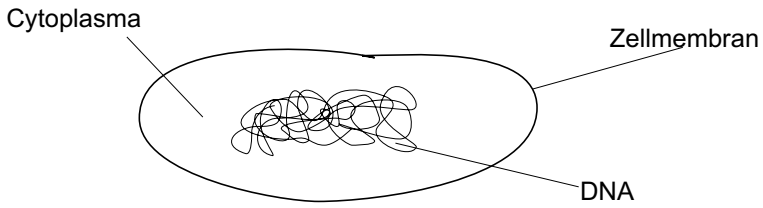


Abbildung 2.1: Schema einer prokaryotischen Zelle

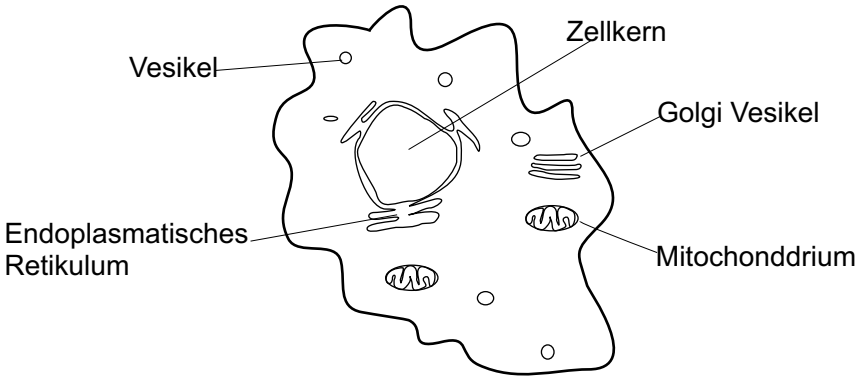


Abbildung 2.2: Schema einer eukaryotischen Zelle

fähig sind. Darunter versteht man die Bildung von Eiweißen (Proteinen) in einem Organismus. Die Vorlage für die Proteine liegt verschlüsselt in der DNA.

Eine der wichtigsten Eigenschaften von Zellen ist die Zellteilung. Nur so sind Zellen in der Lage, sich zu vermehren, um z.B. Organe oder Gewebe auszubilden. In einem mehrzelligen Organismus spezialisieren sich die Zellen, und jeder Zelltyp übernimmt bestimmte Aufgaben. Deshalb unterscheiden sich Zellen in Größe und Aussehen unter Umständen sehr stark. Die Lebensdauer einer Zelle hängt davon ab, worauf sie spezialisiert ist. Epidermiszellen sterben meist nach kurzer Zeit ab, während Zellen in Organen eventuell ein ganzes Leben im Organismus verweilen. Zu den Hauptaufgaben einer Zelle gehören die Produktion (z.B. Drüsen) und die Umwandlung von Stoffen (z.B. Leber, Darm). Abbildung 2.3 zeigt als Beispiel acht Zelltypen, die sich deutlich voneinander unterscheiden [ZD03]: (a) Eubakterien, (b) Archaebakterien, (c) Blutzellen, (d) fossilierte Dinosauriereier, (e) Grünalgen, (f) Purkinje-Zellen, (g) Epithel-Zellen und (h) Pflanzenzellen.

2.1.2 Was ist Krebs?

Krebs ist eine sehr weit verbreitete Krankheit, an der viele Menschen erkranken und die im schlimmsten Fall zum Tod führen kann. Allein in Deutschland erkranken jährlich etwa 358.000 Menschen daran. Mit modernen Behandlungsmethoden, an denen sehr viel geforscht und verbessert wird, können mittlerweile 30% der betroffenen Personen von ihrer Krankheit geheilt werden. Typische Krebserkrankungen sind z.B. Lungenkrebs, Hautkrebs, Darmkrebs, Brustkrebs bei Frauen und Prostata-Krebs bei Männern. Bei Krebs handelt es sich um Mutationen einer Zelle, so dass die Zelle nicht mehr das macht, was sie eigentlich soll. Viele solcher Mutationen werden vom Immunsystem erkannt und bekämpft, aber manche entgehen dem Immunsystem und bleiben im Körper. Mutationen häufen sich in einem Körper mit der Zeit an, und da mit zunehmendem Alter das Immunsystem immer schwächer wird, erhöht sich die Wahrscheinlichkeit, an Krebs zu erkranken, im Alter sehr stark. Eine Untersuchung von Neuerkrankungen nach Alter von 1987 bis 1996 im Saarland hat ergeben, dass bei Frauen 200 Fälle von 100.000 Personen im Alter von 45 Jahren

auftraten. Im Gegensatz dazu waren es im Alter von 75 Jahren schon 1200 Fälle. Die Statistik für Männer sieht ähnlich aus.

Tumore

Mit der Krankheit „Krebs“ wird meistens der Begriff „Tumor“ verbunden. Der Begriff „Tumor“ steht im Allgemeinen für jede umschriebene Schwellung oder Geschwulst von Körpergeweben. Wichtig ist es, dass man Tumore in gutartige (benigne) und bösartige (maligne) Tumore unterteilen kann. Um Tumore zu klassifizieren, ordnet man sie nach dem Ursprungsgewebe ein.

- Karzinome (85%): Deck- und Drüsengewebe (Epithel)
  Tumor: Haut, Lunge, Magen-Darm-Trakt, Drüsengewebe
- Sarkome (2%): Mesothel
  Tumor: Bindegewebe, Knorpel, Knochen, Muskelgewebe
- Leukämien (3,5%): Mesothel
  Tumor: Knochenmark
- Lymphome (5,5 - 9%): Mesothel
  Tumor: Milz, Knochenmark

Da das Epithel sehr teilungsaktiv und damit äußerst anfällig ist, sind die meisten beim Menschen vorkommenden Tumore also die Karzinome.

Benigne Tumore

Gutartige Tumore kennzeichnen sich dadurch, dass sie lokalisiert und umschrieben sind. Das heißt, sie sind eindeutig vom umliegenden Gewebe abgetrennt und nicht direkt damit verwachsen. Sie wachsen relativ langsam und verdrängen das umliegende Gewebe, dringen aber nicht darin ein. Bei benignen Tumoren gibt es keine Metastasierung, es bilden sich also keine „Ableger“ des ursprünglichen Tumors. Um den Tumor herum entsteht eine bindegewebige Kapsel oder Pseudokapsel, die ihn vom übrigen Gewebe separiert. Wird ein benigner Tumore operativ entfernt, so ist er damit vollständig entfernt und tritt meistens nicht noch einmal auf. In der Regel schädigt ein

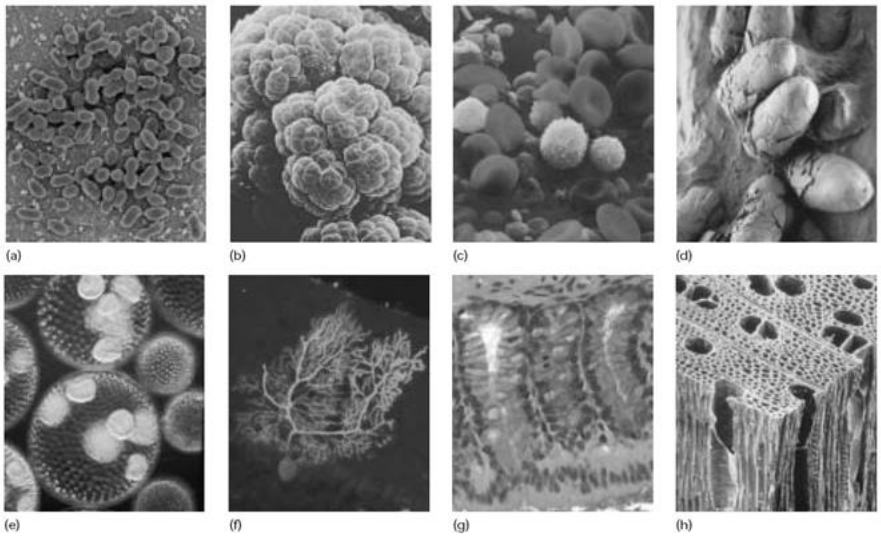


Abbildung 2.3: Unterschiedliche Zelltypen

gutartiger Tumor den Organismus nicht und kann normalerweise auch nicht zum Tode führen. Es gibt aber Ausnahmen, denn wenn der Tumor zu groß wird, kann er durch sein verdrängendes Wachstum andere lebenswichtige Strukturen und Organe „an die Wand drücken“. Beispiele für gutartige Tumore sind z.B. Warzen oder Grützbeutel.

Maligne Tumore

Wenn man von Krebs spricht, dann meint man in der Regel nur maligne Tumore und in der Literatur wird „Tumor“ auch teilweise mit „maligner Tumor“ gleichgestellt. Bösartige Tumore können sehr schnell wachsen und dringen dabei in die unmittelbare Tumorumgebung ein. Man spricht hier von Infiltration. Dies ist möglich, da maligne Tumore nicht lokalisiert und umschrieben und auch nicht durch eine Art Kapsel vom übrigen Gewebe abgetrennt sind. Sie sind in der Lage, die histologische Umgebung des Tumors regelrecht zu zerstören (Destruktion). Die Tumorzellen brechen in Lymph- und Blutgefäße ein und werden dadurch in andere Organe verschleppt. Wenn diese verschleppten Zellen dann wieder zu einem neuen Tumor heranwachsen, entsteht eine Tochtergeschwulst, eine so genannte Metastase. Wird ein solcher Tumor nicht rechtzeitig erkannt, ist die Behandlung sehr schwierig, denn Metastasen sind oft schwer zu lokalisieren oder zu behandeln. Wenn sich schon viele Metastasen gebildet haben, ist die Behandlung sehr schwierig und oft kann dem Erkrankten nicht mehr geholfen werden.

2.1.3 Wie entsteht Krebs?

Wie schon erwähnt, handelt es sich bei Krebs um Mutationen einer Zelle. Einige Mutationen können vom Immunsystem erkannt werden und die betroffenen Zellen werden dann bekämpft. Manche Mutationen entgehen dem Immunsystem aber, weil bösartige Zellen teilweise in der Lage sind, die Erkennungsmechanismen des Immunsystems zu täuschen. In diesem Fall spricht man von einem „tumor escape“. Besonders Zellen, die sich ständig teilen, sind sehr anfällig für Krebs, denn dort können sich die Mutationen schnell verbreiten.

Den Prozess der Entstehung von malignen Tumoren nennt man Karzinogenese oder auch Kanzerogenese. Entscheidend für die Karzinogenese sind die Onkogene. So bezeichnet man Gene in Tumorzellen oder in Tumorviren, die die Entstehung von Krebs auslösen oder begünstigen.

Karzinogenese

Die Karzinogenese ist ein mehrstufiger Prozess, der durch so genannte Karzinogene beeinflusst wird. Diese Karzinogene können entweder die Entstehung eines Tumors auslösen oder auch nur begünstigen. Bei der Karzinogenese wird das „normale Verhalten“ einer Zelle „umprogrammiert“. Die Zelle erfüllt also nicht mehr die Funktionen, für die sie ursprünglich spezialisiert ist, sondern beginnt, unkontrolliert zu wachsen und sich zu vermehren. Man unterscheidet mutagene und epigenetische Karzinogene. Die mutagenen Karzinogene bewirken Veränderungen in der DNA, während die epigenetischen Karzinogene die DNA nicht nachweislich beeinflussen.

Normalerweise sind die Basenzusammensetzungen der DNA im lebenden Organismus (in vivo) außerordentlich konstant. Dies wird durch DNA-Reparatursysteme ermöglicht, die versuchen, Fehler in der Basenzusammensetzung sofort zu korrigieren, falls dies möglich ist. Bei der Karzinogenese werden aber diese Systeme immer mehr außer Kraft gesetzt, so dass nicht nur die befallene Zelle, sondern auch alle ihre Nachkommen fehlerhafte DNA-Bausteine enthalten. Insgesamt werden sechs wesentliche Veränderungen an der Krebszelle vorgenommen [ZD03]:

1. *Self-sufficiency in growth signals:*
   Die Zelle verliert ihre Fähigkeit, kontrolliert zu wachsen. Sie kann sich jetzt selbst mit Wachstumssignalen versorgen. Dadurch wird das Wachstum extrem beschleunigt.

2. *Insensitivity to antigrowth signals:*
   Die Wirkung von wachstumshemmenden Signalen lässt nach. Normalerweise verhindern be-

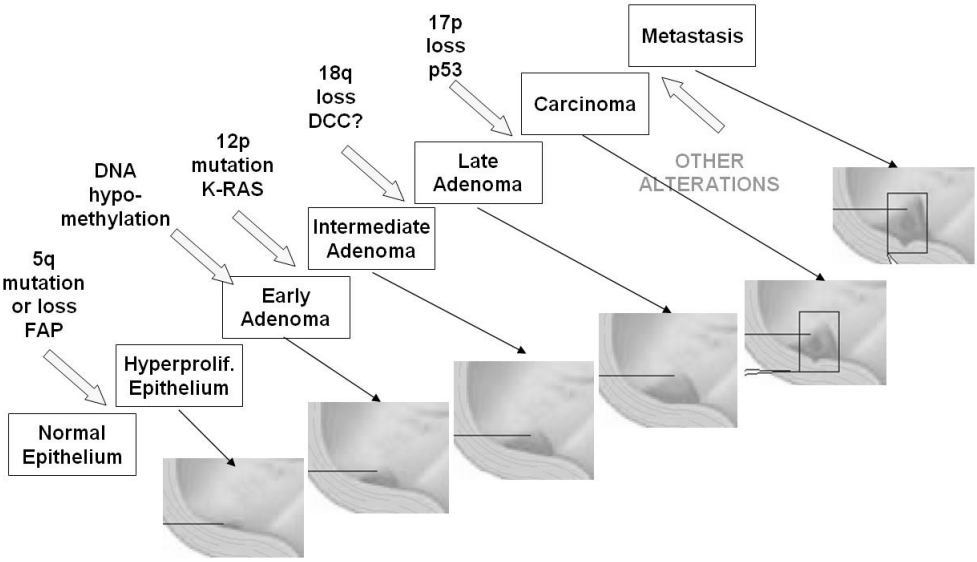


Abbildung 2.4: Schema der Karzinogenese

stimmte chemische Signale, dass die Zelle ungehindert wächst. Aber nachdem die Zelle von Krebs befallen wurde, wird sie immun gegen diese Signale.

3. *Evasion of apoptosis:*
   Der programmierte Zelltod wird umgangen. Viele Zellen sterben ab, nachdem sie eine bestimmte Zeit im Körper waren, dies nennt man Apoptose. Eine Zelle, die schon tot sein sollte, lebt also weiter im Körper.

4. *Limitless replicative potential:*
   Normalerweise kann sich eine Zelle nicht beliebig oft teilen, denn sie wird durch ihre Umgebung im Gewebe daran gehindert, um anderem Gewebe nicht zu schaden. Dieser Schutzmechanismus wird umgangen und die Zelle kann sich unbegrenzt oft replizieren.

5. *Tissue invasion and metastasis:*
   Die Krebszellen dringen in fremdes Gewebe ein und bilden Metastasen. Im Gegensatz zu benignen Tumoren kann ein maligner Tumor in fremdes Gewebe eindringen und es sogar zerstören. Es kommt dann entfernt vom Primärtumor zur Bildung von Metastasen.

6. *Sustained angiogenesis:*
   Die Blutgefäßbildung wird verstärkt. Da der Tumor gut mit Blut versorgt werden muss, werden verstärkt Blutgefäße um ihn herum gebildet.

Wie schon erwähnt, ist die Karzinogenese ein mehrstufiger Prozess. Von Stufe zu Stufe verändert sich die DNA der Zelle immer mehr und bestimmte Gene mutieren oder gehen verloren. Abbildung 2.4 zeigt die einzelnen Stufen der Karzinogenese mit den jeweiligen Genmutationen. Auf der ersten Stufe sprechen wir noch von einem normalen Epithel, dann von einem „Hyperproliferate Epithel“, also von einem Epithel, das sehr schnell wuchert. Dann entwickelt sich ein frühes Adenom, also eine gutartige Geschwulst, welche dann auf der nächsten Stufe fortschreitet und dann ein spätes Adenom bildet. Auf der vorletzten Stufe entsteht dann ein Karzinom, also ein bösartiger Tumor, aus dem dann später Metastasen gebildet werden können.

p53 - Ein Tumorsuppressorgen

Im Jahr 1979 fand man das Eiweiß p53 und kurze Zeit später entdeckte man auch das dazugehörige Gen. Erst dachte man, es würde zu den Onkogenen gehören, aber dann endeckte man, dass p53 die normale Zelle vor der Entartung bewahrt. Deshalb zählt man es zu den Tumorsuppressorgenen, da es die unkontrollierte Zellteilung verhindern soll. Seitdem man diese Eigenschaft entdeckt hat, ist p53 in das Zentrum der Forschung gerückt und wird als „Wächter des Genoms“ bezeichnet. Dieser „Wächter des Genoms“ p53 verhindert in der normalen Zelle, dass ein DNA-Schaden bei der Zellteilung an die Nachkommen weitergegeben wird. Nur wenige p53-Moleküle befinden sich normalerweise im Zellkern. Erst wenn die DNA geschädigt wird, wird p53 aktiv und alarmiert weitere Moleküle. Einige von ihnen binden sich an die DNA. Als Erstes wird die Zellteilung gestoppt. P53 leitet dann zelluläre Reparaturmechanismen ein, die den Defekt beheben sollen. Dann erst darf sich die Zelle weiter teilen. Bei einem zu großen Schaden schützt p53 die Körperzelle auf eine andere Art vor der Entartung. Wenn die DNA so stark geschädigt ist, dass eine Reparatur aussichtslos scheint, ordnet p53 die Apoptose an. So kann das mutierte Erbgut nicht mehr weitergegeben werden. Heute weiß man, dass p53 in nahezu 60% aller menschlichen Tumore ausgefallen ist. Meistens passiert das, wenn das p53 Gen selbst durch eine fehlerhafte Veränderung mutiert ist. Solche Veränderungen im p53 Gen stellen damit die häufigste gemeinsame genetische Veränderung in menschlichen Tumoren dar. Wenn p53 verändert ist, kann es sich nicht mehr wie sonst an die DNA binden und die Zellteilung stoppen, um dann Reparaturen oder die Apoptose einzuleiten. Die Zelle teilt sich trotz des Schadens an der DNA weiter, so dass ein Tumor entstehen kann.

Ursachen für Krebs

Vielfach hört man von krebserregenden Stoffen oder Strahlen. Oft wird Krebs durch solche Umwelteinflüsse ausgelöst oder dessen Wachstum von ihnen begünstigt. Man spricht in diesem Zusammenhang auch von Karzinogenen. Bekannte Ursachen für Krebs sind [Wag96]:

- **Strahlung**
  - *α-Teilchen und γ-Strahlen:*
    Die Bewohner von Hiroshima und Nagasaki wurden bei den Atombombenexplosionen diesen Strahlen ausgesetzt.
    Tumore: Leukämien, multiples Myelom, Kolon-Karzinom, Blasen-Karzinom
  - *Röntgen-Strahlung:*
    Tumore: Leukämien und andere Tumore
  - *UV-Strahlung (280-320 nm):*
    Hellhäutige Personen sollten sich mit Sonnencreme gegen UV-Strahlen schützen, wenn sie sich der Sonne aussetzen.
    Tumore: Basaliom, Plattenepithel-Karzinom, Melanom
- **Chemische Stoffe**
  - *Tabak:* In Tabak sind polyzyklische aromatische Kohlenwasserstoffe und Nitrosamine enthalten. Diese können zu Tumoren in Mund, Pharynx (Rachen), Larynx (Kehlkopf), Ösophagus (Speiseröhre), Lunge und Blase führen.
  - *Schimmelpilz:* Dort enthaltene Aflatoxine können zu Leberkrebs führen.
  - *Ruß:* Besonders bei Schornsteinfegern kam es in früheren Jahrhunderten zu einem Karzinom im Hodensack, ausgelöst durch polyzyklische aromatische Kohlenwasserstoffe.
  - *Anilinfarben:* Bei Arbeitern in Anilinfarbenwerken kommt es zu einem Blasenkarzinom, welches durch aromatische Amine entstehen kann.

In den Medien wurde vor kurzer Zeit oft über Acrylamid berichtet, welches entsteht, wenn Kohlenhydrate und Asparagin (eine Aminosäure) zu stark erhitzt werden. Auch dieser Stoff ist krebserregend.

- **Viren**
  Besonders in ärmeren Ländern, in denen die Hygiene-Standards nicht so hoch sind, kommt es häufig zu Virusinfektionen, die das Risiko von Tumorerkrankungen beim Menschen erhöhen. Viren führen erst zu einer benignen Proliferation (links vom Pfeil) und dann zu einem malignen Tumor (rechts vom Pfeil).
  - *Hepatitis-B-Virus (HBV):* fokale Leberhyperplasie → hepatozelluläres Karzinom
  - *Epstein-Barr-Virus (EBV):* „Hairy"-Leukoplakie, infektiöse Mononukleose → Burkitt-Lymphom, Nasopharynx-Karzinom
  - *HTLV-1 (Onkovirus, Typ C):* „Smouldering Leukemia" → hepatozelluläres Karzinom
  - *HIV-1:* benigne Lymphoproliferation → Burkitt-Lymphom, Kaposi-Sarkom

- **Erbliche Dispositionen**
  Manche Gene erhöhen die Gefahr, an einem Tumor zu erkranken. Diese Gene werden auch Tumor-Dispositionsgene genannt. Hinweise auf erbliche Dispositionen sind:
  - *Familie:*
    * Zwei oder mehr Verwandte ersten Grades mit demselben Tumor
    * Zwei oder mehr Verwandte mit seltenen Tumoren
    * Evtl. drei oder mehr Tumore in typischer Assoziation (z.B. Brustdrüse und Eierstock)
  - *Patient:*
    * Multiple Tumore in einem Organ oder in typischer Assoziation
    * Assoziation von Tumore mit anderen genetischen Auffälligkeiten oder kongenitalen Defekten
    * Tumore an ungewöhnlichem Ort oder in frühen Lebensjahren

2.1.4 Behandlungsmethoden

Trotz intensiver Forschung sterben immer noch viele Krebspatienten an ihrer Krankheit. Zu den bekanntesten Behandlungsmethoden gehören die operative Entfernung des Tumors, Strahlentherapie und Chemotherapie. In den Abschnitten 2.1.4 bis 2.1.4 werden diese häufig angewendeten Behandlungsmethoden erklärt und deren Vor- und Nachteile erläutert.

Operative Entfernung des Primärtumors

Wird der Tumor rechtzeitig erkannt, so dass er ohne größere Probleme operativ entfernt werden kann, wird dem Patienten mit dieser Methode meistens sehr gut geholfen. Heutzutage stirbt fast niemand mehr an einem Primärtumor. Bei über 90% der kranken Personen kann der Tumor erfolgreich entfernt werden. Das Problem bei dieser Methode ist, dass man den Krebs nur dann besiegt hat, wenn sich noch keine Metastasen gebildet haben. Auch wenn sich zum Zeitpunkt der Operation noch keine Metastasen im Körper befunden haben, können sich trotzdem schon so genannte „Schläferzellen" (dormant cells) im Körper verbreitet haben. Diese Zellen verweilen oft jahrelang unerkannt im Körper und können dann plötzlich anfangen zu wachsen, um einen neuen Tumor zu bilden.

Chemotherapie

Falls der Körper des Krebskranken schon mit Metastasen befallen ist, kann ihm mit einer Operation nur teilweise geholfen werden. Wenn sich die Metastasen im Gewebe verteilt haben, kann man sie oft schwer lokalisieren und behandeln. Deshalb greift man bei der Chemotherapie zu chemischen Stoffen, um die Tumorzellen abzutöten. Diese Behandlung ist für den Patienten leider sehr unangenehm und schmerzhaft, da die Therapie nicht zwischen gesunden und kranken Zellen unterscheiden kann. Weil die Chemotherapie alle Zellen angreift, die sich in ständiger Teilung befinden, werden auch gesunde Zellen, z.B. Drüsen, angegriffen.

Strahlentherapie

Eine weitere Möglichkeit ist die Behandlung mit ionisierender Strahlung, um die Krebszellen zu zerstören. Die Therapie basiert darauf, dass strahlungsempfindliche Tumorzellen stärker reagieren als umliegende gesunde Zellen. Kleine und gut zugängliche Tumore können so effektiv behandelt werden. Meistens werden Photonenstrahlen benutzt, aber neuerdings gibt es auch eine Behandlungsmethode mit Protonen [Ess04], bei der die Ärtze den Tumor gezielter und deshalb mit höheren Dosen bestrahlen können. Die Strahlentherapie wird häufig zur Unterstützung anderer Therapien angewendet.

2.1.5 Entwicklung von Tumor-Metastasen

Im Folgenden wird kurz darauf eingegangen, was Zellmigration eigentlich ist und wodurch sie im menschlichen Körper zustande kommt. Diese Zellmigration ist ein sehr wichtiger Faktor in der modernen Krebsforschung, denn durch Migration ist es den Tumorzellen möglich, Metastasen zu bilden.

Zellmigration

Die Zellmigration ist ein ganz natürlicher Prozess, der nicht nur bei Tumorzellen auftritt. So gehört die Migration von Leukozyten, also weißen Blutkörperchen, zur alltäglichen Funktionsweise unseres Immunsystems. Sobald eine Infektion oder Verletzung im Körper registriert wird, werden Lockstoffe ausgesendet, um die Helferzellen an die betroffene Stelle zu dirigieren. Um die Leukozyten an die richtige Stelle zu „navigieren“ nutzt das Immunsystem Lockstoffe. Diese Lockstoffe sind entweder Chemokine (chemoattraktive Stoffe) oder Neurotransmitter. Sie können entweder inhibitorisch oder exitatorisch wirken, also die Migration hemmen (z.B. β-Blocker) oder anregen (z.B. Noradrenalin). Es stellt sich bei der Zellmigration natürlich die Frage, wie sich eine Tumorzelle überhaupt im Gewebe bewegen kann. Die Zelle haftet mit ihrer Zellmembran am Gewebe. Wenn sie sich bewegen will, bildet sie einen kleinen Fortsatz aus, der sich dann etwas entfernt wieder an das Gewebe haftet. Der Zellkörper folgt dann diesem Fortsatz, wobei die Zelle aber nicht die Verbindung mit dem Gewebe verliert. Die Migration ist in Abbildung 2.5 dargestellt. Tumorzellen bewegen sich allerdings wesentlich langsamer als Zellen, die sich in den Blutbahnen befinden.

Chemokine und Neurotransmitter

Chemokine werden besonders bei Entzündungen im menschlichen Körper freigesetzt. Sie sind maßgeblich beteiligt an der Vermehrung, Entwicklung, dem Wachstum und der Migration von Tumor-Metastasen. Die Tumorzellen nutzen Chemokinegradienten, um sich im Körper zu verbreiten. Sie bewegen sich dahin, wo die Konzentration an Chemokinen am größten ist.

Im Gegensatz zu den Chemokinen sind die Neurotransmitter mit psychosozialen Faktoren verbunden. Sie werden nicht vom Immunsystem abgegeben, sondern vom neuroendokrinen System.

Die Migration der Krebszellen wird im Primärtumor durch Neurotransmitter und Chemokine angeregt. Von dort aus breiten sich die Tumorzellen in den Blut- und Lymphgefäßen aus und wandern dann zur Quelle der Chemokine oder Neurotransmitter [JZ04].

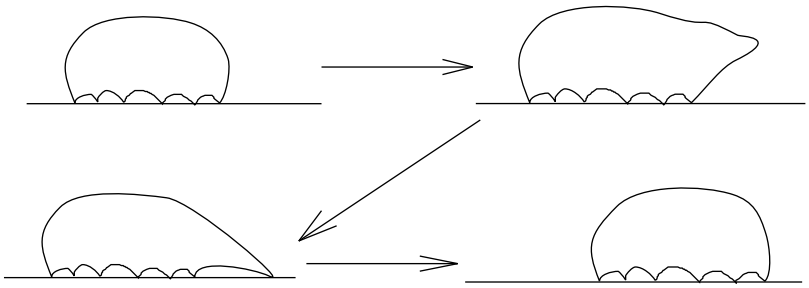


Abbildung 2.5: Schema der Zellmigration bei Tumorzellen

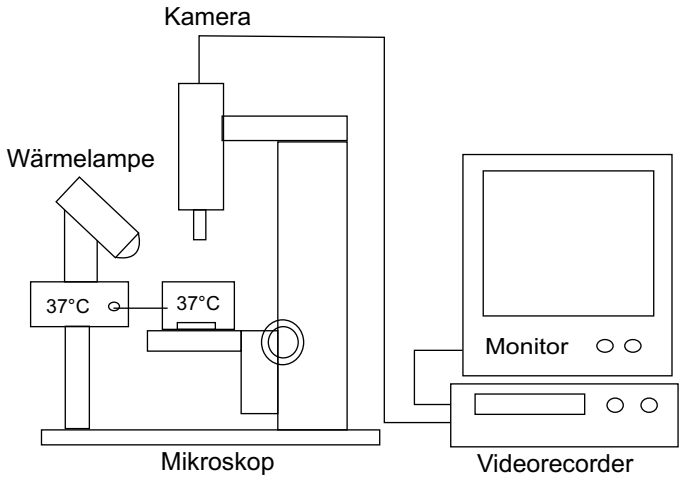


Abbildung 2.6: Schema des Versuchsaufbaus

2.2 Stand der Forschung

Die Immunologen in Witten-Herdecke möchten herausfinden, welche Chemokine und Reize die Krebszellen beeinflussen können. Sie wollen die Migrationsformen verstehen, mit denen sich die Tumorzellen im Körper ausbreiten. Wenn sie diese Prozesse genauer kennen, wird es eventuell möglich sein, die Bildung von Metastasen zu verhindern. Es könnten neue Medikamente entwickelt werden, mit denen die Migration von Krebszellen verhindert werden kann, ohne dabei das Immunsystem zu schwächen. Bei dieser Art von Sekundärprävention sollen dann wirklich nur die Tumorzellen bekämpft werden und nicht mehr die gesunden Zellen des Körpers.

2.2.1 Der Versuchsaufbau

Schon seit einiger Zeit laufen am Institut für Immunologie der Universität Witten-Herdecke Migrationsversuche mit Tumorzellen. Eine Versuchsapparatur (vgl. Abbildung 2.6) besteht aus einem Mikroskop, einem speziellen Objektträger mit den Tumorzellen in einer Collagenmatrix, einer Kamera, einer Wärmelampe und einem Videorekorder, an dem direkt ein Zeitraffer eingestellt werden kann. Insgesamt vier solcher Apparaturen stehen dem Institut zur Verfügung. So können mehrere Versuche gleichzeitig ablaufen. Es kann z.B. eine Tumorzellenart ohne Lockstoff, mit erregendem Lockstoff, mit hemmendem Lockstoff und mit beiden Lockstoffen beobachtet werden. In Kapitel 2.3 wird noch einmal genauer auf den Versuchsaufbau eingegangen.

Die Objektträger für die Migrationsversuche werden am Institut von den Mitarbeitern selber

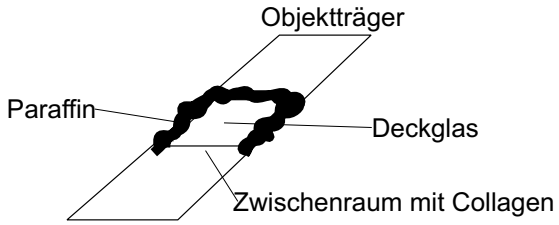


Abbildung 2.7: Schema eines Objektträgers für Migrationsversuche

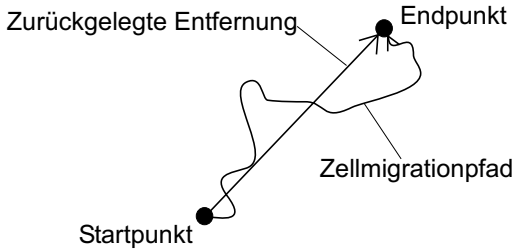


Abbildung 2.8: Migrationspfad und zurückgelegte Entfernung

gefertigt. Auf das normale Trägerglas wird ein kleines Deckglas mit Paraffin aufgebracht, so dass ein kleiner Zwischenraum entsteht. Dieser wird mit Collagen gefüllt, das als Trägermatrix dienen soll. Abbildung 2.7 zeigt einen solchen Objektträger.

Die Trägermatrix

Collagen ist ein bei Menschen und Tieren vorkommendes Strukturprotein des Bindegewebes. Normalerweise finden Versuche immer in einer zweidimensionalen Umgebung statt. Das heißt, die Zellen können sich nur zu den Seiten, nicht aber nach unten oder oben bewegegen. Um den Tumorzellen aber eine möglichst körperähnliche Umgebung zu schaffen, wird mit dem Collagen eine dreidimensionale Umgebung geschaffen. Bei einer Konzentration von 1,67 mg/dl können in-vivo-Bedingungen nachgeahmt werden.

Das Collagen bildet ein Fasernetzwerk aus, an dem sich die Tumorzellen bewegen können. Die Zellen bewegen sich aber nicht nur daran, sie verändern die Struktur oder zerstören das Collagen sogar. Da in den Versuchen ungefähr 30 Zellen gleichzeitig beobachtet werden sollen, kann man sie nur zweidimensional beobachten. Bei einer dreidimensionalen Beobachtung könnte man nur eine Zelle beobachten, da man diese über den gesamten Zeitraum im Fokus behalten müsste. Allerdings geht bei der zweidimensionalen Beobachtung nur wenig verloren, da die Collagenfasern die Eigenschaft haben, sich in x-y-Richtung zu orientieren. Nur wenige Fasern gehen in die Tiefe (z-Komponente), deshalb wandern die Tumorzellen auch meistens in x-y-Richtung.

Was soll beobachtet werden?

Bei den Versuchen soll beobachtet werden, wodurch Migration stimuliert wird und wie man sie hemmen kann. Es bestehen Unterschiede zwischen Leukozyten und Tumorzellen. Manchmal reagieren die beiden Zelltypen auf einen Lockstoff gleich, teilweise ist die Reaktion aber auch sehr unterschiedlich. Wichtig für die Versuche ist besonders, welche Entfernung die Zellen in welcher Zeit zurücklegen (vgl. Abbildung 2.8) und wie lange sie Pause machen. Je weniger Pausen eine Zelle macht, desto schneller bewegt sie sich.

Bisher konnte schon erfolgreich gezeigt werden, dass die Zellen entlang eines chemischen Gradienten gerichtet wandern können [ZE03]. Mit einem speziellen Versuchsaufbau (vgl. Abbildung 2.9)

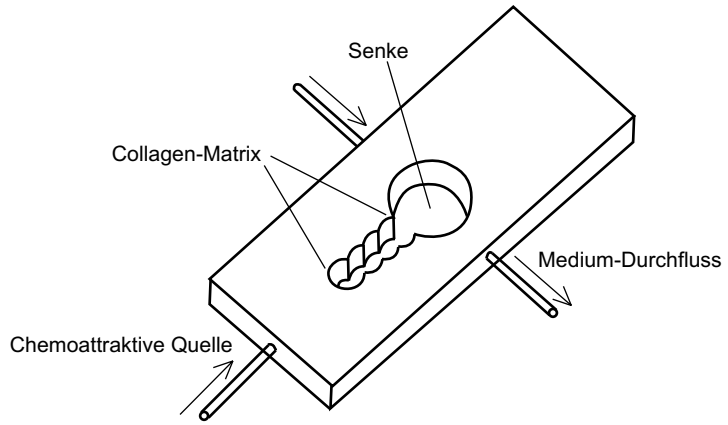


Abbildung 2.9: Versuchsaufbau für Chemotaxis-Versuch

konnte diese Chemotaxis eindeutig belegt werden. Damit bei dem Versuch immer ein Chemokine-Gradient vorhanden ist, wird ständig neues Medium hinzugegeben, damit sich kein Gleichgewicht einstellt. Dabei wanderten 57% der Zellen zur Quelle der Chemokine, 5% in die entgegengesetzte Richtung, 13% wichen nach rechts ab und 25% nach links [ZE03].

Probleme bei Migrationsversuchen

Verschiedene Probleme können bei Migrationsversuchen auftreten, so dass man generell mit einem Fehler von 10% rechnen muss.

Falls die Zellen etwas in der Matrix abtauchen, verändern sie ihre Farbe bzw. bei Schwarzweißaufnahmen ihren Grauwert oder verschwinden ganz. Wenn eine Zelle dann wieder auftaucht, weiß man nicht genau, ob es die ist, die kurze Zeit vorher abgetaucht ist. Treffen sich zwei oder mehr Zellen an einem Punkt, so ist es sehr schwierig, den Überblick zu behalten. Die Zellteilung kann auch ein Problem darstellen, allerdings teilen sich Krebszellen nur alle 24-26 Stunden, so dass dies nicht so sehr ins Gewicht fällt.

2.2.2 Manuelles Tracking

Derzeit werden die gewonnenen Daten am Institut für Immunologie der Universität Witten Herdecke von Hand ausgewertet.

Es wird ein Bildbereich von 300x400 μm mit Hilfe einer normalen Schwarz-Weiß-Videokamera, die auf einem Lichtmikroskop montiert ist, auf VHS-Bänder aufgenommen. Mit dem verwendeten, zeitrafferfähigen Videorecorder wird alle 80 Sekunden ein Bild des Versuchs auf VHS-Kassette aufgezeichnet. Verwendet werden handelsübliche 180 Minuten Kassetten, da sie dickeres Bandmaterial enthalten als Kassetten mit längerer Spieldauer. Aufgrund des Zeitraffers, im Zusammenhang mit dem Schrägspur-Verfahrens bei VHS, wird das Bandmaterial durch den rotierenden Videokopf stark belastet.

Für die Auswertung wird das Video noch einmal zeitlich gerafft. Hierzu wird es mit Zeitraffer umkopiert, so dass man ein Video erhält, welches nur noch ein Bild pro 15 Minuten Versuchsdauer umfasst.

Dieses wird per Overlay mit normalen 25 fps an einem älteren Macintosh Computer (G3 oder älter) in einer Software angezeigt. Die Software dient dazu, die Mausbewegungen der Benutzer aufzuzeichnen. Der Benutzer muss dann manuell die Maus über der zur Auswertung herausgesuchten Zelle halten. Dieser Vorgang wird für ca. 30 Zellen pro Versuch wiederholt.

Dabei kann es vorkommen, dass einzelne Zellen sich als ungeeignet herausstellen. Die Zellen können abtauchen (aus dem Fokus verschwinden), den Bildbereich verlassen, Klumpen bilden (wel-

ches eine Differenzierung in einzelne Zellen verhindert) oder absterben. In diesem Fall wird die Zelle verworfen und mit der nächsten vom Anfang des Bandes weitergemacht.

Durch dieses manuelle Tracken der Zellen entsteht eine Fehlerrate von geschätzten 10%, da durch ungewollte oder unbewußte Mausbewegungen Fehler in die Auswertung kommen. Die Zellen bilden beispielsweise Scheinfüßchen in eine etwaige Bewegungsrichtung aus, welche den auswertenden Menschen dazu veranlassen können die Maus in diese Richtung zu bewegen, ohne dass sich die Zelle als ganzes dort hinbewegt hat.

2.2.3 Störeinflüsse

Auf den Videos werden unterschiedliche Störeinflüsse unweigerlich mit aufgezeichnet. Als erstes wäre die Wärmelampe zu nennen. Diese sorgt dafür, dass die Zellen bei einer konstanten Temperatur von 37 ° C gehalten werden. Das Ein- und Ausschalten der Lampe ist mit einer entsprechenden Veränderung der Helligkeit im Bild zu sehen. Für einen Menschen spielt dies keine Rolle bei der Auswertung, jedoch könnte es das maschinelle Auswerten stören. Weiterhin sind auch langsame Helligkeitsveränderungen zu beobachten, deren Ursache nicht ganz klar ist. Dabei wird das Bild langsam dunkler. Vermutlich greift hier ein Anpassungsmechanismus der Kamera ein, welcher zum Teil dieses ungewünschte Ergebnis auf den Aufnahmen hinterlässt.

Problematischer für die automatische Auswertung könnten sich Verwacklungen im Videobild auswirken. Da die Versuche in einem alten Gebäude mit Holzdecken stattfinden, werden Erschütterungen durch Personen auf dem Stockwerk und auch durch LKWs auf der vorbeiführenden Strasse an den Versuchsaufbau weitergegeben und durch die Vergrößerung entsprechend stark im Videobild sichtbar. Dies führt zu kurzen Positionsänderungen der Zellen im Bild, die nicht mit in die Auswertung fließen sollten.

2.2.4 Archivierung

Die VHS-Kassetten mit den Versuchsaufzeichnungen werden, um z.B. Versuchsergebnisse belegen zu können, archiviert. Die Auswertungen werden mit Verweis auf die jeweilige Kassette abgelegt. VHS-Bänder unterliegen wie jedes „Speichermedium“ einer Alterung, so dass die Qualität nach einigen Jahren stark leidet und nach 10 bis 15 Jahren die Aufnahmen auf den Bändern verloren sind. Digitale Medien unterliegen natürlich auch einer Alterung, die unter Umständen die Daten sogar früher unbrauchbar macht, jedoch erlaubt die Digitaltechnik ein verlustfreies Kopieren auf „frische“ Datenträger bevor die alten unlesbar geworden sind. In Archiven wird durch regelmäßiges Kopieren der Datenbestände gewährleistet, dass sie erhalten bleiben. Zudem werden die zu den Medien passenden Lesegeräte mit eingelagert.

2.2.5 Kameraeigenschaften

Bei den Versuchen wird eine Videokamera verwendet, welche 280 „Linien“ auflöst (und ungefähr 300 Pixel Horizontal). In der Kamera findet ein 1/2-Zoll Schwarz-Weiß-CCD Verwendung. Neuere Modelle haben nur noch einen 1/3-Zoll Sensor, was mit der verwendeten Optik (anderer Öffnungswinkel) einen anderen Bildausschnitt ergibt, welches das bisher verwendete Auswertungssystem nicht berücksichtigen kann. Des Weiteren sind mittlerweile auch keine S/W-Kameras mehr zu bekommen.

2.2.6 Andere Projekte zur Zellverfolgung

Neben dem Projekt CELLTRACK gibt es noch andere Projekte die sich mit dem Problem beschäftigen Zellen in Videos zu verfolgen. Mehrere Projekte benutzen Snakes als Algorithmus um die Zellen oder Objekte in dem Video zu erkennen, hier wird meist von vollständig sichtbaren Objekten ausgegangen um die die Snake gelegt werden kann. Prinzipiel verlaufen alle Verfahren ähnlich zu dem

in Grundlagen-Kapitel erläuterten Snake. Ein Ansatz in Verfolgung von Zellen ist Level-Set, welches auch Einzug in das CellTrack-Projekt gefunden hat und ebenfalls im Grundlagenkapitel erläutert wird. Ein weiterer Ansatz ist das Multi-Agent System zur Zellverfolgung, welches auf der Basis von einem Bereichwachstumsverfahren arbeitet: Hier geht man von der Ausgangsituation aus, die Zellen in der betrachteten Bildfolge besitzen eine helle Aura. Eine häufig verwendete Vorgehensweise ist, „segmentieren, interagieren und reproduzieren“. Der erste Schritt ist die Segmentierung, sie erfolgt über das Bereichwachstumsverfahren, d.h. man beginnt mit bestimmten ausgewählten Pixeln und bestimmt, ob die Randpixel genügend viele Übereinstimmungen haben (mögliche Merkmale sind hier z.B. Grauwerte oder gefundene Kanten mittels Sobel-Operator o.ä.) um zum gleichen Objekt zu gehören (in diesem Fall das Zellinnere) in mehreren Iterationen wird somit das gesamte Objekt gefunden. Hier bekommt jeder Startpixel einen Agenten, der das Verhalten über das Bereichwachstumsverfahren an dieser Pixelmenge beeinflusst. Der zweite Schritt ist die Interaktion, hier können die einzelen Agenten miteinander interagieren, z.B. können sie zwei Pixelmengen vereinen. Der letzte Schritt ist die Reproduktion von Agenten an einer anderen Stelle im gleichen Bild oder im nächsten Bild.

2.3 Grabbing

In diesem Kapitel werden grundlegende Methoden zur analogen und digitalen Aufzeichnung von analogen Videosignalen sowie Kompressionstechniken für digitale Videos behandelt. Im Besonderen geht es dabei um folgende Aspekte:

- Manuelles Tracking der Zellen bisher
- Aufnahme der Zellen auf VHS
- Framegrabbing
- Digitale Videoformate
- Verlustfreie/verlustbehaftete Kompression

Das Tracking der Zellen am Institut für Immunologie an der Universität Witten-Herdecke wird bisher manuell von den Biologen mit Hilfe herkömmlicher Analog-Video-Technik erledigt. Im Folgenden wird erläutert, was diese Analog-Technik ausmacht und wie das analoge Fernsehsignal aufgebaut ist. Hiernach wird die Digitalisierung und, zur besseren Handhabung des Signals, die Kompression von Bilddaten erklärt. Bei der Kompression werden zwei verlustlose Techniken, sowie ein verlustbehafteter Algorithmus, wie er im JPEG oder MPEG Format eingesetzt wird, vorgestellt.

2.3.1 PAL/VHS Eigenschaften

Das VHS-System wurde von JVC 1976 für Heimanwendungen entwickelt. Es zeichnet das Bild in einem Schrägspurverfahren mit Hilfe eines rotierenden Videokopfes auf. Dazu kommt eine Mono-Audio-Längsspur, später wurde eine Stereo-Schrägspur Aufzeichnung mit besserer Qualität möglich. Die Auflösung von VHS beträgt 250 „Linien“ (entspricht letztlich ungefähr 300×250 „Pixel“). Beim PAL VHS werden 50 Halbbilder (vgl. Kapitel 2.3.1) pro Sekunde aufgezeichnet. Das Bildsignal liegt als Compositesignal (FBAS, vgl. Kapitel 2.3.1) vor, in dem Farb- und Helligkeitsinformationen in ein Signal moduliert sind.

Interlaced

Der PAL Fernsehstandard sieht 50 Halbbilder pro Sekunde vor, was dann 25 Vollbildern in der Sekunde entspricht (bei NTSC sind es zum Vergleich 30 Bilder/Sekunde und im Kino werden üblicherweise 24 Bilder/Sekunde verwendet). Ein Halbbild besteht dabei aus den geraden oder ungeraden Zeilen eines Bildes. So dass als erstes ein Bild mit den Zeilen $\{1, 3, 5, 7, \dots\}$ gezeigt

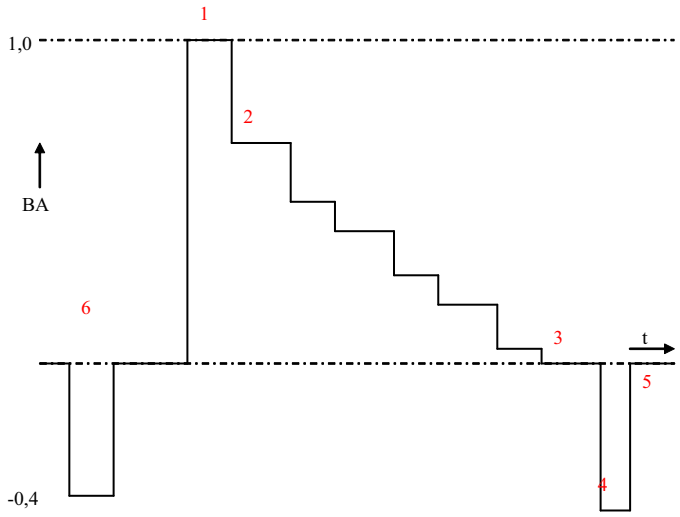


Abbildung 2.10: Das Schwarzweiß-Fernsehsignal: BAS

wird, daraufhin kommen dann die geraden Zeilen {2, 4, 6, 8, ...}. Durch Zusammenfügen der Halbbilder erhält man nicht das Vollbild zurück, da die Kameras auch nur Halbbilder aufzeichnen. Das heißt, dass zwischen dem ersten und zweiten Halbbild 1/50 Sekunden liegen, bei schnellen Bewegungen hat sich ein Objekt zwischen den beiden Bildern soweit bewegt, dass Ausfransungen auftreten, da die Bilder nicht richtig „zusammenpassen“.

FBAS Codierung des Fernsehsignals

Das Fernsehsignal wurde ursprünglich als schwarz-weiß BAS-Signal entwickelt. Unter dem BAS-Signal versteht man das komplette Fernsehsignal für die schwarz-weiß Bildübertragung, das sich aus dem Bildsignal (B), dem Austastsignal (A) und dem Synchonisationssignal (S) zusammensetzt (BAS = Bild-Austast-Synchron-Signal).

Die Abbildung 2.10 zeigt eine Zeile eines Fernsehsignals, wie man es mit einem Oszilloskop darstellen könnte, wenn ein Graubalkentestbild verwendet wird. Das BAS-Signal besteht dabei aus den folgenden Punkten (vgl. auch entsprechend in Abbildung 2.10):

1. Das Bild beginnt mit einem weißen Balken
2. Daran schließen sich immer dunkler werdende graue Balken an.
3. Schwarz ist erreicht, dahinter liegt die vordere Schwarzschulter des Signals.
4. Der Horizontalsynchronimpuls. Der Spannungspegel liegt noch tiefer als bei „schwarz“. Diese Tatsache wird von einer Synchronisationsschaltung im Empfänger bemerkt, die dann den Rücklauf des Elektronenstrahls zum linken Bildrand bewirkt.
5. Die hintere Schwarzschulter.
6. Anfang der neuen Zeile, wieder mit weiß.

Für das Farbfernsehen wurde die Farbe auf das Signal (FBAS = Farb-Bild-Austast-Synchron-Signal) in einer Weise aufmoduliert, so dass vorhandene S/W Geräte nicht unbrauchbar wurden.

In Abbildung 2.11 ist eine Zeile eines PAL-modulierten Fernsehbildes für die Normbalkenfolge mit einer Farbsättigung von 75% (European Broadcasting Union-Testsignal) dargestellt.

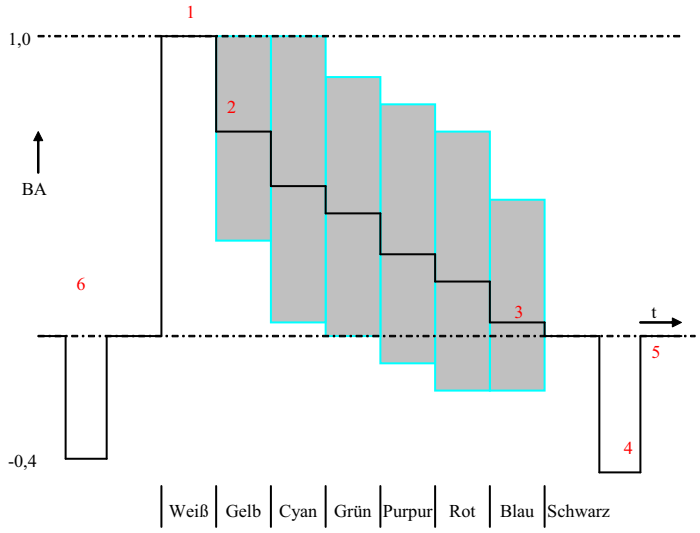


Abbildung 2.11: Das Farb-Fernsehsignal: FBAS

1. Das Bild beginnt mit einem weißen Balken.
2. Daran schließen sich weitere Balken mit den Farben in abnehmender Helligkeit an. Man erkennt, im Bild blau eingefärbt, die übertragene Farbinformation. Die Farbintensität äußert sich in der Stärke des Farbsignals (hier: Höhe des grauen Bereichs im Diagramm), der Farbton in der Phasenlage relativ zu dem Farbträger (vgl. Abbildung 2.12).
3. Schwarz und die vordere Schwarzschulter. Bei schwarz und bei weiß ist keine Farbmodulation zu erkennen, schwarz und weiß enthalten auch nur Helligkeits- und keine Farbinformation. Da bei der von PAL und NTSC verwendeten Quadraturmodulation auch kein Träger übertragen wird, im Gegensatz zu der bei SECAM verwendeten Frequenzmodulation, kann anhand diesen Merkmals PAL/NTSC von SECAM unterschieden werden. Bei SECAM wäre auf dem Helligkeitssignal für weiß und schwarz der (unmodulierte) Farbträger zu erkennen, auf dem Oszilloskop würden diese für das bloße Auge genauso wie ein farbiger Balken aussehen.
4. Der Synchronimpuls.
5. Die hintere Schwarzschulter mit dem PAL-Burst. Normalerweise ist der Farbträger wie oben erwähnt unterdrückt, nur während dieses kurzen Zeitraums „bricht er durch“; es werden ca. 10 Sinuswellen des Farbträgers direkt übertragen; der Zweck ist folgender: der Schaltkreis, der im Empfänger den Farbträger neu erzeugt, wird während dieses Zeitraums jeweils auf den Farbträger des Senders abgestimmt, für den Rest der Zeile kann er dann auf Basis dieser Abstimmung selbständig arbeiten.
6. Der Anfang der nächsten Zeile.

Bei dem Farbbalkentestbild (in Abbildung 2.12 ist das Oszillogramm dargestellt) werden eine Anzahl farbiger vertikaler Balken erzeugt, angefangen mit weiß über sämtliche mit jeweils zwei der Grundfarben RGB darstellbaren Farben und den Grundfarben selbst in absteigender Helligkeit bis hin zu schwarz. Dieses Bild ist besonders gut zur Fehlersuche mit einem Oszilloskop geeignet, da sämtliche Bildzeilen (außer denen in der Bildaustastlücke) die gleiche Information tragen. So kann das Oszilloskop mit der Zeilenfrequenz des Fernsehsignals betrieben werden, und bei entsprechender Triggerung bekommt man ein Bild entsprechend der Abbildungen 2.10, 2.11 oder 2.12. Da alle Zeilen gleich sind, werden alle Zeilen genau übereinander vom Oszilloskop geschrieben.

In der Abbildung 2.12 ist noch einmal die Aufmodulierung des Farbsignals dargestellt.

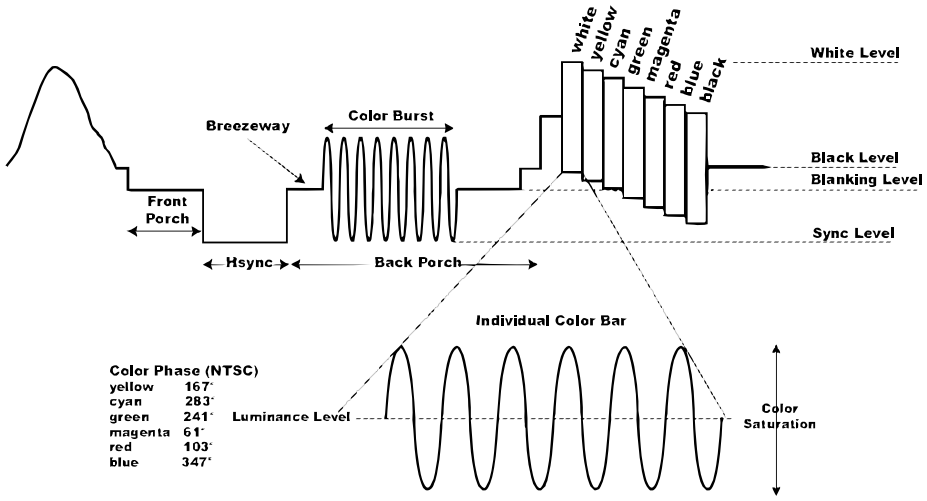


Abbildung 2.12: Farbmodulation im FBAS-Signal

2.3.2 Framegrabbing

Zum Digitalisieren des Videosignals stehen Pinnacle PCTV Stereo Karten zur Verfügung. Diese Karten digitalisieren das analoge FBAS Composite Signal (wahlweise können sie auch RGB) und stellen es in Form seiner YUV Komponenten zur Verfügung (wobei eine „Auflösung“ des YUV Signals von 4:2:2 verwendet wird).

Ein Videosignal wird häufig in seine Helligkeit (Luminanz = Y), die Balance zwischen Rot und Grün (U), sowie die Balance zwischen den Gelb und Blau Anteilen zerlegt (V). Dies bietet den Vorteil, die Helligkeit mit einer höheren Genauigkeit zu verarbeiten als die Farbanteile. Da das menschliche Auge Helligkeitsunterschiede (ungefähr 600 kann der menschliche Sehapparat unterscheiden) deutlich besser auflösen kann als Farben, macht man sich dies in der Videoaufzeichnung zu nutze. Um die Datenmenge des Videosignals zu reduzieren wird nur der Y-Anteil mit voller Auflösung aufgezeichnet. Bei dem Beispiel von YUV 4:2:2 wird dann die Chrominanzauflösung horizontal halbiert, d.h. nur von jedem zweiten Pixel wird neben dem Helligkeitswert auch der Farbwert bestimmt. Einem menschlichen Betrachter fällt der Unterschied ohne direkten Vergleich nicht auf. Bei 4:2:0 wird die Farbauflösung horizontal und vertikal halbiert.

Des Weiteren ist eine Umrechnung von YUV in den RGB Farbraum relativ einfach per Matrizenmultiplikation zu erledigen [Hei96] [Ber96].

Probleme beim Framegrabbing

Probleme bei der Digitalisierung von analogem Material können verschiedene Ursachen haben. Zum einen kann es Probleme mit der Synchronisation der Videoquelle geben, z.B. wenn die analoge Quelle sehr starke Schwankungen in der Länge der Signale liefert. Das kann bei Videobändern durch Abnutzung und dadurch teilweise Längung des Bandes hervorgerufen werden. Dann reagiert das Aufzeichnungsgerät nicht mehr auf den Zeilenimpuls und das Bild wird z.B. am oberen Rand nach links gezogen. Eine fehlende Vertikal-Synchronisation führt zu einem „Durchlaufen“ des Bildes. An starken Kontrastübergängen neigt das analoge Bild auch zu Ausfransungen.

Speicherung der Aufnahmen

Wie weiter oben festgestellt ist der erste digitale Schritt nach der Framegrabber-Karte ein (unkomprimierter) YUV-Datenstrom. Dieser führt bei voller PAL-Auflösung von 768×576 zu einem Datenstrom von ca. 300 MBit/sec. Zum Vergleich: der Datenstrom einer Handelsüblichen DVD

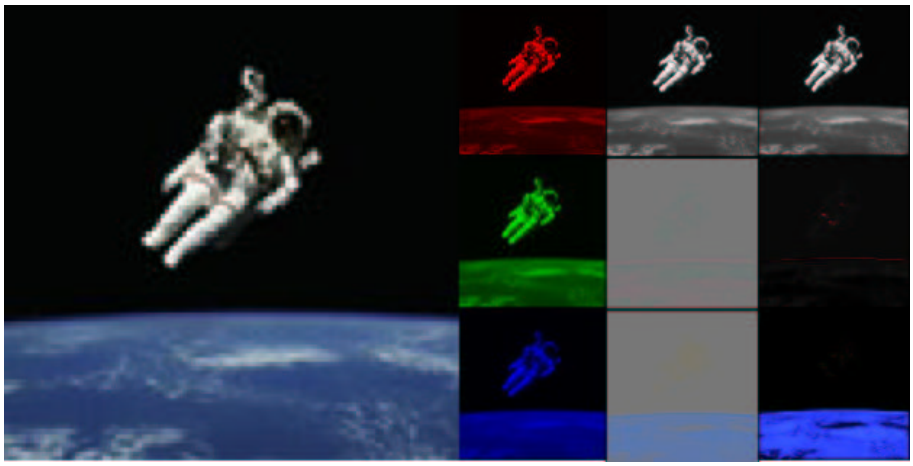


Abbildung 2.13: RGB YUV Layer-Vergleich

liegt ca. zwischen 2 und 12 Mbit/sec (je nach Kompressionsrate und Bildinhalt). Ein Datenstrom mit den gleichen Attributen wie oben jedoch nur in YUV mit 4:2:2 „Auflösung“ digitalisiert hat nur noch ca. 216 MBit/sec. Dabei wird der Y-Anteil mit 13,5 Mhz × 8 bit abgetastet, welches 108 MBit/sec entspricht. Die U- und V-Kanäle werden nur mit je 6,75 Mhz/sec × 8 bit abgetastet, welches je 54 MBit/sec entspricht.

2.3.3 Kompression von Videomaterial

Aufgrund der großen Datenmengen von digitalisiertem Videomaterial stellt sich die Frage, ob und auf welche Art man die Datenmengen reduzieren könnte. Durch die Aufzeichnung in YUV 4:2:2 wurde schon eine gewisse Menge an Daten durch Weglassen von Informationen eingespart. Für die Zwecke der Projektgruppe (PG) würde sogar nur der Y Kanal ausreichen, da die Kameras bisher keine Farbinformationen liefern. Zur besseren Vergleichbarkeit wird hier von einer Speicherung der Farbinformationen ausgegangen. Es könnte sich z.B. im Laufe der PG herausstellen, dass Farbinformationen zur Selektion der Zellen hilfreich sind.

Was bedeuten 216MBit/sec? Wenn man dies umrechnet, erhält man beeindruckende 25MiB/sec (MiB = MebiByte, 1MB = 1000KB, 1MiB = 1024KB, vgl. auch IEC 60027-2) an Daten, die es zu speichern gilt. Auch sollte man bedenken, dass diese Daten nicht nur einmal gespeichert und ausgewertet werden, sondern auch zur Überprüfung der Experimente archiviert werden müssen. Ein Vergleich mit einer handelsüblichen DVD zeigt, dass hier nur ungefähr max. 1,4 MiB/sec anfallen. Allerdings mit dem Nachteil, dass man Informationen verliert [IEC98].

Abbildung 2.13 beinhaltet noch ein Beispiel, welches die Aufteilung in die verschiedenen Kanäle zeigt und gut veranschaulicht, warum die YUV-Layer für eine Kompression besser geeignet sind, als die RGB-Ebenen. Wie man in der Abbildung 2.13 sieht, steckt die meiste Information im Y-Kanal und so können der U- und V-Kanal gut auf größtenteils schwarz reduziert werden. Von links nach rechts sieht man in Abbildung 2.13: Ursprungsbild, RGB Kanäle, YUV Kanäle, YUV Kanäle komprimiert [MV96] [Bar03].

Verlustfreie Kompression mit Huffman-Algorithmen

Die perfekte verlustfreie Kompression wäre eine bijektive Abbildung, die jegliche Redundanz aus dem Ursprungs-Material herausfiltert.

Als erstes wird hier die Kompression nach dem Huffman Algorithmus vorgestellt. Verwendung

findet er z.B. im HuffYUV-Codec, welcher, wie der Name schon vermuten lässt, einen YUV Datenstrom mit dem Huffman-Algorithmus komprimiert [RG].

Der Huffman Algorithmus gehört zu den so genannten „Wörterbuch Algorithmen“. Hierbei werden häufig vorkommende Daten durch kurze Ersetzungen repräsentiert. Dazu werden die Daten stochastisch vor der Codierung ausgewertet. An dem kurzen Beispiel String „HAAAALLO“ kann man die Vorgehensweise verdeutlichen:

Als erstes wird die Tabelle 2.1 mit den relativen Wahrscheinlichkeiten für das Auftreten von Symbolen angelegt.

H	1/8
A	4/8
L	2/8
O	1/8

Tabelle 2.1: Huffman Wahrscheinlichkeiten

Danach werden die zwei Symbole mit der kleinsten Wahrscheinlichkeit zusammengefasst und die Wahrscheinlichkeiten addiert. Diesen Symbolen wird jeweils die Ziffer 0 und 1 zugeordnet, indem sie dem bestehenden Code vorangestellt werden. Dieser Prozess wird wiederholt, bis alle Symbole zusammengefasst sind, und wir wieder die Gesamtwahrscheinlichkeit 1 erhalten.
Die am seltensten auftretenden Symbole „H“ und „O“ erhalten den längsten Code, und „A“ den kürzesten.
Aus der Codierungstabelle 2.2 lässt sich dann der Huffman Baum in Abbildung 2.14 generieren. Folgt man in dem Baum den Ästen, so erhält man den dem jeweiligen Symbol zugehörigen Code.

<table>
<tr><td>H</td><td rowspan="2">2/8</td><td>0</td><td rowspan="2">4/8</td><td>00</td><td rowspan="3">8/8</td><td>000</td></tr>
<tr><td>O</td><td>1</td><td>01</td><td>001</td></tr>
<tr><td>L</td><td>2/8</td><td></td><td>1</td><td>01</td></tr>
<tr><td>A</td><td>4/8</td><td></td><td>4/8</td><td></td><td>1</td></tr>
</table>

Tabelle 2.2: Aufbau des Huffman Baumes

Dieser Code ist eine Bijektion mit Präfixabgeschlossenheit, was bedeutet, dass man durch Hinzufügen einer Binärzahl zu einer gegebenen Repräsentation eines Zeichens kein anderes erhält.
Zum Beispiel ergibt „001“ (für „O“) durch hinzufügen von „0“ oder „1“ keinen gültigen Code. „1001“ oder „0001“ ist nicht in der Tabelle enthalten. Wenn der String von Anfang an richtig dekodiert wurde, ergibt sich der Anfang des folgenden Zeichens daraus, dass das vorhergehende Zeichen komplettiert wurde. Sobald der Durchlauf des Baumes in einem Blatt endet, hat man den Repräsentanten für das codierte Zeichen und weiß, dass die Codierung abgeschlossen ist und mit dem nächsten Bit ein neues Zeichen beginnt [Fie00].

LZW-Kompression

Der LZW- oder auch Lempel-Ziv-Welch-Algorithmus ist ein häufig bei Grafikformaten zur Datenkompression, also zur Reduzierung der Datenmenge, eingesetzter Algorithmus. Ein Großteil der Funktionweise dieses Algorithmus' wurde 1978 von Lempel und Ziv entwickelt und veröffentlicht (bekannt als LZ78) [ZL77], [ZL78]. Einige Detailverbesserungen wurden 1984 von Welch gemacht [Wel83]. Varianten dieser Kompressionsmethode werden unter anderem in den Formaten GIF, TIFF, JPEG und auch PNG/MNG eingesetzt [com].
Da für die PG eventuell das Format MNG (gesprochen [mi:ng]) in Frage kommt, soll hier die grobe Funktionsweise erläutert werden [RP].
Die LZW Algorithmen benutzen eine Kombination aus statischem und dynamischem Wörterbuch.

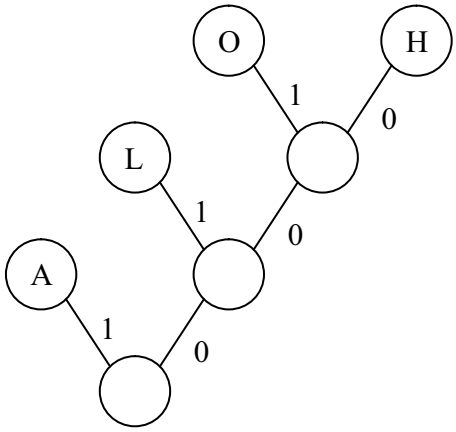


Abbildung 2.14: Huffman-Baum

Meist wird das statische Wörterbuch dabei fest in die En- und Decoder eingebaut, so dass es die zu komprimierende Datei nicht unnötig vergrößert. Das dynamische Wörterbuch ergibt sich direkt aus dem zu en- oder decodierenden Datenstrom.
Das Beispiel in Tabelle 2.3 soll die Vorgehensweise verdeutlichen.

Eingabe	statisches Wörterbuch
HAAAALLO	1H, 2A, 3L, 4O

Tabelle 2.3: LZW: statisches Wörterbuch

Um das dynamische Wörterbuch aufzubauen, wird die aktuell längste im Wörterbuch vorkommende Zeichenkette betrachtet, also hier „H“, und dann mittels der existierenden Schlüssel codiert. Dann wird zu dieser Zeichenkette ein weiteres Symbol aus dem Eingabestrom hinzuaddiert, und dieses Wort („HA“) bildet nun einen neuen Schlüssel. So wird das dynamische Wörterbuch aufgebaut. Das neu aufgenommene Wort kann jetzt zur Codierung der weiteren Eingabe genutzt werden. Dies wird fortgesetzt, bis der Eingabestrom komplett codiert ist.

In der Tabelle 2.4 wird das Beispielwort codiert. Die fett gedruckten Zeichen markieren die Zeichenkette, die neu in das dynamische Wörterbuch aufgenommen wird.

Eingabe	dynamisches Wörterbuch	Ausgabe
HAAAALLO	5HA	1
AAAALLO	5HA, 6AA	12
AAALLO	5HA, 6AA, 7AAA	126
ALLO	5HA, 6AA, 7AAA, 8AL	1262
LLO	5HA, 6AA, 7AAA, 8AL, 9LL	12623
LO	5HA, 6AA, 7AAA, 8AL, 9LL, 10LO	126233
O	5HA, 6AA, 7AAA, 8AL, 9LL, 10LO	1262334

Tabelle 2.4: LZW: dynamisches Wörterbuch

Die Decodierung verläuft entsprechend, wobei hier nicht wie bei der Codierung „vorausschauend“ das dynamische Wörterbuch aufgebaut wird, sondern „zurückblickend“ [Fie00], [Wel83], [ZL77], [ZL78].

2.3.4 Alternativen zur Aufzeichnung

Um für das Ziel der Projektgruppe, nämlich das automatische Verfolgen der Zellen, möglichst gutes Video-Ausgangsmaterial zu haben, gäbe es mehrere Möglichkeiten. Zum einen könnte die Aufzeichnung per Videorecorder umgangen und die Daten der Kamera direkt per Framegrabber Karte in digitaler Form aufzeichnet werden. Damit hätte man den sehr stark verlustbehafteten und andere Probleme bereitenden Weg über das analoge VHS Band umgangen. Dieses Vorgehen hat auch den Vorteil, dass an dem Versuchsaufbau nichts geändert werden müsste und die Aufzeichnung auf VHS parallel dazu weitergehen kann, indem das Signal von der Framegrabber-Karte an den VHS-Recorder weitergegeben wird.

Eine weitere Möglichkeit wäre, die Kamera gegen eine digitale Kamera mit gleichzeitig höherer Auflösung auszutauschen. Eine digitale Kamera könnte dann genauso wie derzeit der Videorecorder in definierten Zeitabständen Bilder machen, die ein angeschlossener Computer per IEEE1394 oder USB auslöst und von der Kamera abholt. Neben der einfacheren Anbindung an den Computer erhält man auch eine bessere Qualität, da man keine Umwege über analoge Signale geht und eine höher auflösenden und qualitativ eventuell auch besseren CCD hat.

Des Weiteren könnte man auch direkt schon eine Vorauswertung der Daten machen. Indem der Computer den Versuch beobachtet und nur bei Änderungen in den Bildern diese auch speichert. Das heißt, dass man bei Änderungen eine hohe zeitliche Auflösung der Bewegung in dem Versuch erhält, während Speicherplatz gespart wird, wenn keine Änderungen stattfinden. Neben den Bildern muss natürlich auch ein Timestamp mit gespeichert werden, um den Versuch zeitlich korrekt rekonstruieren zu können.

2.4 Digitale Bildverbesserung

In der digitalen Bildverarbeitung existieren nicht immer gestochen scharfe und voll ausgeleuchtete Bilder. Es können auch Daten korrumpiert werden, was einen weiteren Verlusst an Daten beinhaltet. Um diese Probleme zu reduzieren und zu beseitigen gibt es einige Algorithmen, die existierende Daten manipulieren, um es für den Menschen möglich zu machen, bestimmte Dinge einfacher zu erkennen (z.B. beim Hochpassfilter).

2.4.1 Wie sieht das Bild im Computer aus

Bei der digitalen Bildverarbeitung werden Bilder im Computer gespeichert. Um ein allgemeines Verständnis zu vermitteln, was genau im Computer passiert und wie später vorgestellte Algorithmen in dem Zusammenhang funktionieren, wird am Anfang dieses Kapitels ein Überblick darüber gegeben wie Bilder in einem Computer gespeichert werden. Mit diesem Verständnis soll es dem Leser einfacher gemacht werden sich in die Problematik der Digitalen Bildverarbeitung und Bildverbesserung einzudenken.

Im Allgemeinen ist bekannt, dass ein Computer nur digitale Daten speichern kann. In CellTrack wird mit schwarz-weiß Bildern gearbeitet. Bei diesem Bildformat wird normalerweise nur der Grauwert des Bildes abgespeichert. Dafür wird das Bild in Pixel[1] aufgeteilt. Das menschliche Auge kann nur Farb-Intensitätsunterschiede ab einer bestimmten Größe erkennen, deshalb wird das kontinuierliche Spektrum der Grautöne in 256 Elemente aufgeteilt. Dabei wird im Allgmeinen definiert, dass der Wert 0 der Farbe Schwarz entspricht und der Wert 255 Weiß darstellt. Diese Zahl kann in einem Byte gespeichert werden. Erst durch ein Wiedergabegerät, dass diese Zahlen interpretiert, werden diese Zahlen zu Bildern für das menschliche Auge.

[1] das ist eine allgemeine Abkürzung für Picture-Element

Abbildung 2.15: Ein Bild in seiner Grauwertdarstellung

Abbildung 2.16: Das zu Abbildung 2.15 gehörende Histogramm

2.4.2 Was ist ein Histogramm?

In einem Histogramm kann man die Häufigkeiten sämtlicher im Bild auftretender Grauwerte ablesen. Hierbei wird in der Regel ein 2-dimensionales Diagramm benutzt, welches auf der X-Achse alle Grauwerte enthält und auf der Y-Achse deren Häufigkeit im Bild gemessen in Pixeln darstellt (vgl. Abbildung 2.16). Es gibt auch andere graphische Darstellungen, wie z.B. ein kumulatives Histogramm, welches aus der Aufsummierung der Grauwerte errechnet wird.

2.4.3 Helligkeitskorrektur

Es kann bei einigen Bildern ausreichen, dass man nicht die komplette Bandbreite der Grauwerte ausnutzt. Der Mensch erkennt nicht zwingend, wenn zusammenhängende Grauwertgruppen auf einen Grauwert reduziert werden. Ein Beispiel ist, wenn die Pixel mit dem Wert 0,1,2,3 auf 0 abgebildet werden (die Pixel 4,5,6,7 auf 4, usw.). Dieses Beispiel wäre eine Variante der 64 Bit Quantisierung. Quantisierung hat meist den Vorteil das Bilder weniger Speicher benötigen und daher besser zu speichern sind. Eine spezielle Variante der Quantisierung ist die Binarisierung. Hierbei

wird das Bild nur auf die Farben 0 und 1 reduziert. Binarisierung wird meist genutzt um Objekte separat vom Hintergrund zu haben. Weitere Informationen zur Binarisierung und den Nutzen des Histogramms zur Schwellenwertbestimmung werden in den folgenden Kapiteln genauer besprochen.

Grauwertverschiebung

Bei Bildern die zu dunkel oder zu hell sind treten häufig Probleme für den Beobachter auf. So sind dunkle Bilder meist kontraststärker, jedoch in der Regel schwerer zu erkennen. Bei hellen Bildern ist dies invers, da sie besser zu erkennen sind, jedoch kontrastärmer für das menschliche Auge erscheinen. Wenn man das Bild insgesamt nur aufhellen oder verdunkeln will, kann man die Grauwerte einfach um einen konstanten Wert verschieben [Ste93].

$$g'(x,y) = g(x,y) + c_1 \tag{2.1}$$

Hierbei ist $g'(x,y)$ der neue Wert an der Stelle (x,y), $g(x,y)$ ist der Originalwert und c_1 ist die Konstante, um die der Wert verschoben wird.

Hierbei muss beachtet werden, dass der Definitionsbereich der Farben bei dieser Kalkulation nicht überschritten wird. Sonst ist mit Informationsverlust zu rechnen und die Wiedergabe produziert unter Umständen falsche Bilder.

Es ist auch möglich die Farbskala des Bildes zu strecken, indem man sie einfach, wie in der folgenden Gleichung beschrieben, mit einen Faktor multipliziert:

$$g'(x,y) = c_2 \cdot g(x,y) + c_1 \tag{2.2}$$

Als Neuerung im Vergleich zur Gleichung 4.1 ist c_2 der neue Streckungsfaktor. Da auch bei dieser Funktion schnell der Definitionsbereich verlassen wird, nutzt man am besten eine Funktion in der Streckungs- und Verschiebungsfaktoren wohldefiniert sind.

$$g'(x,y) = (g(x,y) - g_{min}) \cdot \frac{G_{max} - G_{min}}{g_{max} - g_{min}} \tag{2.3}$$

Die Variable G_{max} (G_{min}) entspricht dem maximalen (minimalen) zur Verfügung stehenden Grauwert. Die maximalen und minimalen Grauwerte im Bild werden durch die Variablen g_{max}, g_{min} dargestellt.

Es gibt auch andere nichtlineare Funktionen um den Grauwert zu verändern. Varianten sind zum Beispiel die Quadrierung oder Wurzelziehen. Im Folgenden werden die Funktionen hier nur aufgeführt, es wird aber nicht weiter auf ihren Nutzen eingegangen (c_1 und c_2 sind hier Konstanten).

$$\begin{aligned} g'(x,y) &= c_1 \cdot g^2(x,y) + c_2 \\ g'(x,y) &= c_1 \cdot \sqrt{g(x,y)} + c_2 \\ g'(x,y) &= \ln(c_1 \cdot g(x,y)) + c_2 \\ g'(x,y) &= e^{c_1 + g(x,y)} + c_2 \end{aligned}$$

Histogrammglättung

Bei der Histogrammglättung werden Bilder, die bedingt durch eine schlechte Grauwertverteilung nicht scharf erkennbar sind, durch einen automatischen Mechanismus so bearbeitet, dass aus der schlechten Grauwertverteilung eine bessere Verteilung wird, und das Bild auf diese Weise schärfer erscheint. Wann ist eine Grauwertverteilung schlecht? Wenn bei einem Bild z.B. 256 Grauwerte zur Verfügung stehen, allerdings das Bild nur die Grauwerte 20 bis 80 benutzt, erscheint das Bild sehr dunkel und es sind nur schwer Konturen zu erkennen. Ähnlich verhält es sich, wenn der benutzte Grauwertebereich im Bild im Bereich von 200 bis 250 liegt. Hierbei erscheint das Bild im Gegensatz zum ersten Beispiel hell, allerdings sind auch hier die Konturen nur schlecht erkennbar.

Mit der Histogrammglättung ist es also möglich, Bilder, die bedingt durch eine schlechte Grauwertverteilung zu hell bzw. zu dunkel sind, in Bilder umzuwandeln, welche eine breitere Palette an Grautönen benutzen.

Die Histogrammglättung, auch Histogrammebnung genannt, funktioniert wie folgt: Zur Verfügung stehen $\mid L \mid$ Grauwerte f. Ziel ist es, die Grauwerte f durch neue Grauwerte f' so zu ersetzen, dass der oben beschriebene gewünschte Effekt eintritt.

Dies geschieht, wenn man f' aus f so berechnet: Zähle alle Pixel im Bild zusammen, die einen Grauwert $\leq f$ haben und teile anschließend durch die Anzahl aller Pixel im Bild. Wir erhalten eine Zahl aus dem Intervall $[0, 1]$, die als prozentuales Vorkommen der Pixel mit Grauwert $\leq f$ verstanden werden kann. Diese Verteilung wird nun auf das Intervall $[0, L-1]$, also auf das Intervall aller Grauwerte „ausgedehnt“ , indem man die Zahl mit $(L-1)$ multipliziert. Der so erzeugte Wert ist jetzt der neue Grauwert f' für jedes Pixel im Bild, das zuvor den Grauwert f hatte [GW02].

$$f' = \frac{L-1}{\text{Gesamtpixelzahl}} \cdot \sum_{k=0}^{f} h(k), \tag{2.4}$$

wobei $h(k)$ die Pixelzahl mit Grauwert k im Bild berechnet.

2.4.4 Rauschelimination

In diesem Kapitel werden Prinzipien vorgestellt, die angewandt werden können, um Rauschen, kleine Kratzer im Bild oder andere kleine störende Elemente im Bild zu minimieren und im Idealfall ganz zu entfernen.

Die Prinzipien, die hier beschrieben werden, basieren auf der Technik des „Filterns“. Ein Filter oder eine Maske ist eine Umgebung, ein kleines Fenster, das auf ein Bild „gelegt“ wird. Die Idee dabei ist, dass bei der Modifizierung eines jeden Pixels die Pixel, die in unmittelbarer Umgebung zu diesem Pixel liegen, bei der Berechnung mit einbezogen werden. Genau dies passiert bei dem Prozess des Filterns. Nun kann man durch Variieren der Größe des Filters und durch unterschiedliche Gewichtung der einzelnen umliegenden Pixel unterschiedliche Resultate erhalten.

Wenn man nun für jedes Pixel im Bild unter Benutzung eines geeigneten Filters einen neuen Grauwert bestimmt, erhält man den oben beschriebenen Effekt, dass Rauschen aus dem Bild entfernt wird. Man spricht hierbei, wenn auf das ganze Bild, also für jedes Pixel, ein Filter „angesetzt“ wird, auch von einer „Faltung“ des Bildes f mit einem Filter h. Die Faltung wird durch folgende Formel beschrieben:

$$g(m,n) := f(m,n) * *h(m,n) := \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f(k,l)h(m-k, n-l) \tag{2.5}$$

Bei dem Prozess des Filterns wird das Bild abhängig von der Wahl der Funktion $h(m, n)$ verschmiert. Alle Pixel werden unter Berücksichtigung der umliegenden Pixel verändert. Dabei wird das ganze Bild an sich unscharf und Details verschwinden. In den folgenden Unterkapiteln werden nun verschiedene Filter und deren Resultate vorgestellt. Anschließend wird eine neue Darstellungweise eines Bildes, der Ortsfrequenzbereich (vgl. Kapitel 2.4.4), präsentiert, in der viel schneller „gefiltert“ werden kann.

Filtern im Ortsbereich

Averaging Filter

Eine Möglichkeit wie Filter aussehen können, ist die, dass von allen Pixeln, die in der Umgebung liegen, ein Durchschnittsgrauwert gebildet wird. Diese Art von Filter nennt man „Averaging Filter“. Hierbei können - müssen aber nicht - alle Pixel aus der Umgebung gleich gewichtet sein.
Ein Averaging Filter kann z.B. ein 3x3 Fenster sein, also eine Größe von 9 Pixeln haben. Was jetzt beim Filtern passiert, ist, dass alle Grauwerte unterhalb des Filters gleich gewichtet werden und hieraus der Mittelwert genommen wird. Der so entstandene Wert wird als neuer Grauwert für den Bildpunkt, der unterhalb des Filters in der Mitte liegt, gewählt.

Medianfilter

Medianfilter eignen sich besonders dann, wenn in dem Bild ein sogenannter „Salt-and-pepper noise“ vorliegt. Im Gegensatz zu dem Averaging Filter, bei dem der Durchschnittsgrauwert berechnet wird, bestimmt der Medianfilter den mittleren Grauwert.
Hat der Filter z.B. eine Größe von 9 Pixeln, wird der Grauwert als neuer Grauwert für den jeweiligen Bildpunkt genommen, der sich von allen Grauwerten unterhalb des Filters in sortierter Reihenfolge in der Mitte - in unserem Beispiel also an der 5. Position - befindet.

Filtern im Ortsfrequenzbereich

Der Ortsfrequenzbereich

In diesem und den folgenden Unterkapiteln geht es um eine andere Darstellungsweise eines Bildes, den Ortsfrequenzbereich. Jede Funktion läßt sich als Summe von sinusförmigen Schwingungen unterschiedlicher Frequenz und Phase darstellen. Bei der sogenannten „Diskreten Fouriertransformation“ wird das Bild aus dem Ortsbereich in den Ortsfrequenzbereich transformiert. Jeder Grauwert, jedes Pixel in dem Bild wird Teil mehrerer „Frequenzen“. Sei $f : \{(m, n)|m = 0, 1, ..., M - 1, n = 0, 1, ..., N - 1\} \to \mathbb{R}$, $k = 0, ..., M - 1$ und $l = 0, ..., N - 1$, dann gilt für die zweidimensionale diskrete Fouriertransformation (DFT) [GW02]:

$$F(k, l) := \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) \cdot exp[-2i\pi(km/M + ln/N)]. \tag{2.6}$$

Beim Vergleich eines Bildes im Ortsbereich mit der zugehörigen Transformierten im Ortsfrequenzbereich stellt man fest, dass Kanten, (starke) Wechsel der Grautöne und Rauschen zu einem hohen Anteil an hohen Frequenzen führen, und dass glatte, „ruhige“ Flächen sich eher in den niedrigen Frequenzen widerspiegeln, also dort zu einem hohen Anteil führen. Bei speziellen Bildern, zum Beispiel bei der Fotografie eines Schachbretts, kann man beobachten, dass sich die Orientierung auffälliger Kantenverläufe zum Teil auch auf die Orientierung der hohen Frequenzen innerhalb der Transformierten auswirken.

Um die Transformierte eines Bildes zu berechnen, kann man den Fast-Fourier-Transformationsalgorithmus (FFT) benutzen, der eine Rechenzeit von $O(n \cdot log(n))$ hat, wobei n die Anzahl der Bildpunkte im Eindimensionalen ist. Separierbarkeit:

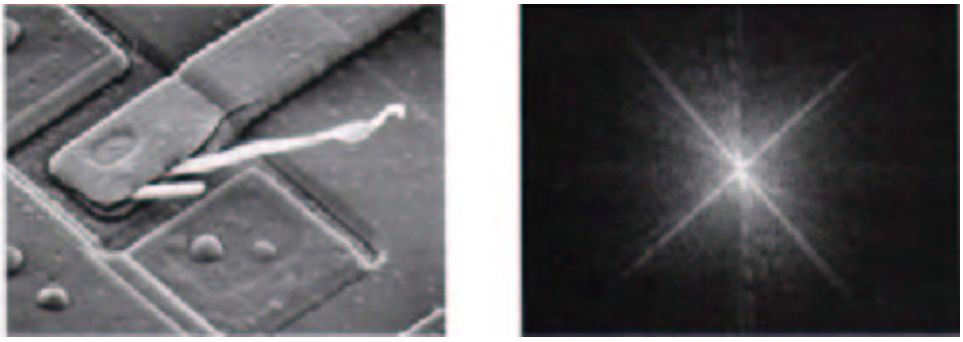


Abbildung 2.17: Ein Bild mit seiner zugehörigen Darstellung im Ortsfrequenzbereich

$$f(m,n) \rightarrow F_m(k,n) \rightarrow F(k,l) \tag{2.7}$$
$$f(m,n) \rightarrow F_n(k,n) \rightarrow F(k,l) \tag{2.8}$$

Wegen der Separierbarkeit der Fourier-Transformation kann man durch zweimaliges Hintereinanderausführen des FFT-Algorithmus ein Bild somit in einer Rechenzeit von $O(n \cdot m \cdot (\log(n) + \log(m)))$ transformieren, wobei $n \cdot m$ der Auflösung des Bildes entspricht.

Das Filtern im Ortsfrequenzbereich

Ein großer Vorteil des Filterns im Ortsfrequenzbereich ist, dass das Filtern hier nur noch eine Multiplikation an Kosten verursacht und man so die kostenintensivere Doppelsumme bei der Faltung im Ortsbereich umgehen kann. Faltungssatz:

$$g(m,n) := f(m,n) * *h(m,n) := \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f(k,l)h(m-k,n-l) \tag{2.9}$$
$$G(k,l) = F(k,l) \cdot H(k,l) \tag{2.10}$$

Nach obigen Bemerkungen läßt sich nun das gewünschte Ziel, das Rauschen zu eliminieren, im Ortsfrequenzbereich so erreichen: Man kappt die hohen Frequenzen. Diese etwas salopp formulierte Methode gestaltet sich allerdings doch etwas schwierig, da beim Entfernen der hohen Frequenzen nicht nur das unerwünschte Rauschen, sondern auch z.T. relevante Information aus dem Bild verloren geht, da wie im vorigen Unterkapitel schon erwähnt wurde, jede Frequenz, also auch die hohen Frequenzen, Informationen über „alle“ Pixel enthält. Entfernt man nun die hohen Frequenzen geht dabei auch z.T. relevante Information über jedes Pixel verloren. Das Bild wird unscharf.

2.5 Segmentierung von Bilddaten

Segmentierung und Tracking sind die Hauptthemen des Projektes. Aufgabe der Bildsegmentierung ist die Zuordnung von Pixeln (einzelne Bildpunkte) zu sinnvollen Objekten. Mit Hilfe der Segmentierung sollen Tumorzellen in dem vorhandenen Bildmaterial identifiziert werden, damit ein Tracking dieser Zellen verbessert werden kann.

Ein Segment ist ein zusammenhängender Bildbereich, in dem alle enthaltenen Pixel die gleiche Eigenschaft haben, beispielsweise Farbton oder Helligkeit. Man spricht von einer vollständigen

Segmentierung, wenn jedes Pixel (mindestens) einem Segment zugeordnet wird. Bei einer überdeckungsfreien Segmentierung wird jedes Pixel höchstens einem Segment zugeordnet. Bei einer vollständigen und überdeckungsfreien Segmentierung ist jedes Pixel also genau einem Segment zugeordnet. Eine Segmentierung nennt man zusammenhängend, wenn jedes Segment ein zusammenhängendes Gebiet bildet. Es sind viele Verfahren zur automatischen Segmentierung bekannt. Grundsätzlich werden sie in pixel-, kanten- und regionenorientierte Verfahren eingeteilt. Zusätzlich unterscheidet man noch modellbasierte Verfahren, bei denen man von einer bestimmten Form der Objekte ausgeht. Die Grenzen zwischen den Verfahren sind oft fließend. Auch kann man verschiedene Verfahren kombinieren, um bessere Ergebnisse zu erzielen. Natürlich kann man auch in einem nichtautomatischen Verfahren die Segmentierung ausführen, indem ein Benutzer eine Einteilung vornimmt. Zwischen den vollautomatischen und den manuellen Verfahren gibt es noch die Möglichkeit der semiautomatischen Bearbeitung.

2.5.1 Probleme

Oftmals ist die Qualität einer Segmentierung nicht optimal. In diesen Fällen kann man ein besseres Verfahren wählen, oder die Ergebnisse optimieren, indem man eine Vor- oder eine Nachbearbeitung anschließt. Beides kann sowohl automatisch (wenn man die Probleme des Prozesses bereits identifiziert hat), als auch manuell erfolgen. Ein Problem vieler Segmentierungsalgorithmen ist die Anfälligkeit für ungleichmäßige Beleuchtung innerhalb des Bildes. Dies kann dazu führen, dass immer nur ein Bildteil korrekt segmentiert wird, in den anderen die Segmentierung aber unbrauchbar ist. Viele Störungen wie Helligkeitsunterschiede und Rauschen kann man mit einer Vorbearbeitung ausgleichen. Häufige Probleme sind beispielsweise Übersegmentierung (zu viele Segmente) und Untersegmentierung (zu wenige Segmente). Dem kann man begegnen, indem man das Verfahren um Wissen der zu verarbeitenden Daten anreichert, im einfachsten Fall kann man die erwartete Anzahl der Segmente angeben. Außerdem kann man einen nachfolgenden Klassifikationsschritt einfügen, um gleich klassifizierte Segmente zusammenzufassen. Natürlich können die Segmente auch per Hand zusammengefasst werden. Viele Algorithmen arbeiten nur auf einkanaligen Graustufenbildern. Bei der Verarbeitung von Mehrkanalbildern (zum Beispiel Farbbildern) bleiben Informationen meist ungenutzt. Man benötigt weitere Bearbeitungsschritte, um mehrere einkanalige Segmentierungen zusammenzufassen.

2.5.2 Pixelorientierte Segmentierung

Pixelorientierte Verfahren treffen für jeden einzelnen Bildpunkt eine Entscheidung, ob er zu einem bestimmten Segment gehört oder nicht. Diese Entscheidung kann, aber muss nicht, durch die Umgebung beeinflusst sein. Pixelorientierte Verfahren sind meist einfach zu berechnen, liefern aber erstmal keine zusammenhängenden Segmente. Das verbreitetste Verfahren ist sicherlich das Schwellenwert-Verfahren [GW02].

Schwellenwert-Verfahren

Wenn Objekte im Bild nicht zu eng beieinander liegen und ihre Grauwerte sich klar vom Hintergrund trennen, ist das einfachste Segmentierungsverfahren das Schwellenwert-Verfahren in einem Grauwert-Bild. Viele Objekte oder Bildregionen werden durch relativ konstante Helligkeitswerte auf ihrer Oberfläche charakterisiert. Dadurch kann ein Schwellenwert definiert werden, der Objekt und Hintergrund voneinander trennt. Schwellenwertverfahren sind effizient, schnell und in einfachen Szenen ist eine vollständige Segmentierung möglich [GW02].

Folgende Arten von Schwellenwertverfahren können unterschieden werden:

- Globale Schwellenwertverfahren
  Hier wird ein einziger Schwellenwert für das ganze Bild verwendet. Das funktioniert aber nur, wenn alle Objekte die gleichen einheitlichen Grauwerte haben und sich stark vom ebenfalls

einheitlichen Hintergrund abheben.

$$g(x,y) = \begin{cases} 1 & f(i,j) \geq T \\ 0 & f(i,j) < T \end{cases} \tag{2.11}$$

Wobei $g(x,y)$ der Grauwert im Zielbild g an der Stelle (x,y) darstellt, der hier entweder den Wert 1 für helle Ausgangspixel und den Wert 0 für dunkle Pixel im Originalbild annimmt.

- Adaptive Schwellenwertverfahren
  Das Bild wird in verschiedene Regionen aufgeteilt, die jede einen eigenen Schwellenwert haben (vgl. Gleichung 2.11).

- Band-Schwellenwertverfahren
  Segmentiert das Bild in Regionen von Pixeln mit einem Grauwert aus einer Menge D und Hintergrund. Das ist nützlich, wenn man zum Beispiel Blutzellen segmentieren möchte, da hier ein Grauwert-Intervall das Zytoplasma darstellt, der Hintergrund ist heller und der Zellkern dunkler.

$$g(x,y) = \begin{cases} 1 & f(i,j) \in D \\ 0 & \text{sonst} \end{cases} \tag{2.12}$$

- Semi-Schwellenwertverfahren
  Hier wird der Hintergrund schwarz eingefärbt, die Objekte behalten aber ihre ursprünglichen Grauwerte.

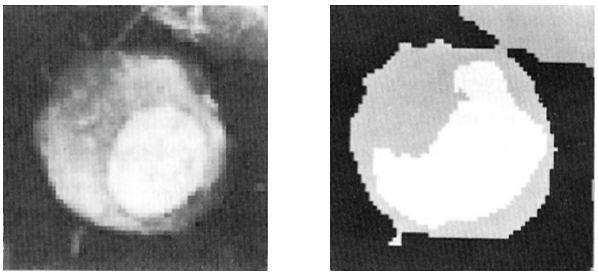
$$g(x,y) = \begin{cases} f(i,j) & f(i,j) \geq T \\ 0 & f(i,j) < T \end{cases} \tag{2.13}$$

Ein entscheidendes Kriterium bei allen Schwellenwertverfahren ist die Wahl eines geeigneten Schwellenwertes (vgl. Abbildung 2.18). Hierzu wird das Histogramm (vgl. Abbildung 2.19) zu Rate gezogen. Durch lokale Maxima erkennt man die Grauwerte, bei denen die verschiedenen Segmente liegen. Optimalerweise ist das Histogramm bimodal, das heißt, es lassen sich zwei klar voneinander getrennte Maxima erkennen. Man kann nun den Schwellenwert als Mittelwert zwischen diesen beiden Maxima wählen. Eine andere Herangehensweise ist, den Schwellenwert auf das lokale Minimum zwischen diesen Maxima zu legen. Hiermit wird eine bessere Trennung erreicht. Bearbeitet man immer wieder Bilder aus der gleichen Quelle, kann man oftmals einen einmal gewählten Schwellenwert auf alle diese Bilder anwenden [Son99].

Erkennung von Punkten/Linien

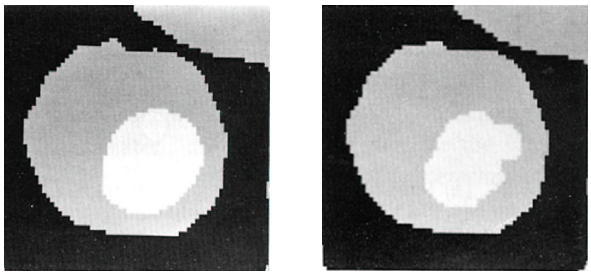
Vereinzelte Punkte, 1-Pixel-breite Linien und Kanten sind die drei häufigsten Unstetigkeiten in einem digitalen Bild. Die übliche Vorgehensweise, um diese Unstetigkeit zu lokalisieren ist eine Maske über das Bild laufen zu lassen. Das Verfahren arbeitet hierbei folgendermaßen: Eine 3x3 Maske (vgl. Abbildung 2.20(a)) wird zentriert über einen betrachteten Bildpunkt gelegt. Dann werden alle unter der Maske liegenden Punkte mit den Werten der Maske multipliziert und es wird die Summe aller so gewichteten Werte gebildet:

$$R = w_1 z_1 + w_2 z_2 + ... + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$



(a) Originalbild

(b) Schwellenwerte zu niedrig gewählt



(c) Schwellenwerte optimal

(d) Schwellenwerte zu hoch gewählt

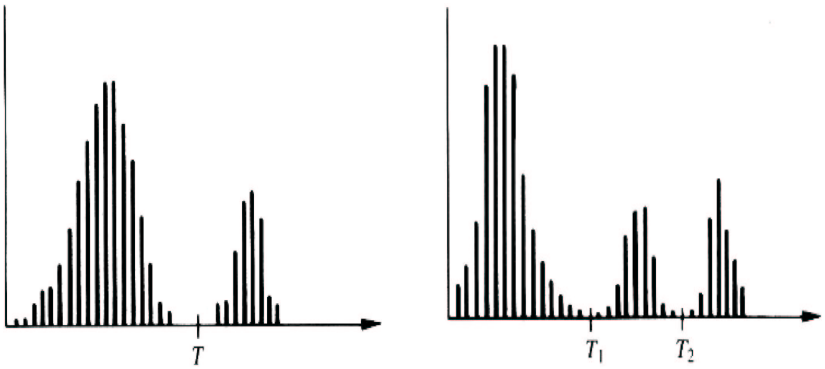
Abbildung 2.18: Auswirkungen der Wahl des Schwellenwertes

Wobei w_i die Gewichtung, und z_i den Grauwert des Pixels an Position i bedeutet. Diese Summe R (bei der Wahl von Abbildung 2.20(b) oder 2.21 (a)-(d) als Maske) ergibt bei gleichmäßigen Flächen den Wert 0. Je mehr sich der Mittelpunkt von seinen Nachbarn unterscheidet, umso größer wird R. Sobald R einen gegebenen Grenzwert T übersteigt, kann man davon ausgehen, einen isolierten Bildpunkt gefunden zu haben. So wird für alle Bildpunkte nacheinander verfahren, bis sich ein Ergebnisbild ergibt, in dem die isolierten Bildpunkte deutlich erkennbar sind. Dieses Verfahren eignet sich nur zur Segmentierung von Punkten mit einer festen Größe von 1 Pixel.

Etwas komplexer als die Punkterkennung ist das Erkennen von 1-Pixel-breiten Linien in einem Bild. Um die Linie in jeder Lage (Horizontal, +45°, Vertikal, -45°) entdecken zu können, sind vier Masken nötig und die Koeffizienten werden den Umständen angeglichen, damit die Summe R bei gleichmäßigen Flächen wieder den Wert 0 ergibt (vgl. Abbildung 2.21) [GW02].

2.5.3 Kantenorientierte Verfahren

Im Gegensatz zu den Schwellenwertverfahren, bei welchem die Grenzen zwischen den Objekten in einer Projektion des Bildes auf ein Grauwert-Histogramm gesucht wird, wird mit der Kantenerkennung versucht, die Grenzen zwischen den Objekten im Bild selbst zu finden. Hierbei wird davon ausgegangen, dass zwischen zwei Objekten im Bild eine Kante existiert. Eine Kante bedeutet in diesem Zusammenhang eine starke Grauwert-Änderung. Probleme treten bei diesen Verfahren meist dadurch auf, dass für die Kantenfindung auch ein Schwellenwert festgelegt werden muss und die Nachteile des Schwellenwert-Verfahrens auch hier gelten.
Befindet sich beispielsweise in einem Bild ein Grauwertgradient, so kann es sich hier entweder um die Grenze zwischen zwei Objekten, oder um die objekteigene Grauwert-Änderung handeln. Weiterhin unterscheiden sich Ergebnisse bei verschiedener Vorverarbeitung der Bilder stark. Es muss also eine geeignete Vorverarbeitung gefunden werden. Probleme bereitet auch, dass Kanten nicht unbedingt geschlossene Linienzüge bilden. Sollen jedoch Objekte gefunden werden, so müssen die Linienzüge geschlossen sein. Die Ergebnisse eines Algorithmus können Polygone (bzw. Linien) sein, aber manche Operationen liefern die Kanten auch als speziell eingefärbte Pixel. Obwohl das Er-



(a) bimodal (b) multimodal

Abbildung 2.19: Grauwert-Histogramme

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

(a) Grundmaske

-1	-1	-1
-1	8	-1
-1	-1	-1

(b) Punkterkennung

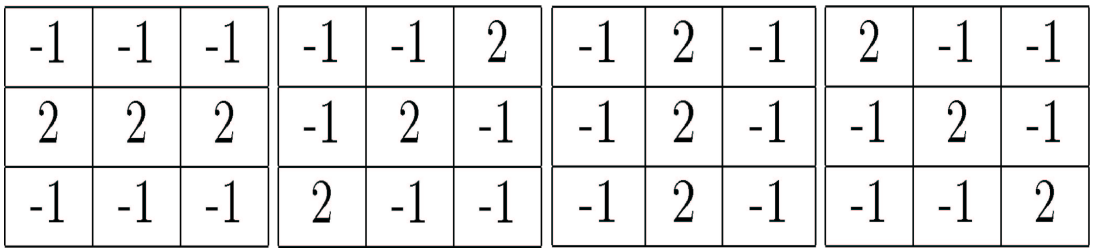
Abbildung 2.20: Masken

kennen von Punkten und Linien in der Segmentierung eine Rolle spielt, ist die Kantenfindung der häufigste Ansatz um wesentliche Unstetigkeiten im Bild zu finden, da Punkte und dünne Linien in den wenigsten praktischen Anwendungen vorkommen. Im Folgenden wird vorausgesetzt, dass die fraglichen Regionen ausreichend homogen sind, so dass der Wechsel zwischen ihnen allein durch die Graulevel-Unterschiede festgelegt werden kann.

Die grundlegende Idee der meisten Kantenerkennungs-Algorithmen ist die Berechnung lokaler Ableitungen. Abbildung 2.22 verdeutlicht dieses Prinzip: Wenn man eine Scanline vertikal über das linke Bild (weißer Streifen auf dunklem Hintergrund) laufen lässt, erhält man eine Grauwert-Linie mit einer positiven Flanke für den weißen Streifen. Bildet man hiervon nun die erste Ableitung, bedeutet ein positives Extremum eine (von links nach rechts) dunkel-hell-Kante und ein negatives Extremum eine hell-dunkel-Kante. Die zweite Ableitung ist meist ebenfalls hilfreich, da jede Nullstelle eine Kante markiert. Die erste Ableitung in jedem Bildpunkt entspricht dem Betrag des Gradienten in diesem Punkt und die zweite Ableitung entspricht dem Laplace-Operator [GW02].

2.5.4 Regionenorientierte Verfahren

Diese Verfahren betrachten Punktmengen als Gesamtheit und versuchen dadurch zusammenhängende Objekte zu finden. Häufige Verwendung finden das „Bereichswachstumsverfahren“ und „Split and Merge“.



(a) Horizontal (b) +45° (c) Vertikal (d) -45°

Abbildung 2.21: Masken zur Linienerkennung

Basisformulierung

Wenn R die gesamte Bildregion repräsentiert, kann man den Segmentierungsprozess als eine Aufteilung von R in n Subregionen sehen, so dass gilt:

(a) $\bigcup_{i=1}^{n} R_i = R$

(b) R_i ist eine verbundene Region, $i = 1, 2, ..., n$,

(c) $R_i \cap R_j = \emptyset$ für alle i und j mit $i \not= j$

(d) $P(R_i) = TRUE$ mit $i = 1, 2, ..., n$

(e) $P(R_i \cup R_j) = FALSE$ mit $i \not= j$

mit $P(R_i)$ als logische Eigenschaften aller Punkte in der Menge R_i. Bedingung (a) besagt, dass die Segmentierung komplett, also jeder Pixel in einer Region sein muss. Bedingung (b) verlangt, dass alle Punkte in einer Region miteinander in Kontakt stehen müssen. Die dritte Bedingung bedeutet, dass die Regionen disjunkt sein müssen. Bedingung (d) wiederum schreibt vor, dass alle Pixel einer Region die vorgeschriebenen Eigenschaften dieser Region erfüllen müssen. Die letzte Bedingung besagt, dass zwei Regionen unterschiedlich bezüglich ihrer Eigenschaften sein müssen, [GW02].

Bereichswachstumsverfahren

Bisher wurde versucht ein Bild in Regionen zu teilen, indem die Kanten zwischen verschiedenen Regionen gefunden werden sollen. Im Bereichswachstumsverfahren dagegen sollen die Regionen von „innen heraus“ gefunden werden. Und wie der Name schon sagt, handelt es sich um ein Verfahren, das kleinere Gruppen von Pixeln zu Größeren zusammenfasst. Der simpelste dieser Ansätze ist „Pixel Aggregation“, welcher mit einer Menge von Initialpunkten startet und zu Regionen wachsen lässt, indem genau die Nachbarn einem Initialpunkt zugeordnet werden, die die gleichen Eigenschaften (z.B. Grauwert, Textur, Farbe) haben.

Initialpunkte finden

Da man nicht davon ausgehen kann, dass die Initialpunkte ideal gewählt wurden (exakt 1 Punkt pro tatsächlichem Teilbereich im Bild), müssen anschließend noch nebeneinanderliegende Bereiche mit gleichen Merkmalen zusammengefasst werden. Abbildung 2.23 zeigt ein mögliches Verfahren zur Bestimmung der Initialpunkte. Das Bild wird zunächst in mehrere Fenster unterteilt, um Störungen z.B. durch unterschiedliche Beleuchtungsverhältnisse zu mindern. Dann werden Punkte markiert, deren Intensitätsgradienten einen festgelegten Wert nicht überschreiten. Dadurch wird verhindert, dass Punkte ausgewählt werden, die z.B. genau auf einer Kante liegen und somit für das Bereichswachstumsverfahren ungeeignet sind.

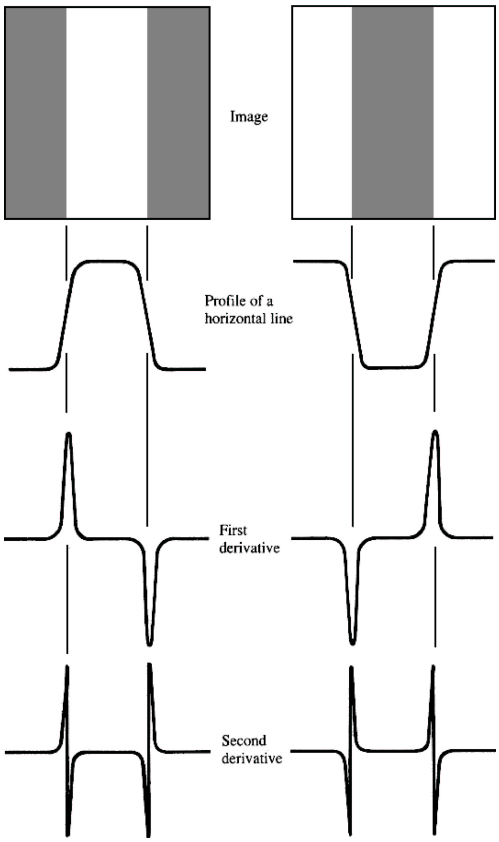


Abbildung 2.22: Kantenerkennung

Anschließend sollten überflüssige Punkte eliminiert werden. Dazu wird überprüft, ob der jeweils betrachtete Punkt von einem anderen aus zu erreichen ist, ohne auf der entsprechenden Wegstrecke einen zu großen Intensitätsgradienten (also eine Objektgrenze) überqueren zu müssen. Da es zu aufwendig wäre, alle möglichen Wege zwischen zwei Punkten zu untersuchen, beschränkt man sich in der Regel auf die simple Verbindung per direktem Streckenzug und untersucht nur die Punkte, die auf dieser Gerade liegen [GW02].

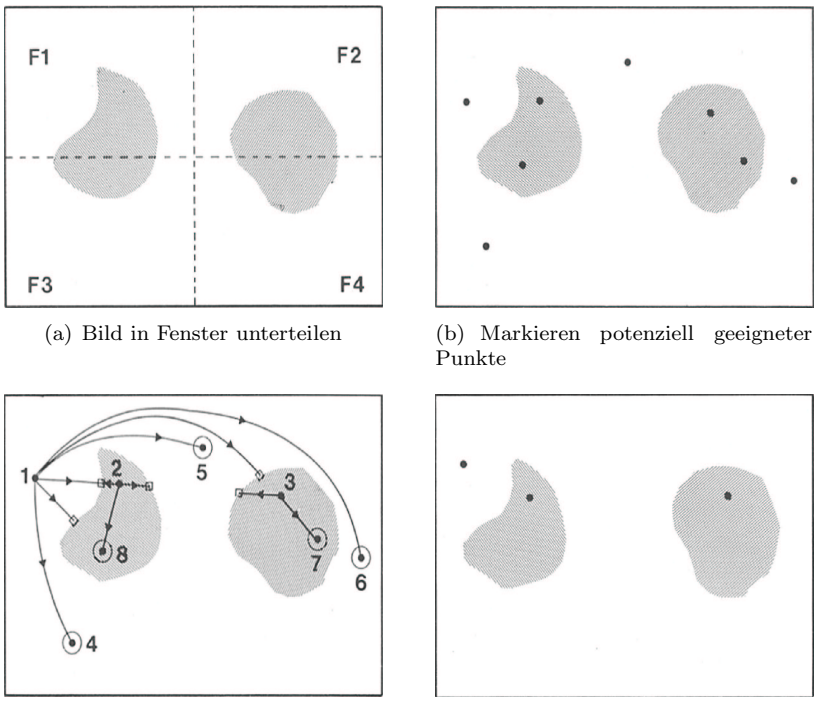
Snakes

Das Prinzip der Snake bezieht sich hautsächlich auf die Konturen und Kanten in einem Bild. In Abbildung 2.24 sind die Hell-Dunkel-Kontraste gut sichtbar.

Die Idee von Kass, Witkin und Terzopoulos [MKT87] war es, ein Konturverlauf in einem Bild mathematisch zu beschreiben. Sie postulierten, jede beliebige Kontur ließe sich grundsätzlich durch eine Snake beschreiben. Das Finden der Originalkontur erfolgt iterativ, das heißt, die Snake muss zu Beginn in der Nähe der gewünschten Kontur platziert werden. Die Snake soll dann Schritt für Schritt das Wunschobjekt auf dem Bild umschließen.

$$E(\mathbf{v}(s,t)) = S(\mathbf{v}(s,t)) + P(\mathbf{v}(s,t)) \tag{2.14}$$

Wie in Abbildung 2.25 ersichtlich ist, wurde die erste Snake in Form einer Ellipse um das zu erkennende Objekt gelegt. Aufgabe des iterativen Algorithmus ist es, mit Hilfe der „wirkenden Energien“ die konkrete Form des Objekts anzunehmen. Wie und in welcher Form sich die Snake an das Objekt nähern soll, hängt von verschiedenen Parametern ab. Parameter sind hier die



(a) Bild in Fenster unterteilen

(b) Markieren potenziell geeigneter Punkte

(c) Eliminieren überflüssiger Punkte

(d) Gefundene Initialpunkte

Abbildung 2.23: Verfahren zur optimalen Verteilung von Initialpunkten

Gewichtungen der einzelnen Energien im Bild und entlang der Snake. Die Energiefunktion (2.14) setzt sich aus der internen Energie S und der externen Energie P zusammen, wobei $\mathbf{v}(s,t)$ die Kurve der Snake zum Zeitpunkt t beschreibt.

Externe Energie

Die externe Energie wird durch folgende Formel mathematisch beschrieben:

$$P(\mathbf{v}) = \int_0^1 Q(\mathbf{v}(s))ds \tag{2.15}$$

Die externe Energie, welche auf die Snake wirkt, ist der Einfluss der Bildeigenschaften. Das heißt, dass die Helligkeit bzw. die Helligkeits-Änderung in jedem Punkt der Snake die Richtung angibt, in die sich der Snakepunkt zu bewegen hat. Der im Bild sich befindende Planet (vgl. Abbildung 2.26(a)) weist einen ziemlich starken Kontrast zum Weltall als Hintergrund auf. In Abbildung 2.26(b) ist nun ein Beispiel einer Snake mit hoher externer Energie. Mit dem Ziel, die Energie so minimal wie möglich zu machen, wird versucht die Snake an die Kanten des Objekts zu legen. In Abbildung 2.26(c) ist die Energie der Snake, gegenüber der vorhergehenden stark minimiert worden.

Die Schrittweite der Iteration kann mittels einer Konstante vorgegeben werden. Zu scharfe Kanten haben jedoch den Nachteil, dass bei zu großen Iterationsschritten das Objekt nicht erkannt wird. Mit Hilfe eines Tiefpasses beispielsweise, lässt sich dieser Effekt vermeiden.

Interne Energie

Die interne Energie wird durch folgende Formel mathematisch beschrieben:

$$S(\mathbf{v}) = \int_0^1 [\alpha(s)|\frac{\partial \mathbf{v}}{\partial s}|^2 + \beta(s)|\frac{\partial^2 \mathbf{v}}{\partial s^2}|^2]ds \tag{2.16}$$

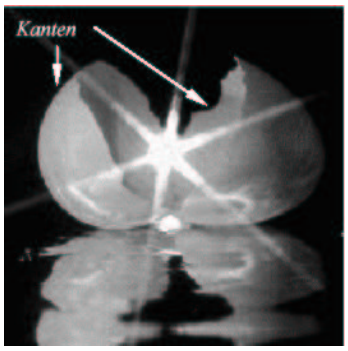


Abbildung 2.24: Kanten als Grauwert-Kontrast

Abbildung 2.25: Funktionsweise einer Snake

Die interne Energie (2.16) bestimmt die Form der Snake. Es wird im Weiteren zwischen Dehnungs- ($\alpha(s)$) und Biegungsenergie ($\beta(s)$) unterschieden.

Wie bereits erwähnt, wird am Anfang die Snake in Form eines Kreises oder einer Ellipse um das Objekt gelegt. Um die Dehnungsenergie (vgl. Abbildung 2.27(a)) zu reduzieren, wird sich die Snake zusammenziehen.
Die Biegungsenergie (vgl. Abbildung 2.27(b)) macht sich durch die Spitzen und Ecken der Snake bemerkbar. Diese sind sehr energieaufwändig. Das Ziel ist es, durch die Reduktion der Biegungsenergie die Form der Snake so „glatt“ wie möglich zu machen. Im Folgenden sind verschiedene Verhalten einer Snake aufgeführt.

Bei Abbildung 2.28(a) ist weder die interne noch die externe Energie optimal gewichtet. Es ist gut sichtbar, wie die Snake die Kontur-Änderung übersehen hat. Die Ecken weisen auf einen hohen inneren Energiegehalt hin.

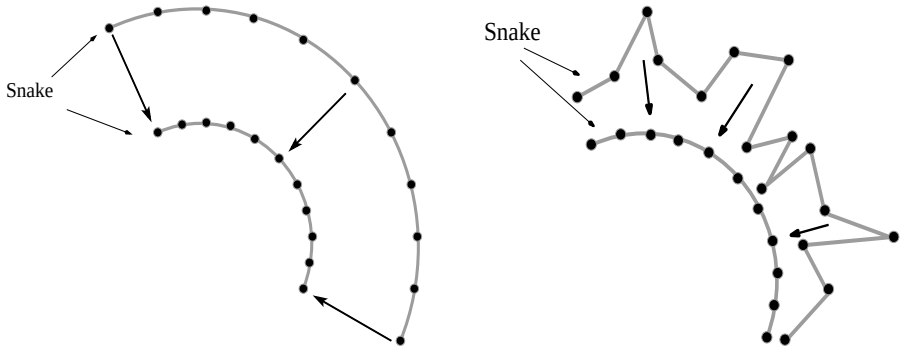
Bei Abbildung 2.28(b) stimmt die Gewichtung der externen Energie. Die Snake hat den Konturübergang offenbar gefunden. Doch wie im obigen Beispiel ist die innere Energie nicht minimal, was aus den Ecken zu entnehmen ist.

In Abbildung 2.28(c) sind die Ecken in der Snake verschwunden, was meist auf eine minimale innere Energie hinweist. Jedoch befindet sie sich keineswegs an der gewünschten Stelle. Die externe Energie ist offensichtlich falsch gewichtet.

Abbildung 2.28(d) zeigt eine optimal gewichtete Snake. Sie zeichnet sich durch die Lage auf der Stelle mit der höchsten Kontur-Änderung und ihrer „glatten“ Form aus.

(a) (b) (c)

Abbildung 2.26: Wirkungsweise der externen Energie



(a) Dehnungsenergie (b) Biegungsenergie

Abbildung 2.27: Interne Energie

Fixpunkte

Da sich die Reduktion der inneren Energie auf die Snake glättend auswirkt, stellen Bildobjekte mit scharfen Konturen und vielen Ecken Probleme dar. Das bedeutet, dass ausgeprägte Stellen des Objekts nicht erkannt werden.

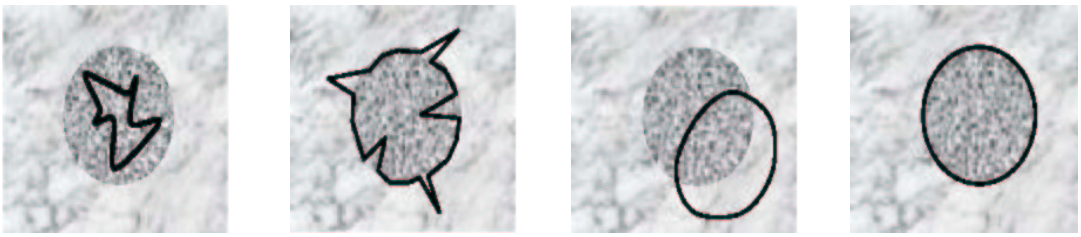
Die in Abbildung 2.29(a) sichtbare Luftseilbahngondel weist im Bereich der Tragrollen zwei markante Ecken auf. Die Snake ist mit den beschriebenen Mitteln nicht in der Lage, diese Stellen des Objekts zu erfassen. Dieser Fehler lässt sich durch das Markieren von Fixpunkten im Bild verhindern. Die Fixpunkte zwingen die Snake, obwohl sie nach minimaler Energie strebt, die Ausprägungen einzubeziehen, (vgl. Abbildung 2.29(b)) [Zau01].

2.6 Tracking-Algorithmen

Dieses Kapitel soll einen Überblick über das Tracking von (deformierbaren) Objekten in der Ebene (2D Bilder) liefern. Hierbei sind besonders die modellbasierten Ansätze interessant, die von aktiven Konturen bis hin zu statistischen Methoden gefächert sind. Die Frage nach der Auswahl robuster Features für das Tracking von Objekten und der Umgang mit Deformation wird erörtert.

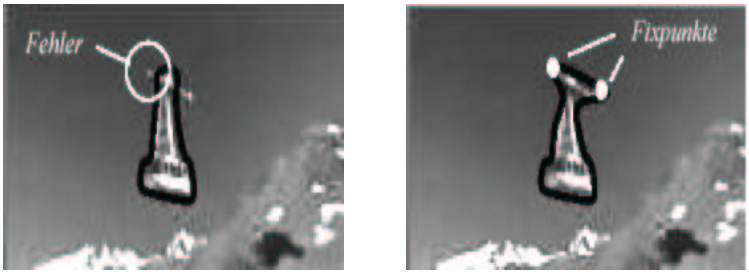
2.6.1 Definition Tracking

Beim Tracking geht es um die Verfolgung von bewegten Objekten mit dem Ziel der Extraktion von Informationen über den Verlauf der Bewegung und der Lage eines Objektes.
Diese Informationen können Geschwindigkeit oder die Richtung der Bewegung sein, sowie die Lage und Form des Objektes in einem späteren Bild.



(a) (b) (c) (d)

Abbildung 2.28: Wirkungsweise der Gewichtung der internen und externen Energie



(a) Fehler in der Snake (b) Fixpunkte gesetzt

Abbildung 2.29: Fixpunkte

2.6.2 Gesamtüberblick über Tracking Algorithmen

Die triviale Lösung des Tracking Problems ist eine Segmentierung von jedem einzelnen Frame. Eine Segmentierung nur auf Einzelbilder angewand ist allerdings in der Realität in den meisten Fällen schwierig und nicht besonders effizient. Zudem muss der Bereich, der in einem Einzelbild bzw. Frame segmentiert wurde, in einem nächsten Frame wiedergefunden werden. Normalerweise wird ein Differenzbild zwischen zwei Frames berechnet, so dass man eine Bewegung in zwei aufeinanderfolgenden Frames erkennen kann.
Es ist möglich, dass ein Objekt, unter Berücksichtigung der Form und seiner Deformation, verfolgt wird. Wenn die Kontourdeformation des Objektes berücksichtigt werden soll, dann ist es sinnvoll, eine kompakte Darstellung des Objektes zu benutzen, wie z.B. das Skelett des Objektes [LL93].
Es kann auch nur der Schwerpunkt des Objektes verfolgt werden. Dabei entsteht ein Weg über die Zeit, der durch jene Schwerpunkte bestimmt wird.
Die hier vorgestellten Methoden zum Tracken sind Snakes und das Blockvergleichverfahren. Beide können mit einem Kalman-Filter unterstützt werden, der Schätzungen des Systems vornimmt, wie im späteren Teil dieses Kapitels noch beschrieben wird.
Im Folgenden sollen ein paar wichtige Voraussetzungen für ein effizientes Tracking vorgestellt, und auf die sich ergebenden Probleme hingewiesen werden.

Vorraussetzungen

Für ein erfolgreiches Tracking, d.h. ein Tracking, das nach dem subjektiven Empfinden des Betrachters qualitativ gute Ergebnisse liefert, müssen bestimmte Annahmen getroffen werden:

- Zunächst wird eine geringe Objektbewegung von Frame zu Frame erwartet. Wenn die Objekte sich von einem auf den anderen Zeitpunkt an einen völlig anderen Ort begeben würden, ist es schwierig mit einem Algorithmus Zellen zu verfolgen, da die Zellen dann auf keinem nachvollziehbaren Weg gewandert sind.

- Es wird eine geringe Deformation des Objektes von Frame zu Frame angenommen. Man

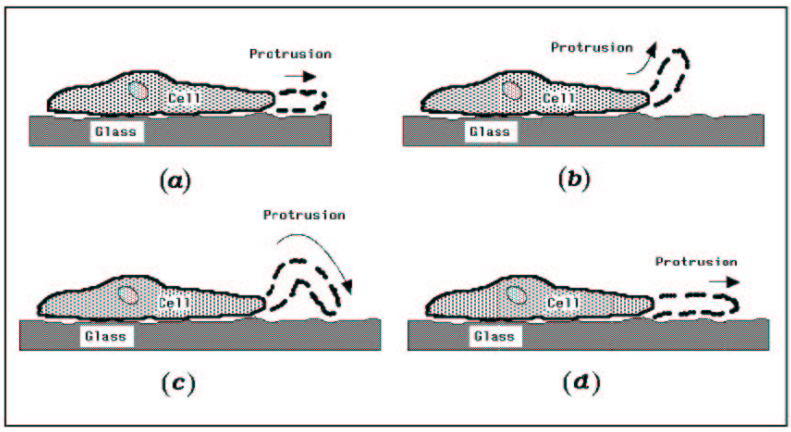


Abbildung 2.30: Die Zelle als 3D-Entität [LL93]

sollte also davon ausgehen können, dass sich z.B. das Objekt nicht von einem Frame auf einen anderen in seiner Größe verzehnfacht.

- Es ist nützlich a-priori Wissen (Vorwissen) über die zu trackenden Objekte zu besitzen, wie z.B. über das Aussehen der Zellen.
- A-priori Wissen über das Gesamtsystem erweist sich auch als überaus nützlich, wie z.B. über die Beleuchtungsintensität.

Allgemeine Probleme beim Celltracking

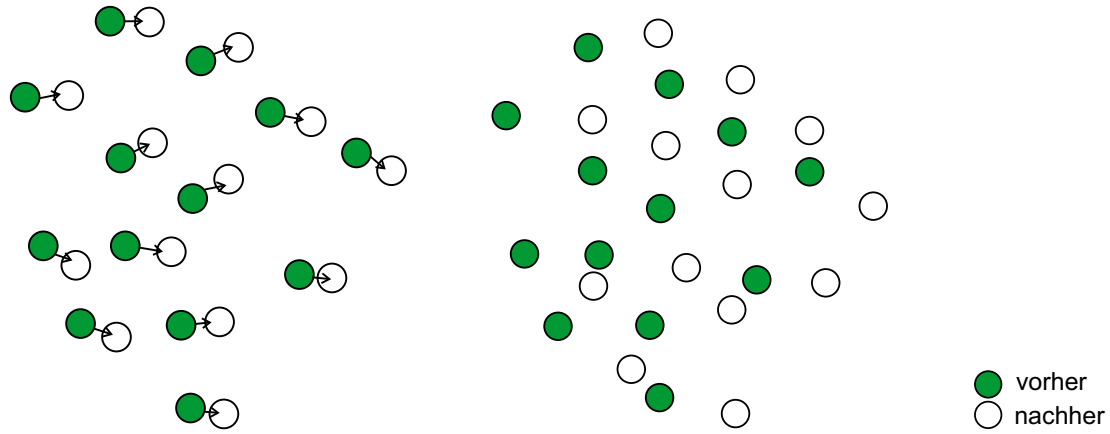
Es ergeben sich beim Tracken von Zellen spezielle Probleme:

- Ungleiche Beleuchtungen im Mikroskop können zu falschen Ergebnissen führen. Eventuell werden Zellen aufgrund von Dunkelheit nicht erkannt.
- Ungleiche Intensität der Zellmembranen führt zu falschen Segmentierungen.
- Die Zelle ist eine 3D-Entität und die Aufnahme wird nur in 2D durchgeführt. Daraus ergeben sich wiederum Fehlsegmentierungen. Das Problem ist in Abbildung 2.30 visualisiert. Man kann erkennen, dass die Zelle sich in unterschiedlicher Weise von dem Glas abheben kann und sich deshalb die Intensität unterscheidet.
- In der Aufnahme sind meistens Variationen und Rauschen enthalten.
- Zellen können kollidieren und dadurch nicht korrekt segmentiert und getrackt werden.
- Zellen können den betrachteten Bereich verlassen.

Allgemeine Probleme sind das Korrespondenzproblem und dessen spezieller Fall, das Blendenproblem, die im Folgenden beschrieben werden.

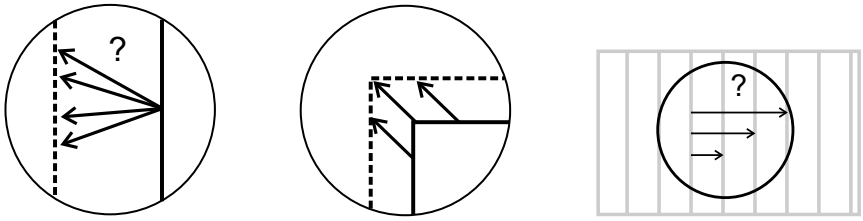
Korrespondenzproblem

Das Korrespondenzproblem bezieht sich auf die Zuordnung der Elemente in zwei aufeinander folgenden Bildern. In Abbildung 2.31 wird das Problem deutlich. Es soll zu jedem Objekt der Verschiebungsvektor (VV) bestimmt werden. Bei Abbildung 2.31(a) ist dies ohne weiteres möglich, da hier die Aufnahmefrequenz hoch genug gewählt wurde. Reduziert man die Aufnahmefrequenz, so erhält man Abbildung 2.31(b). Hier kann man die Verschiebungsvektoren nicht mehr eindeutig bestimmen, d.h. man weiß nicht welcher Punkt mit welchem korrespondiert.



(a) mittl. Teilchenabstand größer als VV (b) mittl. Teilchenabstand kleiner als VV (c) Legende

Abbildung 2.31: Korrespondenzproblem



(a) Bestimmung des VV nicht möglich (b) Bestimmung des VV an Ecken möglich (c) Weder Richtung noch Länge des VV können bestimmt werden

Abbildung 2.32: Blendenproblem

Blendenproblem

Das Blendenproblem ist ein Spezialfall des Korrespondenzproblems. Wenn man einen begrenzten Ausschnitt des Bildes betrachtet, kann man die Bewegung bestimmter Strukturen eventuell nicht eindeutig identifizieren. Zur Verdeutlichung dient Abbildung 2.32. Die durchgezogene Linie zeigt das Objekt im aktuellen Bild und die gestrichelte Linie zeigt das Objekt im Folgebild. Wenn nur eine senkrechte Kante, wie in Abbildung 2.32(a) zu sehen ist, dann kann die Richtung des Verschiebungsvektors nicht genau bestimmt werden. Die einzige Aussage, die man über die Bewegung der Kante machen kann, ist, dass sie sich nach links bewegt hat. Eine eventuelle zusätzliche vertikale Bewegung bleibt dem Betrachter verborgen. Ist allerdings eine Ecke eines Objekts zu sehen (vgl. Abbildung 2.32(b)), so ist der Verschiebungsvektor und damit die Bewegung eindeutig zu bestimmen. Bei periodischen Strukturen ist die Bestimmung des Verschiebungsvektors ein noch viel größeres Problem. In Abbildung 2.32(c) ist nicht nur die vertikale Bewegung nicht bestimmbar, sondern auch die Länge der horizontalen Bewegung ist nicht messbar. Eine Zuordnung zweier Punkte in zwei aufeinander folgender Bilder ist damit unmöglich.

2.6.3 Optischer Fluss

Jedes Pixel eines Bildes ist durch einen einzelnen Grauwert beschrieben, der die Intensität dieses Pixels wiedergibt. Dieser Grauwert kann sich aus vielen Faktoren zusammen setzen, wie etwa, der

Farbintensität, der Position der Lichtquellen oder der Orientierung, dem Reflexionsgrad und der Transparenz des Objekts [Cam94]. Diese Faktoren können sich über einen Zeitraum verändern, wenn sich das Objekt beispielsweise bewegt. Die Verschiebung des Pixels kann durch einen Verschiebungsvektor angegeben werden und durch die Optische Fluss Gleichung:

$$F(x, y, t + 1) = F(x + \Delta x(x, y), y + \Delta y(x, y), t)$$

$F(x, y, t + 1)$ entspricht dem Grauwert an der Position (x, y) zum Zeitpunkt $t + 1$. Der rechte Teil der Gleichung beschreibt die Verschiebung, die das Pixel seit dem Zeitpunkt t erfahren hat. Voraussetzung für die Angabe einer solchen Verschiebung ist jedoch eine konstante Beleuchtung, da sich bei einer Änderung der Beleuchtung auch die Grauwerte der Pixel ändern und somit eine Zuordnung in einem späteren Bild unmöglich werden kann. Der Optische Fluss beschreibt die Lösung der Optischen Fluss Gleichung für alle Pixel des Bildes, dass heißt er entspricht dem Vektorfeld aus den Verschiebungsvektoren $(\Delta x, \Delta y)$. Diese Verschiebungsvektoren zu finden ist jedoch nicht nur bei schwankender Beleuchtung schwierig, sondern kann auch bei konstanter Beleuchtung zu Problemen führen. Man kann dazu beispielsweise ein Bild mit 512×512-Pixeln und 8 Bit Grauwerttiefe betrachten. In diesem Bild haben durchschnittlich 1024 Pixel denselben Grauwert und eine genaue Zuordnung zweier Pixel in zwei Bildern kann dadurch sehr schwierig sein. Darüber hinaus müssen noch zwei weitere Einschränkungen an den Optischen Fluss gestellt werden. Zum einen soll die Glattheit des Flusses möglichst groß sein und zum anderen sollen die Verschiebungsvektoren möglichst klein sein. Die Einschränkungen sollen beim Lösen der Fluss Gleichung helfen, weil durch sie die Entwicklung des Optischen Flusses in der Taylor-Reihe ermöglicht wird [Cam94]. Im Allgemeinen ist der Optische Fluss nicht gleich der 2D-Projektion des 3D-Bewegungsfeldes des Objekts. Dies kann man sich an einer glatten, rotierenden Kugel vorstellen. Sie erzeugt keinen Optischen Fluss, obwohl die 2D-Projektion nicht Null ist. Andererseits würde eine Lichtquelle, die sich bewegt auf einer stationären Kugel einen Optischen Fluss erzeugen, obwohl die 2D-Projektion überall Null ist. Ist die Kugel jedoch texturiert oder besitzt sie keine vollkommen glatte Oberfläche, so approximiert der optische Fluss die 2D-Projektion der Kugel beliebig gut.

2.6.4 Tracking mit Snakes

Bei der Lösung des Trackingproblems mit dem Snakemodell wird die Zelle segmentiert und verfolgt. Wir beziehen uns in diesem Abschnitt komplett auf die Arbeit von F. Leymarie und M. Levine [LL93] und das Kapitel über „Segmentierung von Bilddaten“ in diesem Bericht (vgl. Kapitel 2.5).

Einsatz der Snake zum Tracken von Zellen

Mit dem Snakemodell ist ein Tracken von Objekten (Zellen) möglich. Eine Bildbewegungssequenz ist normalerweise eine Liste von Frames $(f_1, f_2, f_3, ..., f_N)$. Es kann weiterhin die Annahme gemacht werden, dass sich das Objekt (Zelle) nur wenig von Frame zu Frame bewegt und infolgedessen beispielsweise nur jedes 10te Frame betrachtet werden muss. Das Tracken funktioniert dann folgendermaßen:

- Im ersten Frame wird die Snake optimal positioniert. Dieser Schritt kann interaktiv, automatisch oder semi-automatisch durchgeführt werden. Das Funktional wird nun minimiert.
- In den folgenden Frames wird die Information des vorhergehenden Frames dahingehend genutzt, dass die Anfangsposition der Snake in dem aktuellen Frame auf die Endposition der Snake im vorhergehenden Frame gesetzt wird. So benötigt der Minimierungsprozess wesentlich weniger Schritte, als wenn die Snake komplett neu in das Bild hineingesetzt würde.

Eine weitere Einsparung ergibt sich dadurch, dass man nur Teilbereiche des Gesamtbildes betrachtet in denen sich die Snake bewegt (z.B. Orte an denen die Zellen liegen). Die Abbildung 2.33 zeigt das Ergebnis des Tracking mit der Snake.

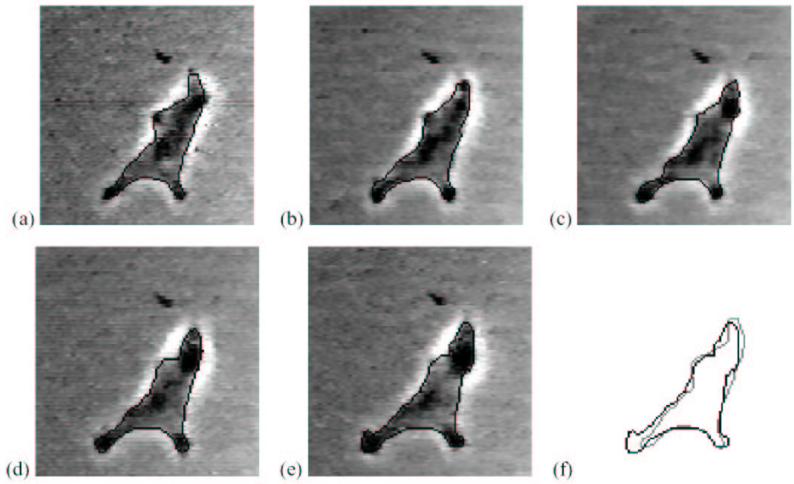


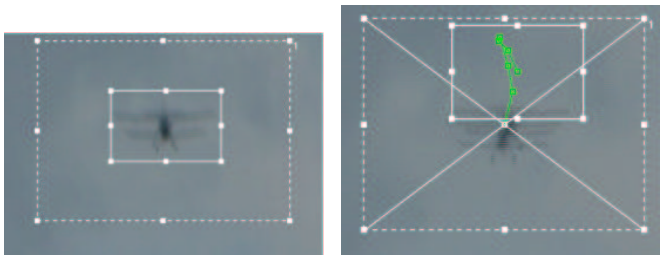
Abbildung 2.33: Ergebnis des Snaketrackings [LL93]

2.6.5 Blockvergleichverfahren

Das Blockvergleichverfahren ist ein gängiges und einfaches Verfahren. Es wird vor allem in der Videobearbeitung zum Stabilisieren eines verwackelten Videos eingesetzt. Das Verfahren arbeitet nach folgenden Schritten:

1. Wähle einen Block im ersten Frame, der das zu trackende Objekt enthält.
2. Wähle einen zweiten umgebenden Block, der die maximale Frame zu Frame Bewegung eingrenzt (dient zur Eingrenzung des Suchbereichs).
3. Extrahiere Merkmale aus dem inneren Block (z.B. Grauwertverteilung als Merkmal).
4. Bestimme im nächsten Frame die Position des inneren Blockes. Diese Positionsbestimmung geschieht durch eine Suche nach der größten Übereinstimmung. Es gibt verschiedene Suchstrategien wie:
   - Vollständige Suche: Hier wird ein Änlichkeitsmaß (Abstandsmaß der Merkmale) für jede mögliche Verschiebung innerhalb eines Suchbereichs bestimmt. Der Vorteil dieser Suchstrategie ist, dass das globale Optimum gefunden wird. Der Nachteil jedoch ist der sehr große Aufwand, der quadratisch mit der Größe des Suchbereichs wächst.
   - 2D-Logarithmische Suche: Bei jedem Suchschritt wird das Ähnlichkeitsmaß an fünf Positionen bestimmt. Entsprechend bestimmter Regeln wird der Suchbereich verschoben und die Suchdistanz halbiert. Dies wird solange durchgeführt bis das Ähnlichkeitsmaß ausreichend gut ist.
   - Weitere Suchverfahren, die vor allem auf das Problem angepasst wurden. Man könnte z.B. an der Stelle mit der Suche beginnen, an der ein Prediktor die neue Position des Objektes vermutet. Hier sei wieder auf den Kalman-Filter hingewiesen.
5. Verschiebe die Bildausschnitte entsprechend der Bewegung des Objektes (wenn es einen weitereren Frame gibt, gehe zu 3., ansonsten beende das Verfahren).

Das Ergebnis von dem Algorithmus zeigt sich in Abbildung 2.34, wobei Abbildung 2.34(a) die Ausgangsposition zeigt und in Abbildung 2.34(b) der Pfad eingetragen ist, den das Objekt (Flugzeug) zurückgelegt hat. Der Pfad besteht nicht aus einer Menge von Rechtecken, sondern nur aus den Mittelpunkten der Rechtecke in den beobachteten Frames.



(a) Ausgangsposition (b) Bild nach der Bewegung des Flugzeugs

Abbildung 2.34: Blockvergleichverfahren

2.6.6 Kalman-Filter

Der Kalman-Filter ist nach Rudolph E. Kalman benannt, der 1960 in einer Veröffentlichung eine rekursive Lösung des linearen Filterns von diskreten Daten beschrieb. Der Filter besteht aus grundlegenden mathematischen Gleichungen, die einen Predictor-Corrector Schätzer darstellen. Dieser soll die geschätzte Fehlerkovarianz minimieren, falls bestimmte Bedingungen erfüllt werden [WB01]. Die Bedingungen ergeben sich aus der Modellierung des zu schätzenden Systems. Beispielsweise muss eine Annahme darüber getroffen werden, in wie weit zwei aufeinander folgende Messungen miteinander korellieren und wie groß das Mess- und das Systemrauschen sind. Dieses Rauschen ist auch das größte Problem bei der Schätzung des Zustandes, in dem sich das System gerade befindet. Würde das Rauschen nicht existieren, könnten die Messungen direkt als Zustand verwendet werden.
Der Kalman-Filter unterteilt sich grob in zwei Schätzungen: eine a-priori Schätzung und eine a-posteriori Schätzung [WB01]. Die a-priori Schätzung ist eine Schätzung des aktuellen Zustandes bevor eine verrauschte Messung des aktuellen Zustandes vorgenommen wird. Diese Schätzung beruht nur auf der Schätzung des vorhergehenden Zustandes und der Fehlerkovarianz. Die a-posteriori Schätzung ist analog, eine Schätzung nach der Messung des aktuellen Zustandes. Sie setzt sich aus der a-priori Schätzung und der aktuellen Messung zusammen, wobei, je nach Abweichung, die a-priori Schätzung oder der aktuelle Messwert mehr Gewicht bei der a-posteriori Schätzung erhält. Die Rekursion des Kalman-Filters besteht nun darin, dass diese beiden Schätzungen nacheinander rekursiv aufgerufen werden und so die Fehlerkovarianz nach und nach minimiert werden soll.

Kalman-Rekursion

Die Kalmanrekursion besteht aus zwei Schritten: Dem Time-Update (Predict) und dem Measurement-Update (Correct), wie Abbildung 2.35 zeigt. Diese beiden Schritte werden rekursiv aufgerufen, dass heißt das Time-Update liefert Daten für das Measurement-Update und diese liefert wiederum Daten für das Time-Update des nächsten Zeitschrittes. Das Time-Update soll eine a-priori Schätzung des aktuellen Systemzustandes berechnen, während das Measurement-Update die a-priori Schätzung korrigiert, indem der aktuelle Messwert in die Schätzung mit einbezogen wird. Bevor die Rekursion jedoch gestartet werden kann, müssen die Werte für die a-posteriori Schätzung des Zustandes $\hat{\mathbf{x}}_0$ und die a-posteriori Schätzung der Fehlerkovarianz $\mathbf{P}_0$ initialisiert werden, da diese Werte für das erste Time-Update gebraucht werden. Die beiden Werte ergeben sich dabei aus der Modellannahme des zu schätzenden Systems.

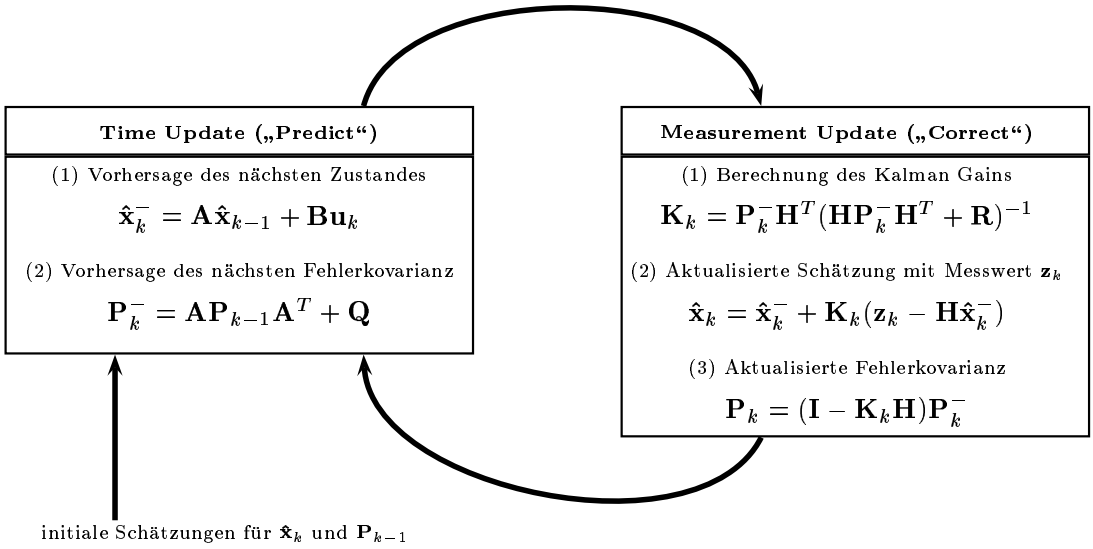


Abbildung 2.35: Die Schritte der Kalmanrekursion [WB01]

Time-Update (Predict)

Zuerst wird eine Vorhersage des aktuellen Zustandes berechnet. Sie setzt sich zusammen aus einer Linearkombination der a-posteriori Schätzung des vorhergehenden Zustandes und einer Kontrolleingabe des aktuellen Zustandes:

$$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k. \tag{2.17}$$

Die Variable $\hat{\mathbf{x}}_k^-$ beschreibt die a-priori Schätzung des aktuellen Zustandes. $\mathbf{A}$ ist eine $n \times n$-Matrix, die die Beziehung zwischen zwei Zuständen beschreibt, und $\mathbf{B}$ eine $n \times l$-Matrix, die einen Zustand mit einer Kontrolleingabe in Beziehung setzt. Schließlich ist $\mathbf{u}_k$ eine Kontrolleingabe, die zum Zeitpunkt k mit in die a-priori Schätzung des aktuellen Zustandes mit einfließen kann. Sie ist allerdings nicht zwingend notwendig. Eine solche Kontrolleingabe könnte beispielsweise das Drücken einer Taste an einem Automaten sein, dessen nächster Zustand von dieser Eingabe abhängt.
Nach der a-priori Schätzung des Zustandes muss die a-priori Schätzung der Fehlerkovarianz für den aktuellen Zeitschritt berechnet werden. Die a-priori Schätzung der aktuellen Fehlerkovarianz setzt sich dabei aus der a-posteriori Schätzung der Fehlerkovarianz des vorhergehenden Zeitpunktes und einer Annahme über das Systemrauschen zusammen.

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}. \tag{2.18}$$

Die Variable $\mathbf{P}_k^-$ beschreibt die a-priori Schätzung der aktuellen Fehlerkovarianz und $\mathbf{P}_{k-1}$ die a-posteriori Schätzung der Fehlerkovarianz des vorhergehenden Zeitpunkts. Die a-priori Fehlerkovarianz ist dabei definiert als:

$$\mathbf{P}_k^- = E[\mathbf{e}_k^-\mathbf{e}_k^{-T}], \text{ mit} \tag{2.19}$$
$$\mathbf{e}_k^- = \mathbf{x}_k - \hat{\mathbf{x}}_k^-. \tag{2.20}$$

Hierbei ist $\mathbf{x}_k$ der Messwert zum Zeitpunkt k und $\mathbf{x}_k^-$ der Erwartungswert des Messwertes zum Zeitpunkt k.
Darüber hinaus ist $\mathbf{Q}$ eine Kovarianzmatrix, die das Systemrauschen simulieren soll. Sie stellt nicht den wahren Wert des Systemrauschens dar, sondern ist eine Beschreibung des Rauschens aus der

getroffenen Modellannahme.
Nach dem Time-Update stehen also die Werte für die a-priori Schätzungen des Zustandes und der Fehlerkovarianz fest und können im Measurement-Update verwendet werden.

Measurement-Update (Correct)

Beim Measurement-Update werden nun alle a-posteriori Werte berechnet, dass heißt es fließen auch die Werte des aktuellen Zeitschritts in die Berechnungen mit ein. Zunächst wird der Gainfaktor berechnet. Er ist später für die Gewichtung der a-priori Schätzung des Zustandes und der Differenz aus a-priori Schätzung und Messwert verantwortlich. Anders ausgedrückt soll der Gainfaktor die Fehlerkovarianz minimieren und so die Schätzungen des Zustandes verbessern. Die Berechnung des Gainfaktors geschieht folgendermaßen:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T (\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R})^{-1}. \tag{2.21}$$

Die $m \times n$-Matrix $\mathbf{H}$ beschreibt dabei die Beziehung zwischen einem Zustand und einem Messwert und $\mathbf{R}$ ist die Kovarianzmatrix des Messrauschens. Auch diese Kovarianzmatrix stammt nicht aus dem System selbst, sondern aus einer Modellannahme für das System, dass heißt sie entspricht nicht dem wahren Wert, sondern nur einer Schätzung. Sieht man sich die Formel für den Gainfaktor an, kann man folgendes feststellen:

$$\lim_{\mathbf{R}_k \to 0} \mathbf{K}_k = \mathbf{H}^{-1}, \tag{2.22}$$

$$\lim_{\mathbf{P}_k^- \to 0} \mathbf{K}_k = 0. \tag{2.23}$$

Anschaulich bedeutet das, wenn das Messrauschen $\mathbf{R}_k$ zum Zeitpunkt k klein ist, dann ist der Gainfaktor $\mathbf{K}$ groß und die a-priori Schätzung des Zustandes erhält weniger Gewicht bei der a-posteriori Schätzung des Zustandes. Dies ist sinnvoll, da der aktuell gemessene Wert eine gute Approximation des wahren Wertes darstellt. Wenn aber die a-priori Schätzung der Fehlerkovarianaz gut war ($\mathbf{P}_k^- \to 0$), dann ist der Gainfaktor klein und die a-priori Schätzung des Zustandes ist hinreichend gut für die a-posteriori Schätzung des Zustandes.
Nachdem der Gainfaktor berechnet wurde, kann mit der a-posteriori Schätzung des Zustandes fortgefahren werden. Sie setzt sich neben der a-priori Schätzung des aktuellen Zustandes auch aus der gewichteten Differenz aus dem aktuellen Messwert und der a-priori Schätzung für diesen Messwert zusammen. Das Gewicht liefert der Gainfaktor mit den oben beschriebenen Eigenschaften:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-). \tag{2.24}$$

Die Variable $\mathbf{z}_k$ ist der gemessene Zustand zum Zeitpunkt k. Der Wert dieser Variable kann durch folgende Gleichung beschrieben werden [WB01]:

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k. \tag{2.25}$$

Die $m \times n$-Matrix $\mathbf{H}$ setzt dabei wieder den Zustand und die Messung in Beziehung. Das Messrauschen, das jede Messung überlagert wird durch die Variable $\mathbf{v}_k$ beschrieben. Sie ist keine Schätzung, sondern der reale Wert des vorhandenen Messrauschens. Auch die Variable $\mathbf{x}_k$ ist in diesem Fall keine Schätzung, sondern der reale Referenzwert, der den Zustand des Systems beschreibt. Die Gleichung verdeutlicht, dass eine Bestimmung des Referenzwertes nicht möglich ist, solange der exakte Wert des Rauschens nicht bekannt ist. Der Referenzwert kann darüber hinaus durch die folgende Gleichung beschrieben werden, welche die Annahme eines typischen Prozessmodells macht [WB01]:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k. \tag{2.26}$$

Die Matrix $\mathbf{B}$ beschreibt wieder, wie ein Zustand des Systems und eine optionale Kontrolleingabe in Beziehung stehen. Das Systemrauschen wird in der Gleichung durch die Variable $\mathbf{w}_k$ beschrieben,

die auch keine Schätzung ist, sondern den realen Wert darstellt.
Nachdem beim Measurement-Update die a-posteriori Schätzung des aktuellen Zustandes erfolgt ist, muss noch die a-posteriori Fehlerkovarianz berechnet werden. Dies geschieht durch die Gleichung, wobei $\mathbf{I}$ die Einheitsmatrix und $\mathbf{K}_k$ der Gainfaktor zum Zeitpunkt k ist:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^- . \tag{2.27}$$

Mit dem Abschluss des Measurement-Updates endet auch ein Zeitschritt. Es stehen also der Wert für den Gainfaktor und die Werte der a-posteriori Schätzungen des Zustandes und der Fehlerkovarianz fest und können im Time-Update des nächsten Zeitschritts verwendet werden, wodurch sich die Kalman-Rekursion fortsetzt.

Anwendbarkeit auf das Tracking

Der Kalman-Filter kann dazu verwendet werden Tracking-Verfahren zu unterstützen. Voraussetzung für den Einsatz des Filters ist ein bereits bestehendes Tracking-Verfahren, zum Beispiel ein Blockvergleichsverfahren, das dem Kalman-Filter Messwerte liefert, die für das Measurement-Update benötigt werden. Diese Messwerte könnten beispielweise die Position des zu trackenden Objekts oder dessen Geschwindigkeit sein. Der Kalman-Filter korrigiert dabei fehlerhafte oder stark verrauschte Messungen und verbessert somit die Ergebnisse des Trackings. Darüber hinaus können mit dem Kalman-Filter Probleme, die beim Tracking auftreten, leichter gelöst werden. Wenn sich beispielsweise zwei zu trackende Objekte, die sich stark ähneln, aufeinander zu bewegen, sich überlagern und danach wieder weiter bewegen, ist es für viele Tracking-Verfahren schwer, nach der Überlagerung die Objekte wiederzufinden. Mit der Modellannahme, dass sich die beiden Objekte ungestört überlagern und gleichmässig weiter bewegen, kann man mit dem Kalman-Filter die beiden Objekte jedoch weiterverfolgen. Denn der Filter würde, durch den Gainfaktor und die Fehlerkovarianzen die fehlerhaften Bestimmungen des Tracking-Verfahrens korrigieren, so dass das Verfahren nach der Überlagerung wieder greift.

2.7 Statistische Datenanalyse

2.7.1 Einleitung

Dieses Kapitel thematisiert zunächst den allgemeinen Umgang mit Daten und Messreihen und nimmt dann Bezug auf den CellTrack-Kontext. Es stellt sich die Frage, mit welchen mathematischen Hilfsmitteln Messreihen, die aus Migrationsversuchen gewonnen werden, untersucht werden können, um daraus Aussagen über Zellbewegungen abzuleiten. Geeignet sind dafür Signifikanztests, deren Anwendung allerdings ein gewisses Basiswissen in den Bereichen Wahrscheinlichkeitstheorie und Statistik voraussetzt. Dieses Grundlagenwissen soll in Kapitel 2.7.3 vermittelt werden, woraufhin in Kapitel 2.7.4 dann ausgewählte Signifikanztests vorgestellt werden. Abschließend erfolgt eine Betrachtung des bisher von den Biologen angewendeten Verfahrens zur Bewegungsanalyse.

2.7.2 Die Auswertung von Messreihen

Der Umgang mit Daten

Unter „Messen“ versteht man die Feststellung der Merkmalsausprägung an einem konkreten Objekt. Beispielsweise könnte man bei einem Studenten die Körpergröße messen. Dann wäre der Student das Objekt, seine Körpergröße das Merkmal und der konkrete Wert – z.B. 1,86 Meter – die Ausprägung dieses Merkmals. Untersucht man auf diese Weise nun mehrere Objekte hinsichtlich desselben Merkmals, so spricht man von einer *Stichprobe* (Auswahl einer Menge von Objekten aus einer Grundgesamtheit). Die Tabellierung der Messwerte heißt *Messreihe*.

Merkmale können nach ihrer Art in vier verschiedene Klassen eingeteilt werden. Jeder Klasse liegt eine so genannte *Skala* zu Grunde, wobei folgende Skalen unterschieden werden:

Nominalskala: Bei nominalskalierten Merkmalen können Merkmalsausprägungen gleich oder verschieden sein. Eine weitere Differenzierung kann nicht vorgenommen werden, da die Ausprägungen lediglich Bezeichnungen darstellen. Ein Beispiel ist das Merkmal „Haarfarbe“ mit den möglichen Ausprägungen „blond“, „braun“, „schwarz“, „rot“,...

Ordinalskala: Auf den Ausprägungen ordinalskalierter Merkmale ist eine Ordnung definiert - sie lassen sich in eine Rangreihe bringen. Das Merkmal „Schulnote“ ist ordinalskaliert, da z.B. die Ausprägungen „2“ „schlechter“ ist als die Ausprägung „1“.

Intervallskala: Bei intervallskalierten Merkmalen lassen sich nicht nur „größer“- und „kleiner“-Beziehungen angeben, sondern die Differenz zweier Ausprägungen hat ebenfalls eine Bedeutung: Die Ungleichheit zweier Merkmalsausprägungen ist quantifizierbar. Untersucht man das Merkmal „Umgebungstemperatur“, so ist die Differenz der Ausprägungen „20°C“ und „25°C“ gleich der Differenz zwischen „26°C“ und „31°C“. Im Gegensatz dazu wäre eine Aussage der Form „Zwei Schüler mit den Noten 1 und 3 stehen in Bezug auf die Leistung im selben Verhältnis wie zwei Schüler mit einer 4 und einer 6.“ nicht sinnvoll.

Verhältnisskala: Verhältnisskalierte Merkmale besitzen schließlich darüber hinaus einen objektiv begründbaren Nullpunkt. Bestes Beispiel für ein verhältnisskaliertes Merkmal ist die Temperatur auf einer Kelvin-Skala: 0K entspricht dem absoluten Nullpunkt, an dem keinerlei Teilchenbwegung mehr stattfindet. Daher sind z.B. „40K“ „doppelt so warm“ wie „20K“. Auf einer Celsius-Skala wäre die Aussage „40°C ist doppelt so warm wie 20°C“ nicht korrekt, da als Nullpunkt willkürlich die Schmelztemperatur von Eis festgelegt wurde.
Von allen Arten von Merkmalswerten beinhalten verhältnisskalierte Merkmale die meisten Informationen.

Messwerte können auch als *Ergebnisse von Zufallsexperimenten* betrachtet werden, da sie lediglich Näherungswerte für wahre (aber unbekannte) Werte darstellen. Diese Tatsache ist durch unweigerlich auftretende Messfehler begründet: Störeinflüsse auf der einen Seite und Ungenauigkeiten beim Ablesen auf der anderen Seite ergeben mehr oder weniger starke Abweichungen von den eigentlich zu messenden Werten. Das Ausmaß dieser Messfehler kann reduziert werden, indem man eine Messung mehrfach durchführt und im Folgenden den Mittelwert der gemessenen Ausprägungen verwendet. Vollständig eliminieren lässt sich der Einfluss des Zufalls jedoch nicht, so dass sich ein großer Teil dieser Ausarbeitung mit statistischen Verfahren zur Einschätzung des „Ausmaßes des Zufalls“ beschäftigen wird (vgl. Kapitel 2.7.4).

Um eine Messreihe sinnvoll (statistisch) auswerten zu können, bietet sich in aller Regel eine Speicherung der Messdaten in tabellarischer Form an. Beispielsweise kann so dargestellt werden, welche Ausprägung ein untersuchtes Merkmal zu einem bestimmten Zeitpunkt hat oder bei wie vielen Objekten das Merkmal die Ausprägung x besitzt. Sollen Daten darüber hinaus auch visuell dargestellt werden, ist in vielen Fällen eine *Klassierung* erforderlich. Dazu ein Beispiel: Angenommen, bei 100 Studenten wurde die Körpergröße in Zentimetern ermittelt, und das Ergebnis soll nun in Form eines Balkendiagramms dargestellt werden. Das Merkmal „Körpergröße“ besitze bei dieser Stichprobe Ausprägungen im Intervall $[165, 204]$. Unter diesen Annahmen bestünde das Diagramm im schlimmsten Fall aus 40 Balken von jeweils nur geringer Höhe. Klassierung bedeutet nun, dass Teilintervalle der Merkmalsausprägung zusammengefasst werden. So würde man z.B. für jedes 5-cm-Intervall einen Balken verwenden, wodurch die visuelle Darstellung übersichtlicher und aussagekräftiger wird. Weitere Informationen zum Umgang mit Messdaten sind z.B. in [Pei85] zu finden.

2.7.3 Grundbegriffe der statistischen Datenanalyse

In diesem Kapitel sollen einige allgemeine Begriffe aus den Bereichen Wahrscheinlichkeitstheorie und Statistik eingeführt werden. Diese Begriffe bilden die Grundlage für eine formale Betrachtung von „Wahrscheinlichkeit“ und finden in den nachfolgenden Kapiteln Anwendung. Weiterführende

Informationen zu Wahrscheinlichkeitsrechnung und Statistik finden sich z.B. in [Kre02], [LW85] und [Käh95].

Wahrscheinlichkeitsräume

Angenommen, man würfelt mit zwei gleichmäßigen Würfeln. Als Ergebnis erhält man dann zwei Augenzahlen zwischen eins und sechs – also gibt es 36 Möglichkeiten. Mit welcher Wahrscheinlichkeit hat man einen Pasch (zwei gleiche Augenzahlen) gewürfelt? Da es sechs verschiedene Pärche gibt, würfelt man in 6 von 36 Fällen einen Pasch, die Wahrscheinlichkeit beträgt folglich $\frac{6}{36} = \frac{1}{6}$.

Nun sollen diese Beobachtungen formal ausgedrückt werden:
Man hat mit dem Würfeln ein *Zufallsexperiment* durchgeführt, dem eine so genannte *Ergebnismenge* Ω zu Grunde liegt. Diese Ergebnismenge umfasst alle möglichen Ausgänge eines Zufallsexperiments, also in diesem Fall die Paare $(1,1),(1,2),...,(6,5),(6,6)$. Oder kurz: $\Omega = \{1,2,3,4,5,6\}^2$.
Das Würfeln eines Paschs entspricht einer Menge von Ergebnissen $(\{(1,1),(2,2),...,(6,6)\})$ – man spricht dabei auch von einem *Ereignis*. Sei $\mathcal{A} = \wp\Omega$, dann umfasst $\mathcal{A}$ alle möglichen Ereignisse, die bei diesem Würfelexperiment eintreten können. $\mathcal{A}$ ist eine *σ-Algebra*:

Definition 2.7.1 (σ-Algebra) *Sei Ω eine nichtleere Menge und $\mathcal{A}$ ein System von Teilmengen von Ω. $\mathcal{A}$ heißt σ-Algebra über Ω, falls gilt:*

1. *$\Omega \in \mathcal{A}$;*
2. *für alle $A \in \mathcal{A}$ ist auch $A^c = \Omega \backslash A \in \mathcal{A}$;*
3. *aus $A_i \in \mathcal{A}$ $(i \in I \subset \mathbb{N})$ folgt $\bigcup_{i\in I} A_i \in \mathcal{A}$.*

Um jedem Ereignis eine Wahrscheinlichkeit zuzuordnen (dass dieses Ereignis eintritt), wird schließlich noch ein *Wahrscheinlichkeitsmaß* benötigt:

Definition 2.7.2 (Wahrscheinlichkeitsmaß) *Sei Ω eine Ergebnismenge, $\mathcal{A}$ eine σ-Algebra über Ω und $P : \mathcal{A} \to [0,1]$ eine Abbildung. P heißt Wahrscheinlichkeitsmaß, wenn gilt:*

1. *$P(A) \geq 0$ für alle $A \in \mathcal{A}$;*
2. *$P(\Omega) = 1$;*
3. *$P(A \cup B) = P(A) + P(B)$ für alle disjunkten $A, B \in \mathcal{A}$.*

Für das Würfel-Beispiel ist allgemein $P(\{(a,b)\}) = \frac{1}{36}$, also $P(\{(1,1),(2,2),...,(6,6)\}) = \frac{1}{6}$.

Das Tripel $(\Omega, \mathcal{A}, P)$ heißt *Wahrscheinlichkeitsraum*.

Zufallsvariablen

Ergebnisse von Zufallsexperimenten sind oftmals Zahlenwerte oder können zumindest als solche interpretiert werden. Untersucht man zum Beispiel das Merkmal „Haarfarbe“, so könnten die möglichen Ausprägungen „blond“, „braun“, „schwarz“,... auf die Zahlen 1, 2, 3... abgebildet werden.

Formal erfolgt eine solche Abbildung durch eine *Zufallsvariable*. Wenn beispielsweise X die Augensumme der beiden Würfel in dem Würfel-Experiment bezeichnet, so gilt:

$$X : \Omega \to \{2,3,...,11,12\}$$
$$X((1,1)) = 2, X((1,2)) = 3, ..., X((6,5)) = 11, X((6,6)) = 12$$

Der Abbildungscharakter einer Zufallsvariablen steht bei der konkreten Verwendung oft im Hintergrund. Notationen der Form $X = a$ oder $X \leq b$ bezeichnen die Ergebnisse eines Zufallsexperiments, dessen zugeordneter Zahlenwert $= a$ bzw. $\leq b$ ist.

Grundsätzlich wird zwischen *diskreten* und *kontinuierlichen Zufallsvariablen* unterschieden. Diskrete Zufallsvariablen haben einen endlichen oder abzählbar unendlichen Wertebereich (z.B. $\mathbb{N}$ oder die Menge $\{0, 1, 2, 3\}$), wohingegen kontinuierliche Zufallsvariablen überabzählbar unendlich viele verschiedene Werte annehmen können (z.B. Werte aus $\mathbb{R}$).

Verteilungsfunktion und Dichte

Zufallsvariablen werden (explizit oder implizit) über Wahrscheinlichkeitsräumen definiert, sind also mit einem Wahrscheinlichkeitsmaß assoziiert. *Verteilungsfunktion* und *Dichte* einer Zufallsvariablen geben an, wie wahrscheinlich bestimmte Ausprägungen sind.

Definition 2.7.3 (Verteilungsfunktion) *Sei X eine Zufallsvariable über dem Wahrscheinlichkeitsraum $(\Omega, \mathcal{A}, P)$. Dann ist die Verteilungsfunktion von X definiert durch*

$$F(x) = P(X \leq x) \ , \quad x \in \mathbb{R}. \tag{2.28}$$

Definition 2.7.4 (Dichte) *Sei X eine kontinuierliche Zufallsvariable über dem Wahrscheinlichkeitsraum $(\Omega, \mathcal{A}, P)$. Dann kann die Verteilungsfunktion auch in der Form*

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)\ dt \ , \quad x \in \mathbb{R}. \tag{2.29}$$

geschrieben werden. $f(t)$ heißt Dichte von X. Es gilt: $\int_{-\infty}^{\infty} f(t)\ dt = 1$.

Der Wert der Verteilungsfunktion an der Stelle x gibt also die Wahrscheinlichkeit an, dass die Ausprägung der Zufallsvariablen höchstens x beträgt. Bei kontinuierlichen Zufallsvariablen entspricht dieser Wert der Fläche, die von der Dichtefunktion, einer Parallelen zur y-Achse durch x und den beiden Achsen umschlossen wird.

Als ein wichtiges Modell in der Wahrscheinlichkeitstheorie und in der Statistik hat sich die *Normalverteilung* erwiesen. Es handelt sich dabei um eine spezielle Verteilung kontinuierlicher Zufallsvariablen.

Definition 2.7.5 (Normalverteilung) *Eine kontinuierliche Zufallsvariable X heißt normalverteilt mit den Parametern μ und σ^2, oder kurz: $N(\mu, \sigma^2)$-verteilt, wenn sie die folgende Dichte hat:*

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2}. \tag{2.30}$$

Der Funktionsgraph zu dieser Dichtefunktion ist die *Gaußsche Glockenkurve* (vgl. Abbildung B.14). Anschaulich gesprochen besagt Normalverteilung, dass die Ausprägung von X mit 68,2%-iger Wahrscheinlichkeit im Intervall $(\mu-\sigma, \mu+\sigma]$ liegt. Die Eigenschaften der Normalverteilung sind gemäß [Kre02] in vielen Fällen gut dazu geeignet, natürliche Vorgänge mittels Zufallsexperimenten zu modellieren.

Definition 2.7.6 (Standard-Normalverteilung) *Eine $N(0, 1)$-verteilte kontinuierliche Zufallsvariable X heißt standard-normalverteilt. Ihre Dichte ist*

$$f(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2} \tag{2.31}$$

und ihre Verteilungsfunktion wird mit $\phi(x)$ bezeichnet.

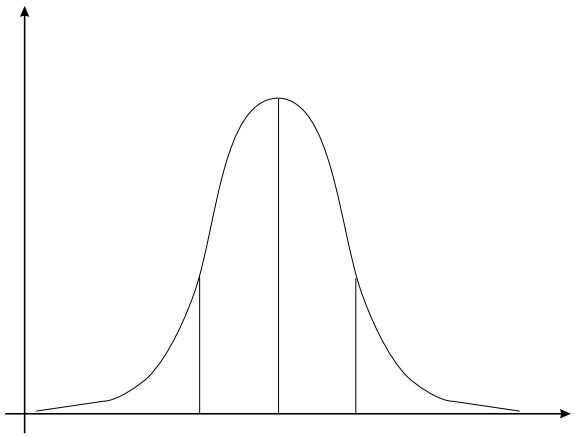


Abbildung 2.36: Die Normalverteilung

Normalverteilte Zufallsvariablen mit beliebigen Parametern lassen sich auf diese Standard-Normalverteilung zurückführen. Es gelten die folgenden Eigenschaften:

- $\phi(-x) = 1 - \phi(x)$
- X ist $N(\mu, \sigma^2)$-verteilt $\quad \Rightarrow \quad F(x) = \phi\left(\frac{x-\mu}{\sigma}\right)$

Diese Eigenschaften kann man sich nun zu Nutze machen: Da das Berechnen der Verteilungsfunktion von normalverteilten Zufallsvariablen recht aufwändig ist, sind die Werte der Standard-Normalverteilungsfunktion an bestimmten (positiven) Stellen x tabelliert. Mit Hilfe der oben erwähnten Eigenschaften ist es nun möglich, auch die Verteilungsfunktion von allgemein normalverteilten Zufallsvariablen an diesen Stellen zu ermitteln.

Momente von Zufallsvariablen

In der *beschreibenden Statistik* sind zu einer beliebigen Reihe von Messwerten $x_1, \ldots, x_n$ verschiedene Maßzahlen definiert, mit deren Hilfe sich die Messreihe charakterisieren lässt. Die für das Projekt wichtigen seien hier zur Erinnerung noch einmal aufgezählt:

arithmetisches Mittel: Es handelt sich hierbei um den Durchschnittswert aller Messwerte. Das arithmetische Mittel berechnet sich nach der Formel

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{2.32}$$

empirische Varianz: Die Varianz ist ein Maß für die Streuung der Messwerte um den Mittelwert. Sie ist wie folgt zu berechnen:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2 . \tag{2.33}$$

empirische Standardabweichung: Manchmal verwendet man an Stelle der Varianz die Standardabweichung. Diese ist nichts anderes als die Wurzel aus der Varianz:

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2}. \tag{2.34}$$

In Bezug auf Zufallsvariablen entsprechen die einzelnen Werte einer Messreihe den bei einer Reihe von Zufallsexperimenten festgestellten Ausprägungen der Zufallsvariablen. Analog zu den erwähnten Maßzahlen der beschreibenden Statistik gibt es daher charakteristische Kennzahlen für Verteilungen von Zufallsvariablen.

Zur Bestimmung des arithmetischen Mittels werden die einzelnen Messwerte aufsummiert und schließlich durch die Anzahl der Werte dividiert. Übertragen auf das hier eingeführte Modell entspricht dieses Verfahren dem Aufsummieren der möglichen Ausprägungen einer Zufallsvariablen, wobei jede Ausprägung mit ihrer Auftretenswahrscheinlichkeit gemäß dem Wahrscheinlichkeitsmaß P gewichtet wird. Es ergibt sich die Definition des *Erwartungswerts*:

Definition 2.7.7 (Erwartungswert) *Sei X eine diskrete und Y eine kontinuierliche Zufallsvariable. Dann sind die Erwartungswerte dieser Zufallsvariablen wie folgt definiert:*

$$E(X) = \sum_i x_i \cdot P(X = x_i) \tag{2.35}$$

$$E(Y) = \int_{-\infty}^{\infty} x \cdot f(x)\ dx. \tag{2.36}$$

Dabei ist zu beachten, dass der Erwartungswert einer Zufallsvariablen nur dann existiert, falls (im diskreten Fall) die Summe konvergiert bzw. (im kontinuierlichen Fall) absolute Integrierbarkeit gegeben ist.

Basierend auf dem Erwartungswert ist die *Varianz* von Zufallsvariablen definiert.

Definition 2.7.8 (Varianz einer Zufallsvariablen) *Sei X eine diskrete oder kontinuierliche Zufallsvariable. Dann ist die Varianz dieser Zufallsvariablen wie folgt definiert:*

$$Var(X) = E((X - E(X))^2). \tag{2.37}$$

Signifikanztests

Mit Hilfe von Beobachtungen, die man an Zufallsstichproben macht, ist es möglich, eine Aussage bzgl. einer Grundgesamtheit hinsichtlich der Dimension akzeptabel – nicht akzeptabel zu prüfen. Betrachtet man zum Beispiel als Grundgesamtheit alle Dortmunder Informatik-Studenten und dazu die Aussage „Wer regelmäßig Übungsblätter bearbeitet, schneidet in Prüfungen überdurchschnittlich gut ab.“. Da es zu aufwändig wäre, diese Aussage durch Befragung aller Studenten zu verifizieren, untersucht man nur eine Stichprobe und ermittelt anschließend, ob die Aussage haltbar ist, d.h., ob Abweichungen von der Aussage auf den Zufall zurückzuführen sind. Die zu diesem Zweck eingesetzten statistischen Hilfsmittel heißen *Signifikanztests*. Das prinzipielle Vorgehen bei solchen Tests soll im Folgenden erläutert werden. In Kapitel 2.7.4 werden dann ein paar ausgewählte Testverfahren vorgestellt.

Zu Beginn eines Signifikanztests muss zunächst einmal festgelegt werden, welche Aussage untersucht werden soll. Hieraus wird dann eine mathematische untersuchbare Aussage, die sogenannte *Nullhypothese*, abgeleitet. Eine solche Nullhypothese könnte z.B. lauten: „Der Erwartungswert von Zufallsvariable X ist größer als der Erwartungswert von Zufallsvariable Y“. Notiert wird diese Nullhypothese dann wie folgt:

$$H_0(E(X) > E(Y)) \tag{2.38}$$

Das Gegenereignis zur Nullhypothese heißt *Alternativhypothese* und wird (in diesem Beispiel) so notiert:

$$H_1(E(X) \leq E(Y)) \tag{2.39}$$

Nach Formulierung dieser Hypothesen wird ein *Testniveau* festgelegt. Dieses gibt an, in welchem Maße man bereit ist, die Nullhypothese abzulehnen, obwohl sie zutrifft. Natürlich wäre es ideal, in so einem Fall niemals abzulehnen, das Testniveau also auf 0 festzulegen; allerdings wächst

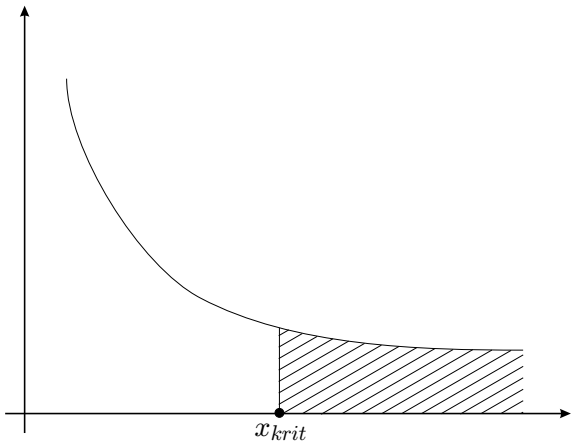


Abbildung 2.37: Illustration zum Signifikanztest

dadurch die Wahrscheinlichkeit, die Nullhypothese zu akzeptieren, auch wenn sie nicht zutrifft. Um dieses nachvollziehen zu können, sollte man sich vor Augen führen, dass Signifikanztests unter Verwendung von Zufallsstichproben arbeiten. Es ist daher niemals möglich, das Zutreffen oder Nicht-Zutreffen der Nullhypothese zu beweisen – man kann lediglich angeben, ob die Werte dafür oder dagegen sprechen, sie anzunehmen.

Nach Festlegung des Testniveaus wird eine Zufallsstichprobe bestimmt und der *Testwert* x_{test} ermittelt. Dieser ergibt sich durch Einsetzen der Stichprobenwerte in eine sogenannte *Teststatistik*. Die Teststatistik ist eine Zufallsvariable mit bekannter Verteilung, und der Testwert ergibt sich dann als eine Realisierung dieser Zufallsvariablen.

Bestimmte Realisierungen sind unter Annahme der Nullhypothese wahrscheinlich, andere Realisierungen eher unwahrscheinlich. Unter Zuhilfenahme des Testniveaus kann für jeden Signifikanztest ein *kritischer Wert* x_{krit} festgelegt werden, der den *Annahmebereich* der Nullhypothese von ihrem *Ablehnungsbereich* (entspricht dem Annahmebereich der Alternativhypothese) trennt. Schließlich wird x_{test} mit x_{krit} verglichen und die Nullhypothese auf Grund dieses Vergleichs entweder angenommen (x_{test} liegt im Annahmebereich von H_0) oder abgelehnt (x_{test} liegt im Annahmebereich von H_1). Im Falle der Ablehnung spricht man auch von einem *signifikanten Ergebnis*; die Abweichung des Testwerts vom Wert 0 ist dann nicht mehr durch Zufall (z.B. Messfehler) zu erklären (Die Teststatistik wird so gewählt, dass eine exakte Erfüllung der Nullhypothese den Wert 0 ergibt).

Abbildung 2.37 zeigt noch mal die wichtigsten Aspekte eines Signifikanztests in einer Skizze. Die Kurve deutet die Dichtefunktion der Teststatistik an. Der Punkt entspricht dem Wert x_{krit} und der schraffierte Bereich ist der Ablehnungsbereich. Liegt x_{test} links von x_{krit} – also im unschraffierten Bereich –, so wird H_0 angenommen.

Wie bereits erwähnt, kann ein Signifikanztest kein sicheres Resultat liefern, da er auf Zufallsstichproben beruht. Die Aussage eines Signifikanztests ist die Annahme oder Ablehnung der Nullhypothese, unter der Bereitschaft, einen Fehler zu begehen. Dieser Fehler lässt sich wie folgt klassifizieren:

Fehler 1. Art: Die Nullhypothese H_0 wird abgelehnt, obwohl sie zutrifft.

Fehler 2. Art: Die Nullhypothese H_0 wird angenommen, obwohl die Alternativhypothese H_1 zutrifft.

Die Wahrscheinlichkeit für einen Fehler 1. Art wird mit α bezeichnet und entspricht dem Testniveau. Ein Fehler 2. Art tritt mit Wahrscheinlichkeit β auf, wobei ein Zusammenhang zwischen α und β besteht. Als Faustregel kann man sich merken: „Je größer α, desto kleiner β.“.

Auf welchen Wert sollte man im konkreten Fall aber das Testniveau α festlegen? Die Beantwortung dieser Frage hängt natürlich davon ab, welche Fehlerart für die untersuchte Aussage schwerwiegender ist. Möchte man Fehler 1. Art möglichst vermeiden, so wählt man α eher klein, also z.B. $\alpha = 0,05$, $\alpha = 0,01$ oder $\alpha = 0,001$. Sollen hingegen Fehler 2. Art reduziert werden, muss man α groß wählen, z.B. $\alpha = 0,1$ oder $\alpha = 0,2$. In der Praxis wird häufig $\alpha = 0,05$ gesetzt.

Zum Abschluss dieses Kapitels noch eine Anmerkung: Die Reihenfolge der Schritte eines Signifikanztests sollte immer so gewählt werden, wie oben beschrieben. Der Grund ist einfach: Würde man als erstes die Zufallsstichprobe bestimmen, so könnte man die Teststatistik und das Testniveau so wählen, dass die Hypothese, deren Gültigkeit man untermauern möchte, angenommen wird. Um sich selbst davor zu schützen, die Statistik zur „glaubwürdigen“ Belegung von Aussagen zu missbrauchen, darf die Stichprobe also immer erst *nach* Festlegung der übrigen Parameter ausgewählt werden.

2.7.4 Ausgewählte Testverfahren

Nun sollen exemplarisch zwei Signifikanztests beschrieben werden: Der t-Test als ein Vertreter der sogenannten *parametrischen* Tests und der χ^2-Test zur Überprüfung von Normalverteilungen. Literaturgrundlage für die Signifikanztests ist [Käh95] und [LW85].

Der t-Test

Den t-Test gibt es in verschiedenen Varianten. An dieser Stelle soll der Zweistichproben-t-Test für unabhängige Stichproben erläutert werden. Dabei sind zwei Stichproben zu betrachten, deren Werte sich als Zufallsvariablen X_i bzw. Y_j darstellen lassen. Folgende Voraussetzungen sind zu erfüllen:

- $X_1, ..., X_m$ und $Y_1, ..., Y_n$ sind unabängige Zufallsvariablen.
- Die X_i sind identisch $N(\mu_1, \sigma^2)$-verteilt.
- Die Y_j sind identisch $N(\mu_2, \sigma^2)$-verteilt.

Die Parameter μ_1, μ_2 und σ^2 sind im Allgemeinen unbekannt – zu bemerken ist aber, dass alle Zufallsvariablen *denselben* Parameter σ^2 haben.

Null- und Alternativhypothese lauten beim t-Test:

$$H_0(\mu_1 = \mu_2) \tag{2.40}$$

$$H_1(\mu_1 \neq \mu_2) \tag{2.41}$$

Da Normalverteilungen durch die Parameter μ und σ^2 eindeutig bestimmt sind, lautet die Nullhypothese in Worten: „Die beiden Stichproben haben die gleiche Verteilung.“.

Die Teststatistik zur Überprüfung der Nullhypothese lautet:

$$T(X,Y) = \frac{\overline{X} - \overline{Y}}{s \cdot \sqrt{\frac{1}{m} + \frac{1}{n}}}, \quad \text{mit} \tag{2.42}$$

$$s^2 = \frac{1}{m+n-2} \left(\sum_{i=1}^{m} (X_i - \overline{X})^2 + \sum_{i=1}^{n} (Y_i - \overline{Y})^2 \right). \tag{2.43}$$

$T(X,Y)$ ist eine t_{m+n-2}-verteilte Zufallsvariable (die t_r-Verteilung ist eine im Verlauf der Normalverteilung ähnliche zum Ursprung symmetrische Verteilung).

Zur Testentscheidung wird nun das *Quantil* $t_{m+n-2;1-\alpha/2}$ der t_r-Verteilung benötigt. Es handelt sich hierbei um einen Wert mit der Eigenschaft, dass in Bezug auf die t_{m+n-2}-Verteilung $\left(1 - \frac{\alpha}{2}\right) \cdot$

100% der Werte kleiner und $\left(\frac{\alpha}{2}\right) \cdot 100\%$ der Werte größer sind als $t_{m+n-2;1-\alpha/2}$. Die Nullhypothese wird nun verworfen, falls folgende Ungleichung erfüllt ist:

$$|T(X,Y)| > t_{m+n-2;1-\alpha/2} \tag{2.44}$$

Quantile sind dabei in der Regel tabelliert, so dass sie nicht immer neu berechnet werden müssen.

Sind die Voraussetzungen für den (parametrischen) t-Test nicht erfüllt – liegt also z.B. keine Normalverteilung vor –, so kann alternativ ein nichtparametrischer Test verwendet werden. Der Mann-Whitney-U-Test ist ein solcher nichtparametrischer Test. Er soll an dieser Stelle allerdings nicht weiter betrachtet werden. Der interessierte Leser sei an die Literatur verwiesen (z.B. [Käh95]).

Der χ^2-Anpassungstest

Eine Voraussetzung für den t-Test ist die Normalverteilung der betrachteten Zufallsvariablen, wobei die exakten Parameter μ_1, μ_2 und σ^2 allerdings auch unbekannt sein dürfen. Wie kann man aber nun für eine Zufallsvariablen feststellen, ob sie normalverteilt ist? Die Antwort auf diese Frage ist der χ^2-Anpassungstest, eine Variante des χ^2-Tests. Er dient zur Überprüfung einer beliebigen Verteilung und geht von folgenden Voraussetzungen aus:

- X ist eine Zufallsvariable; ihr Wertebereich kann disjunkt zerlegt werden: $I_1 \cup I_2 \cup \ldots \cup I_r$
- $p_j = P(X \in I_j)$, $j = 1, ..., r$, sind die Wahrscheinlichkeiten, dass eine Ausprägung in ein bestimmtes Intervall fällt.
- Die entsprechenden Wahrscheinlichkeiten der *angenommenen* Verteilung sind $p_1^0, ..., p_r^0$

Wie können diese Wahrscheinlichkeiten bestimmt werden, wenn doch die Parameter der Normalverteilung ggf. unbekannt sind? In diesem Falle werden zwei *Schätzer* für die Parameter verwendet:

$$\widehat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.45}$$

$$\widehat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \widehat{\mu})^2. \tag{2.46}$$

Es soll untersucht werden, ob X die angenommene Verteilung besitzt. Dies ist der Fall, wenn die tatsächlichen Wahrscheinlichkeiten p_i und die angenommenen Wahrscheinlichkeiten p_i^0 übereinstimmen. Die Nullhypothese lautet also:

$$H_0((p_1, ..., p_r) = (p_1^0, ..., p_r^0)) \tag{2.47}$$

Für die Teststatistik werden noch die absoluten Häufigkeiten y_j benötigt, mit denen Werte aus den oben definierten Intervallen I_j angenommen werden:

$$y_j = |\{x_i | x_i \in I_j\}| \quad i = 1, ..., n; j = 1, ..., r \tag{2.48}$$

Als Teststatistik verwendet der χ^2-Anpassungstest die sogenannte χ^2-Abstandsfunktion:

$$Q(y_1, ..., y_r; p_1^0, ..., p_r^0) = \sum_{j=1}^{r} \frac{\left(y_j - np_j^0\right)^2}{np_j^0} = \left(\sum_{j=1}^{r} \frac{y_j^2}{np_j^0} \right) - n \tag{2.49}$$

Die Testentscheidung erfolgt wiederum durch den Vergleich des ermittelten Werts mit einem Quantil. Es handelt sich bei dem χ^2-Test allgemein um das Quantil $\chi^2_{r-k-1;1-\alpha}$, wobei k die Anzahl der geschätzten Parameter bezeichnet. In dem speziellen Fall des χ^2-Anpassungstests ist $k = 2$. Die Nullhypothese wird verworfen, falls die folgende Ungleichung erfüllt ist:

$$Q(y_1, ..., y_r; p_1^0, ..., p_r^0) > \chi^2_{r-3;1-\alpha} \tag{2.50}$$

2.7.5 Untersuchte Parameter des Zell-Trackings

Nach dieser Einführung in die Grundlagen der Statistik stellt sich nun die Frage, wie diese Ergebnisse im Rahmen des CellTrack-Projekts verwendet werden können. Dazu sollte man sich zunächst vor Augen führen, wie die Untersuchungen ablaufen, die die Biologen mit den Zellkulturen durchführen.

Folgende Parameter sollen bei einer Untersuchung gemessen werden:

- **Migrationsaktivität** (locomoting cells [%]): Wie viele Zellen haben sich im Beobachtungszeitraum bewegt?
- **Migrationsgeschwindigkeit** (speed $[\mu m/h]$): Wie schnell haben sich die Zellen bewegt, wenn sie nicht gerade pausiert haben?
- **Migrationsdistanz** (distance $[\mu m]$): Welchen Weg haben die Zellen zurückgelegt?
- **Pausenlänge** (length of break $[min]$): Wie lange haben die Zellen pausiert?
- **Pausenfrequenz** (frequency of breaks $[12h^{-1}]$): Wie oft wurden Pausen eingelegt?
- **Richtung** (direction [°]): In welche Richtung haben sich die Zellen bewegt? Für diesen Parameter gibt es keine allgemeingültige Definition. Vorstellbar ist aber z.B. der Winkel zwischen der horizontalen Achse (als Bezugspunkt) und dem Richtungsvektor $\mathbf{p} = \overline{p_i p_{i+1}}$, wobei p_i die Position einer Zelle zum Zeitpunkt i kennzeichnet.
- **Persistenz** (persistence): Wie zielstrebig haben die Zellen sich in eine Richtung bewegt? Zur Ermittlung der Persistenz wird eine Zelle zu zwei verschiedenen Zeitpunkten i und $i + n$ betrachtet und der Quotient aus dem euklidischen Abstand von p_i und p_{i+n} und der Migrationsdistanz berechnet.

Um diese Maße errechnen zu können, werden als Messdaten offensichtlich die Koordinaten der betrachteten Zellen zu den Beobachtungszeitpunkten benötigt. Außerdem sollte bezüglich der Koordinaten ein Nullpunkt festgelegt werden. Alle Parameter lassen sich dann aus diesen Daten ermitteln. Die Merkmale „Koordinaten“ und „Zeitpunkt“ sind intervallskaliert und erfüllen damit die Voraussetzung für die statistische Auswertung.

Die Untersuchungsdurchführung ist bei allen Parametern ähnlich, daher wird im Folgenden das Vorgehen anhand der Migrationsaktivität erläutert. Dieser Parameter ist für die Biologen zugleich der wichtigste.

Konkret soll untersucht werden, welchen Einfluss die Zugabe verschiedener Stimuli zu der Zellkultur für Auswirkungen auf das Migrationsverhalten der Zellen hat. Es werden pro Versuch daher zwei Ansätze verwendet: Einer mit und einer ohne Stimulus. Aus beiden Ansätzen werden je 30 Zellen ausgewählt, die in 1-minütigen (bei Leukozyten) bzw. 15-minütigen Abständen (bei Tumorzellen) beobachtet werden (Leukozyten weisen eine stärkere Schwankung in ihrem Verhalten auf als Tumorzellen). Nach einer Anfangsphase erreichen die Zellen den sogenannten *steady state*, d.h. ihre Aktivität hat sich „eingependelt“. Aus den Beobachtungsdaten wird nun pro Ansatz die mittlere Aktivität im steady state berechnet. Dieser Versuch wird noch zweimal wiederholt, so dass am Ende 3 Wertepaare vorliegen. Und an dieser Stelle kommt die Statistik ins Spiel: Die Frage ist ja, ob die Zugabe des Stimulus eine Veränderung der Zellaktivität hervorgerufen hat, die nicht allein durch Messfehler oder sonstige Einflüsse erklärt werden kann. Die beiden Messreihen mit jeweils 3 Werten werden also einem Signifikanztest unterworfen, wobei bislang ausschließlich der t-Test bei einem Testniveau von $\alpha = 0,05$ (vgl. Kapitel 2.7.4) verwendet wurde.

Voraussetzung für den t-Test ist allerdings, dass die zu Grunde liegenden Stichprobenwerte normalverteilt sind. Diese Voraussetzung wurde bislang jedoch nicht explizit überprüft, sondern lediglich anhand der beobachteten Schwankungen sowohl bei Tumorzellen als auch bei Leukozyten angenommen. Details zur Versuchsdurchführung und -auswertung sind in [NZ00] und [ZE04] zu finden.

2.7.6 Zusammenfassung

Zur Untersuchung von Zellmigrationen kann eine ganze Reihe verschiedener Parameter herangezogen werden (vgl. Kapitel 2.7.5). Die Hauptfrage ist dabei stets, ob die Zugabe eines Stimulus zu der Zellkultur die Parameter des Migrationsverhaltens beeinflussen kann. Diese Frage kann mit Hilfe von statistischen Werkzeugen, den Signifikanztests, untersucht werden. Man muss dabei allerdings immer im Hinterkopf behalten, dass Signifikanztests nicht in der Dimension „wahr – unwahr“, sondern in der Dimension „wahrscheinlich – unwahrscheinlich“ entscheiden. Es wird also niemals eine Aussage der Form „Der Stimulus hat die Aktivität beeinflusst.“ verifiziert werden können. Ein Signifikanztest liefert vielmehr Ergebnisse wie „Die Wahrscheinlichkeit, dass die Änderung der Aktivität auf den Stimulus und nicht auf den Zufall zurückzuführen ist, kann als hoch angesehen werden.“.

Ein weiterer wichtiger Punkt ist die Beachtung der Voraussetzungen von Signifikanztest. So geht der t-Test z.B. von der Normalverteilung der Stichprobenwerte aus. Natürlich kann man es auf Grund von Beobachtungen als wahrscheinlich ansehen, dass diese Bedingung erfüllt ist – mathematisch genau ist dieses Vorgehen allerdings nicht. Mit dem χ^2-Anpassungstest wurde in Kapitel 2.7.4 ein Werzeug vorgestellt, mit dem sich die Überprüfung der Normalverteilungsannahme mathematisch durchführen lässt.

Sollte wider Erwarten die Annahme der Normalverteilung nicht gerechtfertigt sein, so muss auf einen anderen Test ausgewichen werden. In diesem Fall kann z.B. der nicht-parametrische U-Test von Mann-Whitney verwendet werden. Es ist dabei allerdings zu beachten, dass nicht-parametrische Tests schwächer sind als parametrische. Konkret bedeutet das: Liegen tatsächlich unterschiedliche Verteilungen vor, wird das ein parametrischer Test eher erkennen als ein nicht-parametrischer.

Kapitel 3

Verfahren

Inhaltsangabe

3.1 Grabbing

In CellTrack werden die Bilddaten der Zellen in digitaler Form benötigt. Als Grundlage für die Digitalisierung wurde Video for Linux II gewählt, da hierdurch eine breite Palette an Hardware zu Verfügung steht. Im folgenden Abschnitt wird Video for Linux II näher erläutert und ein ein einfacher Filter zur Rauschunterdrückung vorgestellt.

3.1.1 V4L2 - Video for Linux II

Video for Linux II (v4l2) ist eine Erweiterung von Video for Linux, die jedoch nicht abwärtskompatibel zu v4l ist. v4l2 beseitigt einige Designfehler von v4l und unterstützt mehr Geräte. Wie erwähnt ist v4l2 nicht abwärtskompatibel, deswegen können Treiber von v4l nicht genutzt werden

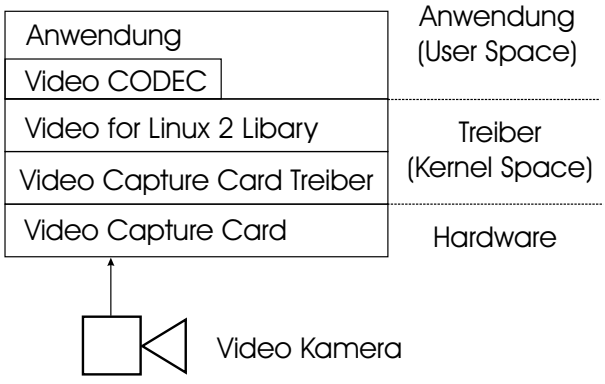


Abbildung 3.1: Die Funktionsweise von v4l2

Device Name	Type of Device
/dev/video	Video capture interface
/dev/vfx	Video effects interface
/dev/codec	Video codec interface
/dev/vout	Video output interface
/dev/radio	AM/FM radio devices
/dev/vtx	Teletext interface chips
/dev/vbi	Data services interface

Tabelle 3.1: Video for Linux II - Devices

und müssen angepasst werden. Es gibt aber einen Layer im high-level Treiber, der die alten `ioctl`-Befehle umwandelt, so dass auch manche Programme, die für die alte API geschrieben wurden, lauffähig sind.
Inzwischen ist v4l2 im Standardkernel von Linux integriert. Es dient als einheitliche Programmier-Schnittstelle zu diversen Videogeräten. Die grobe Funktionsweise ist in Abb. 3.1 dargestellt. Eine Kamera oder ein anderes Aufnahmegerät ist an eine Video Capture Karte angeschlossen. Für diese Karte läuft im Linux-Kernel ein Softwaretreiber. Über diesem Treiber, aber immer noch im Kernel, liegt die Video for Linux 2 Libary, die API für den Programmierer. Da v4l2 nicht alle Videoformate unterstützt werden manchmal auch noch CODECS benötigt, um die gewünschten Videos zu erhalten.
Im Projekt wurde v4l2 genutzt um die Videodaten von den Kameras der Mikroskope zu digitalisieren und so qualitativ besser Aufnahmen zu erhalten.

Video for Linux II-Devices

Video for Linux II unterstützt die unterschiedlichsten Videodevices, wie man auch unter [Dir99] nachschlagen kann. Eine Auflistung ist in Tabelle 3.1 zu finden.

Für das Projekt sind nur die *Video Capture Devices* von Bedeutung, da Videosignale von Kameras digitalisiert werden sollen.
Die einzelnen Schritte, die nötig sind um Daten von einer Videoquelle über die API von Video for Linux II zu erhalten [Dir01], werden im Anschluss erläutert.

Öffnen von Video for Linux II - Devices

Der erste Schritt, um Daten von einem Video for Linux II-Device zu bekommen, ist es, das Device zu initialisieren und zu öffnen.
Mittels `VIDIOC_QUERYCAP` wird überprüft, ob überhaupt ein Video for Linux II-Device vorhanden ist. Anschließend muss sichergestellt sein, dass das Device auch Frames oder Daten digitalisieren kann. Dafür muss das Flag `V4L2_CAP_VIDEO_CAPTURE` in den Device Flags gesetzt sein. Sind diese Voraussetzungen erfüllt, muss man nur noch festlegen, in welchem Format man die Daten bekommen will. Hierfür dient der Aufruf von `VIDIOC_S_FMT`. Vorher muss man noch die Struktur `v4l2_format` mit Inhalt füllen. Als `type` muss `V4L2_BUF_TYPE_VIDEO_CAPTURE` gesetzt werden, um Videodaten einzulesen. Daneben muss hier noch die Struktur `v4l2_pix_format` mit Bildgröße und Farbtiefe gefüllt werden. Wichtig ist noch das Pixelformat (vgl. Tabelle 3.2). Hier wird angegeben, wie die Bildinformationen abgespeichert werden. Nun ist das Device bereit um Daten zu liefern.

Daten von Video for Linux II-Devices lesen

Nachdem das Device geöffnet wurde, können jetzt die Daten gelesen werden. Hierfür gibt es prinzipiell drei verschiedene Arten. Der Benutzer kann einen eigenen Buffer verwenden, einen Buffer erzeugen lassen oder die Daten über einen `read()`-Aufruf einlesen. Vorher muss mit einem `select()`-Aufruf gewartet werden, bis Daten geliefert werden. Wird das Einlesen beendet muss

Pixelformat	Tiefe	Beschreibung
V4L2_PIX_FMT_RGB332	8	RGB-3-3-2, ein Byte pro Pixel RGB
V4L2_PIX_FMT_RGB555	16	RGB-5-5-5 Gepacktes RGB Format
V4L2_PIX_FMT_RGB565	16	RGB-5-6-5 Gepacktes RGB Format
V4L2_PIX_FMT_BGR24	24	RGB-8-8-8 Gepackt in 24-bit
V4L2_PIX_FMT_RGB24	24	RGB-8-8-8 Gepackt in 24-bit
V4L2_PIX_FMT_BGR32	32	RGB-8-8-8
V4L2_PIX_FMT_RGB32	32	RGB-8-8-8
V4L2_PIX_FMT_GREY	8	Graustufen
V4L2_PIX_FMT_YUV410	9	YUV 4:1:0
V4L2_PIX_FMT_YUV420	12	YUV 4:2:0
V4L2_PIX_FMT_YUYV	16	YUV 4:2:2
V4L2_PIX_FMT_UYVY	16	Wie YUYV, nur U-Y-V-Y Byte Reihenfolge

Tabelle 3.2: Video for Linux II - Pixelformate

bei den ersten beiden Varianten über `VIDIOC_STREAMOFF` dem Device mitgeteilt werden, dass keine Daten mehr benötigt werden.

3.1.2 Rauschfilter

Bei dem Videomaterial fiel auf, dass die Bilder leicht verrauscht waren. Um diesen Effekt zu minimieren, wurden n aufeinander folgende Frames betrachtet und für jedes Pixel der Helligkeitswert H aus diesen Frames interpoliert:

$$H(x,y) = \frac{\sum_{i=1}^{n} H(x,y)_i}{n}. \tag{3.1}$$

Die Videokarte liefert ca. 20 Bilder pro Sekunde. In dieser Zeit bewegen sich die Zellen nicht spürbar, deshalb verschwimmen die Bewegungen der Zellen nicht, aber das Rauschen wird merklich geglättet.

3.1.3 libMNG - eine Bibliothek für den Zugriff auf Videos im MNG-Format

Die libMNG ist eine Bibliothek in C++, mit der man auf Videos im MNG-Format zugreifen kann. Das MNG-Format ist eine Erweiterung des PNG-Formats um Videos abspeichern zu können. Für das Projekt wurde am Anfang dieses Format gewählt, da es die Daten verlustfrei abspeichert. Im Laufe der Implementierung gab es jedoch Probleme mit dem damals aktuellen Code. Funktionen um die Videos abzuspeichern waren noch nicht implementiert.

3.2 Filter

Wie im Teil des Grundlagenkapitels über digitale Bildverbesserung bereits angesprochen wurde, gibt es sowohl Filter im Ortsbereich, als auch Filter im Ortsfrequenzbereich (vgl. Kapitel 2.4.4). In CellTrack wurden nur Filter verwendet, die im Ortsbereich arbeiten. Die Transformation der Bilder vom Ortsbereich in den Ortsfrequenzbereich, der üblicherweise mit der Fouriertransformation durchgeführt wird, wurde nicht durchgeführt, weil dies Laufzeitverschlechterungen verursacht hätte. Für CellTrack reichten Filter aus, die für das Tracking und Segmentieren eine Vorverarbeitung garantieren sollten. Die Filter sind lineare Filter, die mit dem Bild nach dem Faltungssatz gefaltet werden.

$$f(m,n) * *h(m,n) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f(k,l)h(m-k,n-l) \tag{3.2}$$

Die Matrix **F** steht für das Bild und die Matrix **H** ist der Filter, der auf das Bild angewendet werden soll. Es wird im Weiteren auf die beiden im System benutzten Filter, den Gaußfilter und den Sobeloperator, näher eingegangen.

3.2.1 Gaußfilter

Der Gaußfilter dient der Glättung des Bildes [GW02]. Er wird in CellTrack dazu verwendet Rauschen herauszufiltern. Die Matrix **h** ergibt sich aus der diskreten zweidimensionalen Anwendung der Gauß'schen Normalverteilung zur Glättung. Die nächsten beiden Matrizen zeigen **h** für den 3x3-Fall und den 5x5-Fall:

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \cdot \frac{1}{16} \text{ und } \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} \cdot \frac{1}{256}$$

Der Gaußfilter ist ein lokaler Operator, bei dem weiter entfernte Nachbarn des betrachteten Punktes immer weniger stark eingehen. Kanten bleiben so eher erhalten, als mit einem Mittelwertfilter. Der Effekt des Operators steigt mit der Größe der Matrix **H**.

3.2.2 Sobeloperator

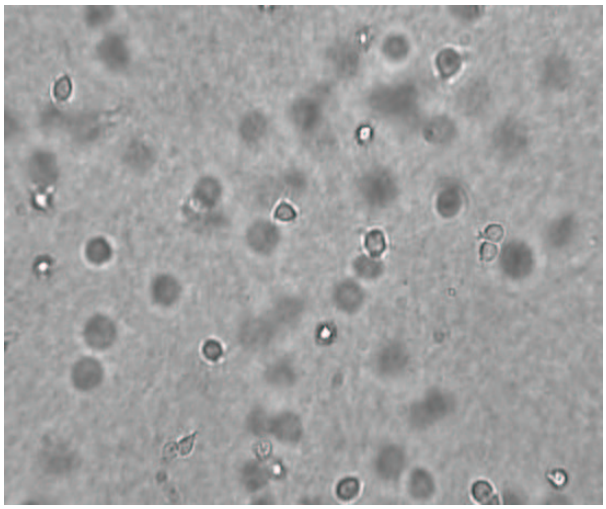
Der Sobeloperator gehört zu der Familie der Kantenoperatoren [GW02]. Er soll den Übergang zwischen homogenen Regionen im Bild herausstellen. Kanten sind meistens mit Objektgrenzen assoziiert. In CellTrack ist das der Übergang zwischen Hintergrund und Zelle. Der Kantenfilter hat die Aufgabe die Übergänge zu verstärken. Die linearen lokalen Kantenoperatoren schätzen dabei die erste bzw. die zweite Ableitung der Bildfunktion mit Differenzenoperationen ab.
Beim Sobeloperator werden vertikale und horizontale Kanten hervorgehoben. Die Faltungsmatizen für die beiden Richtungen sind

$$\mathbf{H_x} = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \text{ und } \mathbf{H_y} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

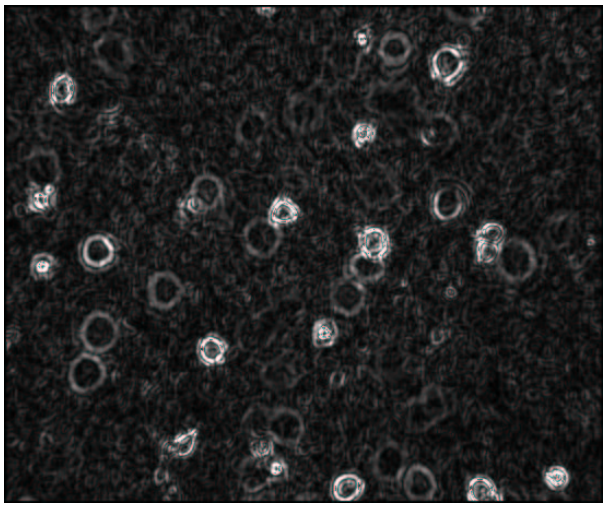
Die Kombination der beiden Ergebnisse liefert durch $\mathbf{H} = \sqrt{\mathbf{H_x}^2 + \mathbf{H_y}^2}$ ein richtungsunabhängiges Ergebnis. In CellTrack stellte sich heraus, dass die Anwendung des Sobeloperators nach der Anwendung eines Glättungsoperators geschehen sollte, so dass weniger Rauschen verstärkt wird. Das Ergebnis des Operators kann man in Abbildung 3.2 begutachten.

3.2.3 Optimaler Schwellenwert

Der optimale Schwellenwert ist eine Mischung aus einer Fuzzy Schwellenwertbestimmung und einer fest kodierten Schwellenwertauswahl. Der Fuzzy Logik benutzende Teil berechnet für jede Position im Histogramm des Bildes die Zugehörigkeiten jedes Farbtones anhand einer S-Fuzzy Funktion. Die gewählte Bandbreite der S-Funktion ist 50. Die Funktion wird so über das Histogramm gezogen, dass der Crossover-Punkt einmal an jeder Position des Histogramms liegt und dann die Zugehörigkeit mit der S-Funktion berechnet wird. Dies führt zu einer starken Verwischung des Histogramms. Schwache Maxima im Histogramm, wie beispielsweise durch einen schwarzen Bildrand hervorgerufen, werden somit entfernt (vgl. Abbildung 3.3, das Histogramm ist geglättet, es gibt nur noch ein Maximum). Dieses verwischte Histogramm macht sich der fest kodierte Teil zunutze. Es wurde heuristisch ermittelt, dass sich für das vorliegende Bildmaterial eine optimale



(a) Originalbild



(b) Gaußgefiltertes Gradientenbild (Sobeloperator)

Abbildung 3.2: Kombination von Gaußfilter und Sobeloperator

Segmentierung einstellt, wenn ein Schwellenwert genau auf halber Höhe der linken Flanke des Berges gewählt wird (vgl. Abbildungen 3.4 und 3.5). Der zugehörige Farbindex stellt einen optimalen Schwellenwert dar.

3.3 Segmentierung

Nach den ersten Tracking-Tests mit dem Blockvergleichverfahren wurde deutlich, dass eine Segmentierung zur Verbesserung der Ergebnisse unumgänglich war. Da parallel an mehreren Trackingverfahren gearbeitet wurde, entstanden zwei Segmentierungsverfahren, die auf die Bedürfnisse der Trackingverfahren angepasst wurden. Zur Unterstützung des Trackings sollten in jedem Frame Ausschnitte, in denen sich eine Zelle befindet, segmentiert werden. Die Segmentierer sollten dann angeben, welcher Teil des Ausschnitts zum Hintegrund gehört und welcher Teil zu der Zelle, also zum Vordergrund. Die benutzten Verfahren sind Bereichswachstumverfahren und Fuzzy Segmentation. Sie werden in den nächsten Abschnitten genauer beschrieben.

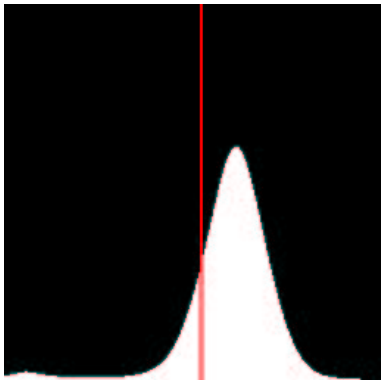


Abbildung 3.3: Histogramm mit Fuzzy-Methoden geglättet

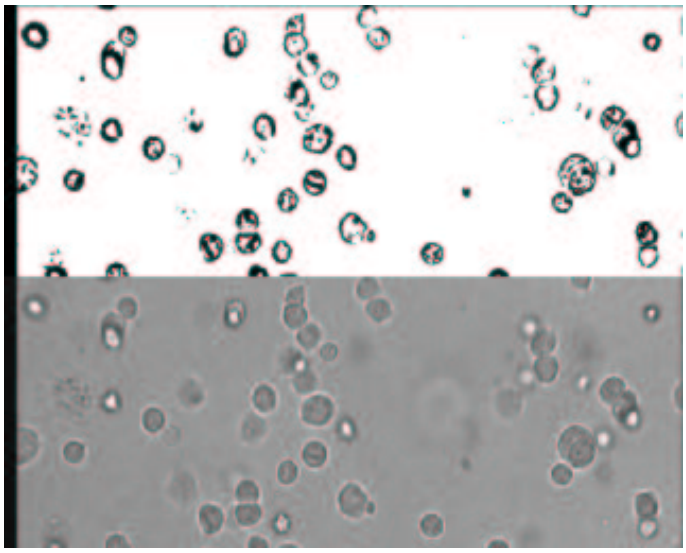


Abbildung 3.4: Optimaler Schwellenwert auf Bild in guter Qualität angewandt

3.3.1 Bereichswachstumverfahren

Dem Bereichswachstumverfahren wird ein Bildausschnitt übergeben, der die Zelle enthält [AB94]. Der Bildausschnitt sollte so gewählt werden, dass die Zelle vollständig in ihm liegt und möglichst keine Zellbereiche benachbarter Zellen in ihm auftauchen. Ausgehend vom Mittelpunkt der Zelle werden alle Nachparpixel auf ein bestimmtes Kriterium hin überprüft. Ist dieses Kriterium erfüllt, wird das Pixel zum Bereich, der zur Zelle gehören soll, hinzugefügt. Der segmentierte Bereich, der zu einer Zelle gehört, wird durch eine eindeutige Einfärbung der Pixel realisiert. Als sinnvolles Kriterium für die Zugehörigkeit hat sich die Grauwertdifferenz zwischen zwei Pixeln erwiesen, da die Grauwertverteilung über eine Zelle relativ homogen ist und sich die Zelle vom Hintergrund abheben sollte. Die Grauwertdifferenz als Kriterium hat jedoch die Schwachstelle, dass der Schwellenwert, der darüber entscheidet, ob das Kriterium erfüllt wird oder nicht, nicht konstant über einer Zelle ist und der Rand der Zelle nicht immer eine scharfe Kante ist. Dies macht es erforderlich, den Schwellenwert ständig zu aktualisieren, um nicht zuwenig von einer Zelle und zuviel vom Hintergrund zu segmentieren. Erschwerend kommt hinzu, dass nur relative Angaben über den Schwellenwert gemacht werden können, wodurch der Wahl des Mittelpunktes entscheidende

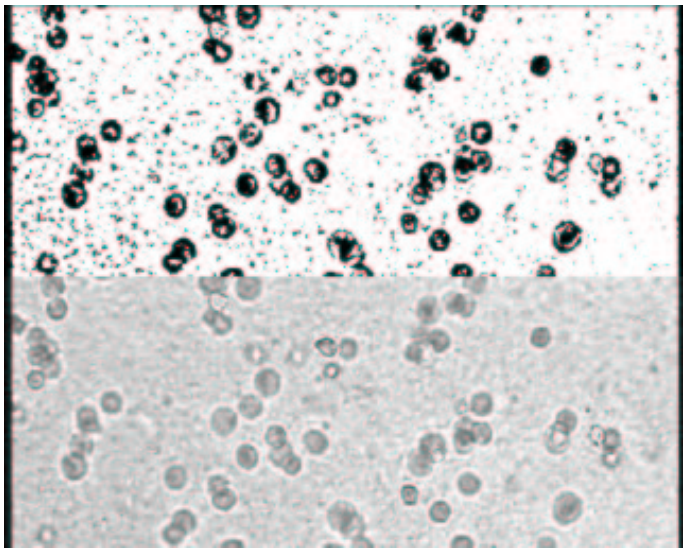


Abbildung 3.5: Optimaler Schwellenwert auf Bild in schlechter Qualität angewandt

Bedeutung zu kommt. Abhängig vom Grauwert des Mittelpunktes ergeben sich andere Schwellenwerte für die Segmentierung einer Zelle. Unter der Annahme, dass sich die Zelle in der Mitte des übergebenen Bildausschnitts befindet, war eine Kontrolle des Mittelpunktes nur indirekt über die Wahl des Bildausschnittes möglich.

Um das Problem abzuschwächen, wird zuvor eine Binarisierung des Bildes durchgeführt (vgl. Abbildung 3.6). Für die Binarisierung wird ebenfalls ein Schwellenwert benötig, der festlegt, ab welchem Grauwert ein Pixel weiß gezeichnet werden soll und somit als Hintergrund angesehen wird. Alle anderen Pixel gehören dann zur Zelle und werden schwarz gezeichnet. Die Wahl eines Schwellenwertes für die Binarisierung ist deutlich einfacher, als die Wahl eines Schwellenwertes für jede Zelle. Für die Binarisierung reicht es, einen Schwellenwert für alle Zellen zu bestimmen, da er auf Basis der Grauwerte des Hintergrundes berechnet werden kann und sich alle Zellen sichtbar vom Hintergrund abheben sollten (vgl. Kapitel 3.2.3). In dem so entstandenen Bild hebt sich die Zelle als schwarze Fläche deutlich vom weißen Hintergrund ab. Der Verlust einiger Konturbereiche der Zelle ist dabei nicht so schwerwiegend, da durch den gewonnenen Kontrast die Segmentierung sehr vereinfacht wird. Trotz dieser Verbesserungen bleiben aber noch Probleme bei der Segementierung.

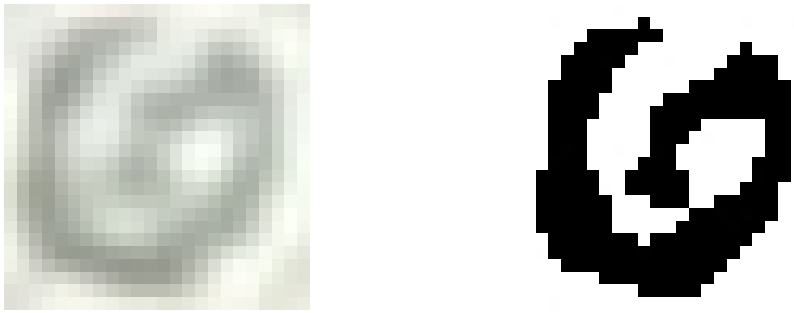


Abbildung 3.6: Beispiel eines typischen Bildausschnitts für das Bereichswachstumverfahren und das dazugehörige binarisierte Bild.

Abbildung 3.7: Eine vollständig segmentierte Zelle (rot ≙ Pixel gehört zur Zelle, grün ≙ Pixel gehört nicht mehr zur Zelle).

Ausgehend vom Startpunkt wird nun auf diesem Bild das eigentliche Bereichswachstumverfahren gestartet. Zuerst wird das nach oben benachbarte Pixel betrachtet, ob es schwarz oder weiß ist. Ist das Pixel schwarz, so gehört es mit zur Zelle und wird rot eingefärbt, andernfalls gehört es nicht zur Zelle und wird grün eingefärbt. Die erneute Einfärbung dient dazu, dem Algorithmus zu zeigen, welche Pixel bereits betrachtet wurden und somit jedes Pixel nur einmal getestet wird. Stößt der Algorithmus auf ein weißes oder grünes Pixel, so muss das Verfahren in diese Richtung nicht weiter fortgestezt werden. Nachdem das Verfahren so lange nach oben gegangen ist, wie es möglich ist, wird das rechte, das untere und das linke Pixel vom letzten besuchten Pixel, nacheinander betrachtet. Ist eines dieser Pixel schwarz, wird das Bereichswachstumverfahren in diese Richtung fortgesetzt. Wenn ein neues Pixel erreicht wird, wird zuerst wieder das nach oben benachbarte Pixel getestet.
Wichtig dabei ist es, alle bereits besuchten Pixel in einem Vektor zu speichern. Falls das Verfahren in eine Sackgasse läuft, kann der Vektor rückwärts durchlaufen werden, um ein Pixel zu ermitteln, dessen vier benachbarte Pixel noch nicht alle überprüft wurden. Das Verfahren endet, sobald der Vektor vollständig abgebaut wurde und somit alle Pixel, die zur Zelle gehören könnten, getestet wurden. Der Bereich, in dem sich der Startpunkt des Verfahrens befindet, ist damit vollständig segmentiert (vgl. Abbildung 3.7). Auf Grund der Binarisierung kommt es aber oft vor, dass eine Zelle nicht als vollständig geschlossene Fläche auftritt, sondern aus mehreren kleinen Bereichen aus dem binarisierten Bild zusammengesetzt werden muss.
Um das Problem zu umgehen, wurden zwei Ansätze gewählt. Zuerst wird, ausgehend vom Startpunkt, das nächstgelegene schwarze Pixel im binarisierten Bild gesucht, auf dem dann das eigentliche Bereichswachstumverfahren ausgeführt wird. Die Suche findet in alle acht möglichen Richtungen statt, so dass sich die Anzahl der Startpunkte von eins auf acht erhöht. Da sich dies auch als unzureichend darstellte, wurden zusätzlich auch noch Zufallspunkte hinzugenommen. Diese Zufallspunkte haben neben ihrer Position auch noch eine zufällige Richtung, in der, falls sie nicht auf einem schwarzen Pixel liegen, nach dem nächsten schwarzen Pixel gesucht wird. Als sinnvoll haben sich vier bis acht zufällige Punkte erwiesen, so dass die Gesamtanzahl der Startpunkte zwischen 12 und 16 liegt.
Um eine erste Evaluierung der Segmentierung zu erhalten, wird die Anzahl der Pixel, die für eine Zelle gefunden wird, mit der Anzahl der schwarzen Pixel, die durch die Binarisierung entstehen, verglichen. Ist die Anzahl der segmentierten Pixel viel geringer als die Anzahl der schwarzen Pixel, so wird das Verfahren mit neuen Zufallspunkten wiederholt.
Die Ergebnisse der einzelnen Segmentierungen werden danach zu einem Bild zusammengefasst, so dass sich ein möglichst vollständig segmentiertes Bild einer Zelle ergibt. Für den Fall, dass es nicht möglich ist genügend Pixel der Zelle zu segementieren, wurde ein weiteres Abbruchkriterium hinzugefügt, so dass nach spätestens zehn Iterationen das Verfahren auf jeden Fall terminiert. Da

die Evaluierung innerhalb des Verfahrens aber nicht alle Problemfälle abfangen kann, werden die Ergebnisse nach dem Verfahren noch einmal überprüft. Die externe Methode vergleicht die Größe des segmentierten Bereichs mit früheren Segmentierungen, fängt Fehler wie etwa zu kleine Bereiche ab und berechnet die genaue Position der Zelle im Bild anhand der extremen x- und y-Koordinaten des segmentierten Bereichs.

3.3.2 Fuzzy Segmentation

Die Fuzzy Segmentation ist ein Segmentierer, der verschiedene Methoden der Bildsegmentierung zusammenschaltet, und speziell für die Segmentierung von Zellen entwickelt wurde. Die Eingabe ist ein zweidimensionales Bild und eine untere und obere Schranke für die Flächengröße einer Zelle. Das Bild zeigt die zu segmentierenden Zellen manchmal mehr, manchmal weniger deutlich abgegrenzt zu dem Hintergrund. Ziel der Segmentierung ist es, die Bereiche des Eingabebildes zu markieren, in denen sich Zellen befinden, und alles andere als Hintergrund auszuzeichnen. Der Zugerhörigkeitswert, dass ein Pixel auf jeden Fall zu einer Zelle gehört, wird in einer Matrix festgelegt. Dabei heißt ein Zugehörigkeitswert von Eins, dass alle Verfahren, die im folgenden ersten Schritt benutzt werden, an dem entsprechenden Pixel eine Zelle erkannt haben. Ein Zugehörigkeitswert von Null bedeutet andersherum, dass kein Verfahren auf dem betrachteten Pixel eine Zelle vermutet. Diese Zuordnung eines Zugehörigkeitswertes zu einem Pixel kann auch als eine charakteristische Funktion aus der Fuzzy-Theorie verstanden werden. Die Unschärfe ist dabei der Übergang von Hintergrund zu der Zelle. Informationen zur Fuzzy-Theorie sind in [BKI03] nachzulesen. An dieser Stelle zeigt sich die Flexibilität der Fuzzy Segmentation, die aus der Kombination unterschiedlicher Verfahren herrührt. Das gesamte Verfahren gliedert sich insgesamt in fünf wesentliche Schritte.

1. Im ersten Schritt der Fuzzy Segmentation wird das Eingabebild abgetastet.
   Die erste Abtastung erfolgt in Scanlinien von oben nach unten. Für jede Scanlinie wird zuerst der mittlere Grauwert ermittelt, welcher danach mit den Grauwerten der Pixel in der Spalte verglichen wird. Liegt die Differenz zwischen dem mittleren Grauwert und dem Grauwert des Pixels unter einer vorgegebenen Schranke, so wird der Zugerhörigkeitswert erhöht. Andernfalls bleibt der Zugehörigkeitswert gleich.
   In einem nächsten Vergleich wird geprüft, ob der Grauwert des Pixels unter dem optimalen Schwellenwert des Bildausschnitts liegt. Wenn dies der Fall ist, wird der Zugehörigkeitswert erhöht. Der nächste Vergleich bezieht sich auf die Kantendetektion. Dabei werden zwei benachbarte Grauwerte von Pixeln voneinander abgezogen und die Differenz betrachtet. Liegt die Differenz über einem bestimmten Schwellenwert, so geht man davon aus, dass es sich hier um eine Kante im Bild handelt, und damit eventuell auch um eine Kante der Zelle. Deshalb wird auch hier der Zugehörigkeitswert erhöht. Die Schritte werden danach noch einmal für Scanlinien von links nach rechts durchlaufen. Die Vergleiche sind hier analog dem Fall von oben nach unten.
   Nach dem ersten Schritt sollte man eine zweidimensionale diskrete Matrix berechnet haben, deren Spaltenanzahl der Breite des Eingabebildes und deren Zeilenanzahl der Höhe des Eingabebildes entspricht. Die Einträge dieser Matrix sollen dann den Zugehörigkeitsgrad der Pixel zu einer Zelle beschreiben. In den Folgenden Schritten fungiert die errechnete Matrix als Eingabe und nicht mehr das Bild.

2. Im zweiten Schritt der Fuzzy Segmentation wird ein Filter über die Matrix gelegt. Der Filter setzt die Einträge auf Null, die in der Matrix mehr als vier benachbarte Einträge gleich Null haben. Es wird hier die achter Nachbarschaft vorausgesetzt, doch eine vierer Nachbarschaft ist genauso denkbar. Der zweite Schritt ist mit einem Rauschfilter zu vergleichen, der kleine Rauschartefakte beseitigt. Allerdings darf man von diesem Schritt nicht viel erwarten. Er dient lediglich zur groben Bildverbesserung.

3. Der dritte Schritt soll dafür sorgen, dass nicht nur der Rand der Zelle, sondern auch das Innere der Zelle segmentiert wird. Zu dem Zweck laufen von oben, unten, links und rechts Scanlinien über die Matrix. Trifft eine Scanline auf einen Matrixeintrag, der ungleich Null ist, dann wird von diesem Eintrag an weiter in die entsprechende Richtung gesucht. Wird ein weiterer Eintrag ungleich Null gefunden, so erhöhen sich die Zugehörigkeitswerte der Einträge, die zwischen den beiden gefundenen Einträgen liegen um einen Wert von 0.05, der heuristisch bestimmt ist. Wenn in die verfolgte Richtung kein weiterer Punkt gefunden wird, so bleiben die Einträge in der Matrix unverändert. Nachdem diese Scanlines von allen Seiten die Matrix durchlaufen haben, werden nur die Einträge in der Matrix größer als Null beibehalten, in denen sich durch die Überschneidung von mindestens zwei Scanlinien der Wert erhöht hat, das heißt in denen mindestens ein Wert von 0.1 steht. Alle anderen Einträge werden auf Null gesetzt. Nach dem dritten Schritt sollten dann die Lücken in der Matrix, die dem Inneren der Zellen entsprechen, zum Teil aufgefüllt sein.

4. Im vierten Schritt findet eine Mustererkennung statt, bei der kleine Untermatrizen der Matrix auf ein bestimmtes Besetzungsmuster getestet werden. Es liegt ein Regelwerk vor, dass bestimmt, bei welcher Besetzung der Untermatrix bestimmte Einträge in der Untermatrix erhöht, oder erniedrigt werden müssen. Das Regelwerk ergibt sich hier aus dem Wissen, wie ein Zellrand aussieht und dem Konstruktionsverfahren für die Matrix. Ein Regelwerk besteht hauptsächlich aus „WENN ... DANN ... “ Regeln. Es ist hier genau angegeben unter welchen Umständen ein Punkt zum Zellrand gehört und unter welchen Umständen nicht. In der Matrix ist festgehalten, zu welchem Grad die einzelnen Pixel zu einer Zelle gehören. Beispielsweise werden um einen Eintrag in der Matrix alle benachbarten Werte auf den Fuzzy Wert überprüft. Wenn alle benachbarte Werte unter einem bestimmten Wert liegen, dann wird der untersuchte Wert auf Null gesetzt. Es sind so Regeln entworfen worden, die zum Beispiel alleinstehende Werte eliminieren, oder Lücken auffüllen. Die Zuordnung erfolgt ähnlich den Regeln in einem Expertensystem, wie es in der Fuzzy-Logik bekannt ist [BKI03]. Es geht dabei im Wesentlichen darum aus dem bisherigen unscharfen Wissen mit Hilfe der Regeln Schlüsse auf die Zugehörigkeit der Pixel zu ziehen.

5. Der fünfte Schritt des Algorithmus dient dazu, zusammenhängende Gebiete in der Matrix zu erkennen und diese dann zu nummerieren. Der Teil ist als „(Sequential) Connected Component Algorithmus“ bekannt [GW02]. Man geht in diesem Schritt davon aus, dass die Einträge in der Matrix, die ungleich Null sind, tatsächlich Zellen sind und ordnet die Einträge entsprechend der achter Nachbarschaft einer Gruppe zu. Die Eingabe ist also theoretisch eine Matrix, in der genau an den Stellen Werte größer als Null stehen, wenn der an der Stelle zugehörige Pixel ein Teil einer Zelle ist.

   Der Algorithmus bestimmt ein ganzzahliges Bild, bei dem jedes zusammenhängende Gebiet mit einer anderen natürlichen Zahl (Etikette) markiert ist. Die Lösung wird dadurch bestimmt, dass die Matrix zeilenweise abgearbeitet und eine Äquivalenzrelation von Etiketten mitgeführt wird. Der Algorithmus arbeitet mit der vierer Nachbarschaft, wie es in Algorithmus 1 beschrieben ist. Die $n \times m$ Matrix ist mit $\mathbf{P}$ bezeichnet.

Nach dem fünften Schritt ist die Fuzzy Segmentation beendet und es können die Einträge der Matrix mit der entsprechenden Etikette zurückgeliefert werden. Zusammenhängende Gebiete, die nicht groß genug sind, d.h. einen bestimmten Schwellenwert unterschreiten, können als Rauschen angenommen und damit in der Ausgabe ausgelassen werden. Theoretisch sollte die Anzahl der Zellen im Eingabebild mit der Anzahl zusammenhängender Gebiete übereinstimmen. Dies ist allerdings nur unter optimaler Kalibrierung der Parameter und einem wenig verrauschten Eingabebild der Fall. Die verschiedenen einstellbaren Schranken und die Auswahl der Verfahren im ersten Schritt sind für die Güte des Verfahrens verantwortlich. Sie sind empirisch bestimmt. Bei der

Algorithmus 1 Sequential Connected Component

```
1: for i = 1, ..., n do
2:   for j = 1, ..., m do
3:     if P[i, j] > 0 then
4:       if P[i, j − 1] > 0 then
5:         Übernimm Etikette von (i,j − 1) nach (i,j).
6:         if (P[i − 1, j] > 0) ∧ (Etikette k von (i,j) ≠ Etikette k′ von (i − 1,j)) then
7:           Vereinige Äquivalenzklassen von k und k′ in der Äquivalenzrelation.
8:         end if
9:       end if
10:      if P[i, j − 1] = 0 then
11:        if P[i − 1, j] > 0 then
12:          Ordne (i,j) die Etikette von (i − 1,j) zu.
13:        else
14:          Gib (i,j) eine neue Etikette, z.B. die kleinste bisher nicht vergebene natürliche Zahl.
15:        end if
16:      end if
17:    end if
18:  end for
19: end for
20: for i = 1, ..., n do
21:   for j = 1, ..., m do
22:     Ersetze die Etikette von (i,j) durch die kleinste Zahl in ihrer Äquivalenzklasse.
23:   end for
24: end for
```

Entwicklung von CellTrack wurden auch noch andere Verfahren zur Kantenerkennung getestet. Unter anderem wurde ein Gaußfilter mit anschließendem Sobeloperator angewendet und eine Fuzzy-Segmentierung hinzugenommen. Diese Versuche endeten allerdings in schlechteren Ergebnissen und werden deshalb hier nicht weiter beschrieben. Zusammenfassend kann man aber sagen, dass die Fuzzy Segmentation eine gute Grundlage für die Zusammenschaltung von verschiedenen Verfahren im Segmentierungsbereich bietet und nach Anpassung der Parameter an das Eingabebild sehr gute Ergebnisse liefert. Das Ergebnis ist in Abbildung 3.8 visualisiert. Auf der linken Seite der Abbildung sieht man den Bildausschnitt, der segmentiert werden soll. Auf der rechten Seite ist das Ergebnis der Segmentierung dargestellt, welches durch die rote Färbung der segmentierten Gebiete verdeutlicht wird.

3.4 Tracking

Tracking bezeichnet das Verfolgen von Bewegung in Videos. Dies war der wichtigste Abschnitt im CellTrack-Projekt, da das Tracking der Zellbewegungen die zentrale Zielsetzung war. Um die Zellbewegungen verfolgen zu können, wurden mehrere Verfahren implementiert und getestet, in wie weit sie erfolgreich eingesetzt werden konnten. Dabei war es wichtig, dass jedes Verfahren auf allen zur Verfügung stehenden Videos brauchbare Ergebnisse lieferte. Durch die Verwendung mehrerer Verfahren, konnten die einzelnen Ergebnisse kombiniert und somit das Gesamtergebniss des Trackings erheblich verbessert werden.

3.4.1 Snakes

Die Snake wurde bereits im Grundlagenkapitel sowohl für ihren Einsatz für das Segmentieren (vgl. Kapitel 2.5.4), als auch für die Anwendung für das Tracking skizziert (vgl. Kapitel 2.6.4). Nachdem

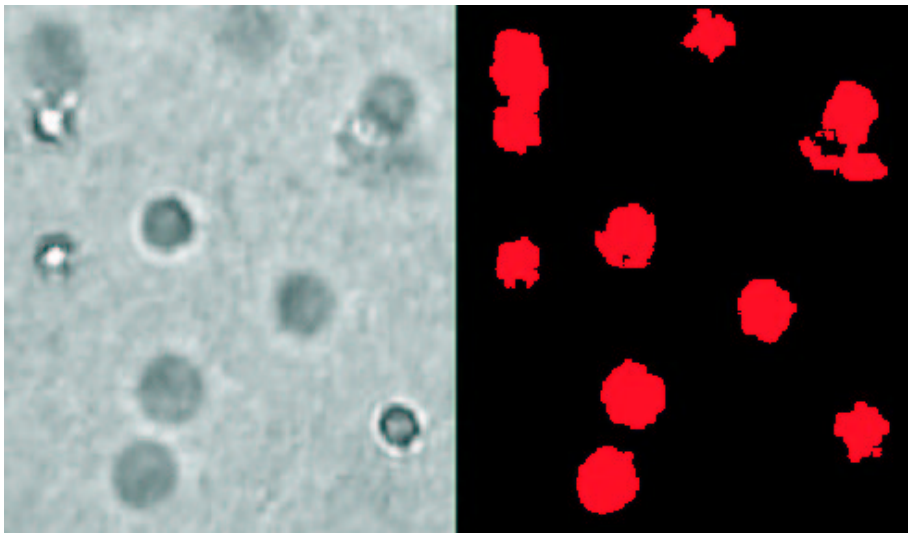


Abbildung 3.8: Ergebnis der Fuzzy Segmentation

in CellTrack das Blockvergleichverfahren implementiert worden war, erschien es sinnvoll, einmal einen komplexeren Ansatz zu verfolgen. Da die Snake in der Literatur anscheinend immer gute Ergebnisse liefert, blieb die Hoffnung, dass die Snake auch für CellTrack gute Ergebnisse liefern könnte.
Im nächsten Unterkapitel soll die Snake vom theoretischen Aspekt genau unter die Lupe genommen werden.

Energiegleichung der Snake

Die Snake ist durch eine Energiegleichung beschrieben, wie in Kapitel 2.5.4 vorgestellt wurde. Die Energien und ihre Modifikationen werden in den nächsten Abschnitten genauer beschrieben. Die Snake ist eine Kurve $v(s,t)$, welche aus einer endlichen Anzahl von n Knoten besteht, die Snaxel genannt werden. Die Kurve wird zum Zeitpunkt t beschrieben. Die Energiegleichung ist kontinuierlich und muss noch diskretisiert werden, um eine anwendbare Lösung herbeizuführen. Für die diskrete Lösung wird die Snake durch Snaxel auf einem diskreten Gitter angegeben. Eine genaue Herleitung steht in [LL93], aber auch [MKT87] beschreibt im Anhang eine numerische Methode zur Lösung der Energiegleichung. Weitere Informationen zu der theoretischen Löung kann man in [Ker03] nachlesen. Die wichtigsten Schritte können im Folgenden nachvollzogen werden. Die Variationsrechnung wird bei der Lösung als Hilfsmittel herangezogen. Das allgemeine Variationsproblem lautet nach [KM01]:

$$I_F = \int_a^b F(x, y(x), y'(x), ..., y^n(x))dx \stackrel{!}{=} Minimum \tag{3.3}$$

für das Intervall $[a,b]$ und der Funktion F

In unserem Fall ist eine Kurve $\mathbf{v}(s)$ zu bestimmen, so dass:

$$I_F = \int_a^b F(s, v, v_s, v_{ss})ds \stackrel{!}{=} Minimun \tag{3.4}$$

für das Zeitintervall $[a,b]$.

Ein Variationsproblem dieser Art kann mit der Euler-Lagrange Gleichung gelöst werden:

$$F_v - \frac{\partial}{\partial s} F_{v_s} + \frac{\partial^2}{\partial s^2} F_{v_{ss}} = 0. \tag{3.5}$$

Angenommen $\alpha(s)$ und $\beta(s)$ sind Konstanten, dann löst die folgende Gleichung die Energiegleichung:

$$\frac{\partial Q}{\partial \mathbf{v}} - \alpha \mathbf{v}_{ss} + \beta v_{ssss} = 0. \tag{3.6}$$

Anders aufgeschrieben sind das zwei unabhängige Gleichungen:

$$-\alpha \frac{\partial^2 x}{\partial s^2} + \beta \frac{\partial^4 x}{\partial s^4} = -\frac{\partial Q}{\partial x} \tag{3.7}$$

$$-\alpha \frac{\partial^2 y}{\partial s^2} + \beta \frac{\partial^4 y}{\partial s^4} = -\frac{\partial Q}{\partial y} \tag{3.8}$$

In die diskrete Energiegleichung $E(i)^* = S(i)+P(i)$ müssen nun die diskreten Eulergleichungen einfließen. Die Eulergleichungen enthalten Differentialquotienten, welche über Differenzenquotienten angenähert werden können. Diese Annährung ist unter dem Stichwort der „Finiten Differenzen“ bekannt. Dabei berechnet man die erste bis vierte Ableitung wie folgt:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{3.9}$$

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{3.10}$$

$$f'''(x) \approx \frac{f(x+2h) - 3f(x+h) + 3f(x) - f(x-h)}{h^3} \tag{3.11}$$

$$f''''(x) \approx \frac{f(x+2h) - 4f(x+h) + 6f(x) - 4f(x-h) + f(x-2h)}{h^4} \tag{3.12}$$

Benutzt man nun die finiten Differenzen für die zweite und vierte Ableitung eines x_i, so erhält man folgende Gleichungen:

$$\frac{\partial^2 x_i}{\partial s^2} = \frac{1}{h^2}(x_{i-1} - 2x_i + x_{i+1}) \tag{3.13}$$

$$\frac{\partial^4 x_i}{\partial s^4} = \frac{1}{h^4}(x_{i-2} - 4x_{i-1} + 6x_i - 4x_{i+1} + x_{i+2}) \tag{3.14}$$

Es ergibt sich eine diskrete Gleichung der Eulergleichung, bei der Annahme von äquidistanten Punkten (d.h. h fällt heraus):

$$-\alpha(x_{i-1} - 2x_i + x_{i+1}) + \beta(x_{i-2} - 4x_{i-1} + 6x_i - 4x_{i+1} + x_{i+2}) = -\frac{\partial Q}{\partial x} = -f_x^i \tag{3.15}$$

$$-\alpha(y_{i-1} - 2y_i + y_{i+1}) + \beta(y_{i-2} - 4y_{i-1} + 6y_i - 4y_{i+1} + y_{i+2}) = -\frac{\partial Q}{\partial y} = -f_y^i \tag{3.16}$$

Diese Gleichungen können in Matrixform umgeschrieben werden.

$$\mathbf{A}\mathbf{x} = \mathbf{f}_x(\mathbf{x}, \mathbf{y}) \tag{3.17}$$

$$\mathbf{A}\mathbf{y} = \mathbf{f}_y(\mathbf{x}, \mathbf{y}) \tag{3.18}$$

Die Matrix $\mathbf{A}$ ist eine symmetrische, pentagonale und zyklische Bandmatrix und besitzt soviele Spalten bzw. Zeilen, wie es Snaxel gibt:

$$\begin{pmatrix}
2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & \dots & \dots & 0 & \beta & -\alpha-4\beta \\
-\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & \dots & \dots & 0 & \beta \\
\beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & \dots & \dots & 0 \\
0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & \dots & 0 \\
\vdots & & \ddots & \ddots & \ddots & \dots & \ddots & & \\
\vdots & & & \ddots & \ddots & \dots & \ddots & \ddots & \\
0 & \dots & \dots & 0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta \\
\beta & 0 & \dots & \dots & 0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta \\
-\alpha-4\beta & \beta & 0 & \dots & \dots & 0 & \beta & -\alpha-4\beta & 2\alpha+6\beta
\end{pmatrix}$$

Die internen Kräfte sind komplett in der Bandmatrix spezifiziert. Das Gleichungssystem kann iterativ gelöst werden, indem man den Vektor der Snaxel $\mathbf{v}(s)$ als veränderlich mit der Zeit t ansetzt, also $\mathbf{v}(s,t)$. Dies führt zu:

$$-\frac{\partial \mathbf{v}}{\partial t} = \mathbf{A}\mathbf{v} - \mathbf{f}_{\mathbf{v}}(\mathbf{x}_t, \mathbf{y}_t) \tag{3.19}$$

$$\frac{\partial \mathbf{v}}{\partial t} = \frac{\mathbf{v}_t - \mathbf{v}_{t-1}}{\Delta t} \tag{3.20}$$

Normalerweise bezeichnet man mit Δt die Zeitschrittweite des Verfahrens. Mit $\gamma = \frac{1}{\Delta t}$ und der Annahme, dass sich die externen Terme innerhalb eines Zeitschritts nicht ändern (also $\mathbf{f}_{\mathbf{v}}(\mathbf{v}_t) \approx \mathbf{f}(\mathbf{v}_{t-1})$), ergibt sich die Gleichung:

$$\mathbf{A}\mathbf{v}_t - \mathbf{f}_v(\mathbf{x}_{t-1}, \mathbf{y}_{t-1}) = -\gamma(\mathbf{v}_t - \mathbf{v}_{t-1}) \tag{3.21}$$

Die Gleichung ist nun leicht nach $\mathbf{v}_t$ aufzulösen, falls $\mathbf{A}$ invertierbar ist, so dass man die Gleichung 3.22 als iterative Lösung erhält.

$$\mathbf{x}_t = (\mathbf{A} + \gamma\mathbf{I})^{-1}(\gamma\mathbf{x}_{t-1} - \mathbf{f}_x(\mathbf{x}_{t-1}, \mathbf{y}_{t-1})) \tag{3.22}$$

$$\mathbf{y}_t = (\mathbf{A} + \gamma\mathbf{I})^{-1}(\gamma\mathbf{y}_{t-1} - \mathbf{f}_y(\mathbf{x}_{t-1}, \mathbf{y}_{t-1})) \tag{3.23}$$

In Gleichung 3.22 steht $\mathbf{I}$ für die Einheitsmatrix der Größe $n \times n$. Es sei darauf hingewiesen, dass der Term $(\mathbf{A} + \gamma\mathbf{I})^{-1}$ vorberechnet werden kann. In CellTrack wurden diese Gleichungen als Grundlage für die Berechnung der minimalen Energie benutzt.

Modellerweiterung: Verstrebungen

In CellTrack sind die Energien noch erweitert worden. Grund der Erweiterung war, dass das Standard-Snakemodell in der obigen Form nicht akzeptabel funktionierte. Die Kontur der Snake fiel oft in sich zusammen. Eine zusammengefallene Snake konnte sich nach dem Grundmodell nicht mehr aufblähen. Der Zusammenfall der Snake wurde deshalb als Abbruchkriterium für die Snake eingeführt. Hier sollten die anderen Verfahren oder der Benutzer das Tracking weiterführen. Aufgabe war es, die Häufigkeit des Zusammenfalls der Snake zu verringern. Es entstanden Überlegungen allein durch eine bessere externe Energie den Zusammenfall in den Griff zu bekommen, doch der Erfolg war nur gering und war zu sehr von den Videodaten abhängig.
Eine neue Idee leistete eine starke Verbesserung der Snake. Es sollten Verstrebungen in die Snake eingebaut werden, um diese stabiler zu gestalten.

Eine Verstrebung ist eine Verbindung zweier Snaxel $s_i = (x_i, y_i)$ und $s_j = (x_j, y_j)$. Die beiden verbundenen Snaxel haben einen bestimmten Abstand zueinander $x_i - x_j = d_x$ und $y_i - y_j = d_y$. Integriert man die Verstrebungen in das bestehende Snakemodell, so kann man die Abstandsbeziehungen aller Snaxel in einem Gleichungssystem beschreiben. Im Groben soll das Gleichungssystem wieder in Standardform $\mathbf{Bx} = \mathbf{d}$ sein. Das Gleichungssystem beschreibt die Verstrebungen für n Snaxel, wobei die $\frac{n}{2}$te Spalte und die $\frac{n}{2}$te Zeile mit Nullen besetzt sind. Eine detaillierte Sicht auf das Gleichungssystem liefert Gleichung 3.24. Es sei angemerkt, dass die Verstrebungen auch anders, als in Matrix $\mathbf{B}$ festgelegt werden können. Die hier vorgestellte Variante legt $\mathbf{B}$ so fest,

dass die Distanzen der Verstrebungen möglichst hoch sind. Wenn die Kontur z.B. ein Kreis ist, dann liegen die beiden Punkte, die die Verstrebung ausmachen, direkt gegenüber. In einem anderen Ansatz ist es beispielsweise möglich die Konstruktion der Verstrebungen mit der externen Energie, also mit dem Gradientenbild in Verbindung zu bringen. Dieses Detail wurde in CellTrack nicht umgesetzt.

$$\epsilon\mathbf{Bx} = \epsilon\mathbf{d} \Leftrightarrow \epsilon \begin{pmatrix} 1 & 0 & \dots & & 0 & -1 & 0 & \dots & 0 \\ 0 & 1 & & & & 0 & -1 & & \vdots \\ \vdots & & \ddots & & \vdots & & 0 & \ddots & 0 \\ & & & 1 & & & & & -1 \\ 0 & \dots & & & 0 & & \dots & & 0 \\ -1 & & & & & 1 & & & \vdots \\ 0 & -1 & & & \vdots & & \ddots & & \\ \vdots & & \ddots & 0 & & & & 1 & 0 \\ 0 & \dots & & -1 & 0 & & \dots & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \epsilon \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix} \tag{3.24}$$

Das Gleichungssystem wird analog für die y-Richtung aufgestellt. Danach wird $\epsilon\mathbf{B}$ auf $\mathbf{A}$ und $\epsilon\mathbf{d}$ auf $\mathbf{f}$ aufsummiert. Eine besondere Bedeutung kommt dem Parameter ϵ zu, der den Einfluss der neuen Energie im Modell steuert. ϵ ist ein Penalty-Faktor, der in diesem Fall die Steifigkeit der Strebe reguliert. Ein hoher Wert sorgt dafür, dass die Kontur wenig elastisch (steif) ist, während ein niedriger Wert die Elastizität der Kontur erhöht. Das erweiterte Modell generiert eine Snake, die nicht mehr so leicht zusammenfällt und auch beim Tracking nicht so schnell auf andere Zellen überspringt, wie die Snake des ursprünglichen Modells.

Modellerweiterung: Externe Energie

Normalerweise arbeitet die Snake mit einer externen Energie, die das gaußgefilterte Gradientenbild und die Bildintensität zugrunde legt, wie man in Formel 3.25 sieht.

$$\begin{aligned} P(\mathbf{v}) &= \int_0^1 Q(\mathbf{v}(s))ds \\ &= \int_0^1 E_{intensity} + E_{edge}ds \\ &= \int_0^1 \pm I(x,y) - |\nabla G_{sigma} * *I(x,y)|^2 ds \end{aligned} \tag{3.25}$$

Die Bildintensität wird in CellTrack nicht in die externe Energie eingebracht. Dafür wird statt der Bildintensität eine Segmentierungsenergie hinzugezogen. Gemeint ist damit, dass man eine Segmentierung durchführt und dann an den Stellen die externe Energie erhöht, an denen die Segmentierung eine Zelle erkannt hat. Benutzt wird die Fuzzy Segmentation, um diesen Zusatz für die externe Energie zu bestimmen (vgl. Kapitel 3.3.2). In der segmentierten Umgebung wird nur der größte zusammenhängende Bereich als Zelle angenommen. Das funktioniert deshalb, weil die Umgebung, in der segmentiert wird, nicht viel größer als eine Zelle ist. Probleme gibt es, wenn sich Zellen in dem Bereich überlappen oder teilweise nicht mehr in der Umgebung liegen. Es kann dann passieren, dass die Snake leicht überspringt.
In CellTrack wurden auch andere Energien getestet. Es wurde das Verfahren *Gradient Vector Flow* zum Testen in das Modell eingebaut [XP98]. Der Ansatz lieferte allerdings keine besseren Ergebnisse und benötigte eine höhere Laufzeit. Deshalb wurde dieser Ansatz verworfen.
Weitere getestete Energien kamen aus einem Artikel, der das Tracken von Leukozyten behandelt [NRL02]. Es wird beschrieben, wie die Form von Zellen in das Energiemodell hinzugezogen werden kann. Leukozyten besitzen eine elliptische Form und variieren im Laufe der Zeit nicht stark in der

Abbildung 3.9: Initiale Markierung für die Snake

Größe oder der Form. Da die Zellen in CellTrack in vielen Fällen stark in ihrer Form schwanken, wurde auch dieser Ansatz verworfen.

3.4.2 Umsetzung der Snake

Grundlage ist, dass der Benutzer einen rechteckigen Bereich im Bild markiert hat, in welchem sich die zu trackende Zelle befindet. Die Snake soll dann im ersten Frame initial gesetzt werden und sich dann über die Zeit mit der Zelle mitbewegen.

Schon in den vorigen Kapiteln über die Snakes wurde der theoretische Hintergrund der Snake beschrieben. Es soll nun in diesem Abschnitt beschreiben werden, welche Klassen in CellTrack die Snake implementieren. Die Implementierung der Snake verbirgt sich in CellTrack in den Klassen `CMultiFeatureTracking` und in `CExtEnergy`. Die Beziehung der beiden Klassen sind in Abbildung 4.7 verdeutlicht. Die wichtigste Memberfunktion von `CMultiFeatureTracking` ist *snakeTracking*. Als Eingabe bekommt die Methode unter anderem einen Zeiger auf einen Bildausschnitt, in dem eine Zelle enthalten ist, und einen Zeiger auf `CCellList`. Der Bildausschnitt könnte z.B. wie in Abbildung 3.9 gewählt werden.

Die Klasse `CExtEnergy` beinhaltet Methoden, wie z.B. Filter, die für die Berechnung der externen Energie der Snake benötigt werden. Ein Objekt der Klasse wird in `CMultiFeatureTracking` erzeugt, um die externen Energien in x-Richtung und y-Richtung zu berechnen. Dies funktioniert durch den Aufruf von *computeEnergy*. Nach dem beenden von *snakeTracking* soll die Zellliste um die Trackinginformationen der Snake für den neuen Frame erweitert werden. Für die Zellliste bedeutet das, dass ein `CSnakeContainer` eingefügt werden muss. Der Algorithmus 2 beschreibt grob die Arbeitsweise der Snake. Verschiedene Schritte, wie die Berechnung der Initialkontur, der Bounding Boxen und des Vertrauenswerts werden in den nächsten Unterkapiteln näher erläutert. Nachdem der Algorithmus 2 durchgelaufen ist, ergibt sich z.B. für das Eingabebild aus Abbildung 3.9 eine Snakekontur, wie sie in Abbildung 3.10 zu sehen ist.

Algorithmus 2 Snake

1: t: aktuell betrachteter Frame
2: α: 2 (Spannung)
3: β: 3 (Steifigkeit)
4: γ: 1 (Schrittweite)
5: ϵ: 400 (Elastizität)
6: **if** t ist erster Frame in der Trackingsequenz **then**
7: Initialkontur erstellen.
8: Iterationen der Snake: $n = 100$
9: **else**
10: Kontur von Frame $t-1$ aus `CCellList` holen.
11: Iterationen der Snake: $n = 10$
12: **end if**
13: Fläche und Schwerpunkt der Snake berechnen.
14: Externe Energien mit Hilfe von `CExtEnergy` bestimmen und in $\mathbf{f_x}$ und $\mathbf{f_y}$ ablegen.
15: $\mathbf{CA} = (\mathbf{A} + \gamma\mathbf{I})^{-1}$ berechnen (unter Berücksichtigung von α, β und ϵ).
16: **for** $i = 1, ..., n$ **do**
17: **if** t ist nicht erster Frame in der Trackingsequenz **then**
18: Externe Energie der Verstrebungen unter Berücksichtigung von ϵ in das Snakemodell einbauen.
19: **end if**
20: $\mathbf{kontur}_{i,x} = \mathbf{CA} * ((\gamma * \mathbf{kontur}_{i-1,x}) - \mathbf{f_x})$
21: $\mathbf{kontur}_{i,y} = \mathbf{CA} * ((\gamma * \mathbf{kontur}_{i-1,y}) - \mathbf{f_y})$
22: **end for**
23: Rechteckige Bounding Box um die Snake berechnen.
24: Vertrauenswert $r \in [0, 1]$ mit Hilfe der Fläche berechnen.
25: `CSnakecontainer` in der `CCellList` für Frame t abspeichern.

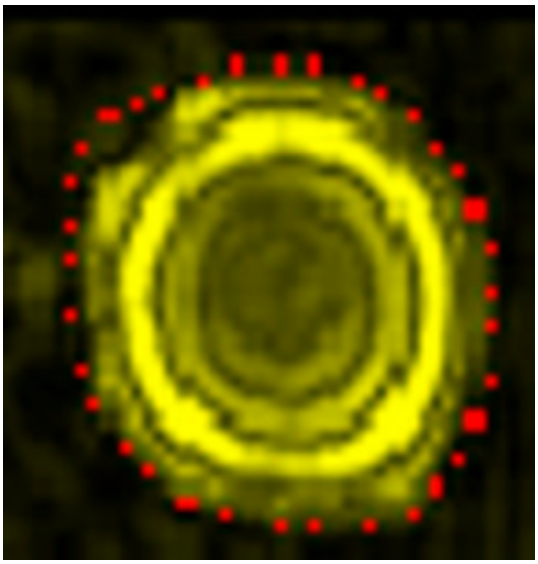


Abbildung 3.10: Snakekontur

Parameter

Die Benennung der Parameter α, β, γ und ϵ sind äquivalent zu der Benennung in den vorigen Kapiteln. Die zugeordneten Werte sind empirisch bestimmt, d.h. nach vielen Tests mit CellTrack und verschiedenen Videos haben sich die Werte als die Besten herauskristallisiert. Es ist bekannt, dass man die Iteration beenden kann, wenn sich die Snake von einer auf die andere Iteration so gut wie gar nicht mehr ändert. In CellTrack wurde dies nicht berücksichtigt, sondern eine statische Anzahl von n Iterationen festgelegt.

Initialkontur

Im ersten Frame einer Trackingsequenz muss eine Initialkontur berechnet werden, die später durch die Iterationen verbessert werden soll. In CellTrack wird von dem Bildausschnitt, der dem Snakealgorithmus übergeben wird, zunächst eine Segmentierung durchgeführt. Wenn die Segmentierung einen segmentierten Bereich zurückliefert, dann wird auf diesen Bereich ein Dilationsoperator angewandt. Es wird dann der Rand des segmentierten dilatierten Bereiches als Initialkontur benutzt.

Im Algorithmus 3 wird beschrieben, wie man mit Hilfe eines Stacks aus dem segmentierten dilatierten Bild die initiale Snakekontour erhält. Dabei ist ein Pixel im Bildausschnitt als segmentiert gekennzeichnet, wenn im zweidimensionalen Array **segment**[.][.] eine Eins eingetragen ist. In CellTrack wurde die Fuzzy Segmentierung benutzt. Wenn der Segmentierer kein Ergebnis zurückliefert, wie es vor allem bei sehr kleinen Bildausschnitten der Fall ist, dann wird der Rahmen des Bildausschnitts als Initialkontur angenommen.
Im Algorithmus 3 taucht der Parameter *pixeldump* auf. Mit ihm kann gesteuert werden, in welchen Abständen Pixel in die Kontur aufgenommen werden. Wird der Parameter auf Eins gesetzt, so wird jeder Pixel, der sich auf dem Rand befindet, zur Initialkontur hinzugefügt. Nach mehreren Versuchen stellte sich heraus, dass vier ein guter Wert ist. Wählt man den Parameterwert höher, so sind oftmals zu wenig Snaxel vorhanden. Das führt dann dazu, dass die Snake schneller zusammenfällt. Die Anzahl der Snaxel ist auch direkt für die Laufzeit des Verfahrens verantwortlich, da die Dimension der benutzten Matrizen mit der Anzahl der Snaxel übereinstimmt.

Algorithmus 3 Initialkontur

```
1: pixeldump = 4, d.h. nur jeder vierte Pixel wird zur Kontur hinzugefuegt
2: for i = 0, ..., segment.breite do
3:    for j = 0, ..., segment.hoehe do
4:       if segment[i, j] = 1 then
5:          Start.x = i
6:          Start.y = j
7:          BREAK
8:       end if
9:    end for
10: end for
11: stack.push(Start)
12: while !stack.empty() do
13:    ActPoint = stack.top()
14:    stack.pop()
15:    Setze segment[ActPoint.x, ActPoint.y] = 0 (als betrachtet markieren)
16:    if i MOD pixeldump = 0 then
17:       Füge ActPoint zu der Kontur hinzu.
18:    end if
19:    i = i + 1
20:    if segment[ActPoint.x, ActPoint.y − 1] = 1 then
21:       stack.push((ActPoint.x, ActPoint.y − 1))
22:    end if
23:    if segment[ActPoint.x + 1, ActPoint.y − 1] = 1 then
24:       stack.push((ActPoint.x + 1, ActPoint.y − 1))
25:    end if
26:    if segment[ActPoint.x + 1, ActPoint.y] = 1 then
27:       stack.push((ActPoint.x + 1, ActPoint.y))
28:    end if
29:    if segment[ActPoint.x + 1, ActPoint.y + 1] = 1 then
30:       stack.push((ActPoint.x + 1, ActPoint.y + 1))
31:    end if
32:    if segment[ActPoint.x, ActPoint.y + 1] = 1 then
33:       stack.push((ActPoint.x, ActPoint.y + 1))
34:    end if
35:    if segment[ActPoint.x − 1, ActPoint.y + 1] = 1 then
36:       stack.push((ActPoint.x − 1, ActPoint.y + 1))
37:    end if
38:    if segment[ActPoint.x − 1, ActPoint.y] = 1 then
39:       stack.push((ActPoint.x − 1, ActPoint.y))
40:    end if
41:    if segment[ActPoint.x − 1, ActPoint.y − 1] = 1 then
42:       stack.push((ActPoint.x − 1, ActPoint.y − 1))
43:    end if
44: end while
```

Bounding Box

In Algorithmus 2 wird nach der Berechnung der neuen Snaxelpositionen, also nach den n Iterationen, eine Bounding Box der Kontur bestimmt. Um zu verstehen, warum dies geschieht, muss man einen Blick auf den gesamten Trackingprozess werfen. Im ersten Frame des Trackingprozesses markiert der Benutzer einen rechteckigen Bereich um die Zelle herum. Auf diesem Bereich arbeitet der Algorithmus 2. Wenn sich nun die Kontur der Snake bewegt, dann muss sich auch dieser rechteckige Bereich bewegen, um für das nächste Frame bzw. den nächsten Trackingschritt wieder eine vernünftige Eingabe zu liefern. Der alte rechteckige Bereich ist schließlich nach dem verschieben der Kontur nicht mehr gültig.

Deshalb wird von der Kontur eine rechteckige Bounding Box erstellt. Diese Bounding Box wird zu allen Seiten hin um fünf Pixel erweitert. Der Wert von fünf Pixeln rechtfertigt sich dadurch, dass in den Versuchen in CellTrack keine Zellen beobachtet wurden, die sich von einem Frame auf den anderen um mehr als fünf Pixel in eine Richtung bewegten. Wenn es denn trotzdem einmal vorkommen würde, dass eine Zelle über mehrere Frames so schnell wäre, dann würde die Snake die Zelle auf jedem Fall wegen der Bounding Box verlieren. Der Grund dafür, dass man die Bounding Box nicht um mehr als fünf Pixel erweitert ist, dass bei einem größeren betrachteten Bereich die Wahrscheinlichkeit steigt, dass die Snake auf andere Zellen überspringt.

Vertrauenswert

Der Vertrauenswert der Snake soll angeben, wieviel Vertrauen man in das Ergebnis der Snake hat. Der Vertrauenswert stützt sich auf die Veränderung der Fläche, die von der Snakekontur eingeschlossen wird. Es wurden hier Erfahrungswerte eingesetzt. Wenn die Fläche kleiner als 50 Flächeneinheiten oder größer als 1900 Flächeneinheiten ist, dann wird der Vertrauenswert der Snake auf Null gesetzt, d.h. hier hat die Snake versagt. Man kann hier von einem absoluten Fehler sprechen. Der relative Fehler, der zu einem Vertrauenswert von 0.5 führt, entsteht dadurch, dass man die Flächen von zwei Zeitschritten miteinander vergleicht. Wenn die Fläche sich um mehr als den Faktor 1.2 vergrößert oder sie sich um weniger als den Faktor 0.8 verkleinert hat, dann wird ein Vertrauenswert von 0.5 angenommen. In CellTrack sind dies die einzigen Kriterien, die den Vertauenswert ausmachen. Es sei bemerkt, dass die Berechnung des Vertrauenswertes sich noch auf andere Maßzahlen stützen könnte, wie z.B. den Umfang. Je mehr Maßzahlen zur Verfügung stehen, desto besser kann man den Vertrauenswert beschreiben. Im Rahmen von CellTrack ist es allerdings nicht gelungen, aus dem gegebenen Videomaterial und den berechneten Maßzahlen bestimmte Regeln zur Bestimmung des Vertrauenswertes anzugeben. Es zeigte sich, dass die auftretenden Fälle zu unterschiedlich waren, um sie eindeutig zu klassifizieren. Das Modell, dass sich alleine auf die Fläche bezieht, lieferte schon annehmbare Ergebnisse, so dass hier nicht weiter nachgeforscht wurde.

3.4.3 Blockvergleichverfahren

Das Blockvergleichverfahren wurde nach dem Prinzip, das im Kapitel Grundlagen beschrieben wurde (vgl. Kapitel 2.6.5), implementiert. Der wesentliche Teil des Blockvergleichverfahrens besteht aus einer Kreuzkorrelation, aus der eine maximale Übereinstimmung zweier Bildbereiche bestimmt wird. Da es mehrere Punkte maximaler Übereinstimmung geben kann, werden zuerst alle Punkte in einem Array gespeichert. Dieses wird, nachdem die beiden Bilder vollständig durchlaufen sind, ausgewertet, um letztlich den Punkt minimalster Abweichen des Featurebildes (Bildausschnitt mit der Zelle, die wiedergefunden werden soll) vom umgebenden Bild zu ermitteln.
Schon die ersten Tests mit dem Blockvergleichverfahren zeigten, dass selbst das Wiederfinden eines kleinen Bildes in einem großen umgebenden Bild sehr ressourcen- und rechenzeitaufwändig ist. Deshalb fiel früh die Entscheidung, den Suchbereich (umgebender Bereich, in dem das Featurebild wiedergefunden werden soll) möglichst klein zu halten. Um dies zu erreichen, wurde ein Verfahren implementiert, das überprüft, ob es außerhalb des Featurebildes Bewegung gibt, die der Zelle im Featurebild zugeordnet werden kann. So wird der Suchbereich um das Featurebild nur soweit

vergrößert, dass er alle Bewegung in der Nähe mit einschließt. Um die Bewegung zu detektieren, wird zeilen- und spaltenweise ein Differenzbild aus dem aktuellen und dem vorhergehenden Bild bestimmt. Zeigen die Grauwertänderungen der Pixel eine Bewegung an, d. h. liegen sie über einem bestimmten Schwellenwert, so wird die entsprechende Zeile (oder Spalte) zur Größe des Featurebildes hinzugefügt. Dies geschieht so lange, bis eine Zeile (oder Spalte) keine Bewegung mehr aufweist. Die dadurch entstandenen Koordinaten des Suchbereichs werden dann, zusammen mit den Koordinaten des Featurebildes, in einen `ImageContainer` (vgl. Kapitel 4.2.3) geschrieben. Als Bezugsgröße für die Koordianten wurde das originale Videobild mit den Maßen 720×576 Pixel gewählt, um die Positionen im Bild konsistent zu halten und unnötige Umrechnungen zu den Ergebnissen anderer Verfahren zu vermeiden.
Darüber hinaus wurde bald ein weiteres Problem des Blockvergleichverfahrens, das Rechtecke zum Vergleich verwendet, deutlich. Da die zu trackenden Zellen meistens eine rundlich Form aufweisen, erweist sich eine rechteckige Markierung der Zelle als schwierig. Je nachdem wie genau eine Zelle durch ein Rechteck markiert wird, bzw. wie viel Hintergrund mit markiert wird, entscheidet darüber, ob das Tracking mit dem Blockvergleichverfahren erfolgreich war oder nicht. Wird der Rahmen um eine Zelle zu groß gewählt, erhält der Hintergrund bei der Kreuzkorrelation ein zu großes Gewicht gegenüber der Zelle. So kann es leicht passieren, dass nicht die Zelle getrackt wird sondern der Hintergrund und ein Rauswandern der Zelle aus dem Featurebild kaum algorithmisch festgestellt werden kann. Um dies zu verhindern, muss auch der Featurebereich möglichst klein sein, die Zelle aber noch vollständig umschließen. Deshalb wurde entschieden, nach einem erfolgreichen Tracking den Suchbereich zu segmentieren. Als erstes Verfahren wurde hierfür das Bereichswachstumsverfahren (vgl. Kapitel 3.3.1) verwendet. Als Ausgangspunkt für das Bereichswachstumsverfahren wurde der Mittelpunkt des gefundenen Featurebildes gewählt. Dieser liegt innerhalb der Zelle, wodurch das Bereichswachstumsverfahren eine Segmentierung der Zelle liefert, aus der dann das Featurebild gewonnen werden kann.
Nachdem auch die Tracking-Verfahren implementiert wurden und sich zeigte, dass es erhebliche Laufzeitprobleme des Gesamtprogramms gibt, wurde auf eine Segementierung zur Verkleinerung des Featurebildes zugunsten der Laufzeit verzichtet. Da aber trotzdem ein möglichst kleines Featurebild erforderlich ist, wird auf das Featurebild der `Snake` zurückgegriffen, die dieses ebenfalls durch einen Segementierer berechnet (vgl. Kapitel 3.4.1).

3.4.4 Level-Set

Level-Set wurde für das Projekt CellTrack als ein mögliches Verfahren ausgewählt, da verschiedene Paper bei ähnlichen Bildern, wie sie von den Zellen vorliegen, sehr gute Ergebnisse für ein etwaiges Segmentieren oder Tracken versprachen [Was04], [MRA04], [VC02]. Besonders bei nicht sehr kontrastreichen Kanten bietet Level-Set eine gute Erkennung. Für das Tracken bestand die Idee, das Verfahren für jedes Bild und jede Zelle mit der zuletzt gefundenen Kontur erneut aufzurufen. Auch eine Segmentierung schien über dieses Verfahren realisierbar, wurde aber später nicht weiter verfolgt, da dies schon aus Zeitgründen nicht mehr innerhalb des Projektes hätte verwirklicht werden können.

Das Level-Set Verfahren wurde 1988 von den Mathematikern J.A. Sethian und S. Osher vorgestellt. Level-Set ist im allgemeinen eine numerische Technik, die entworfen wurde um die Entwicklung von Kanten zu verfolgen unter Zuhilfenahme einer geometrischen Beschreibung der Kontur. Dabei wird das zweidimensionale Problem in ein Dreidimensionales transformiert. Die meisten numerischen Techniken versuchen, bewegte Kanten mit Hilfe einer Menge von Marker-Punkten auf der Kante zu verfolgen. Die Position der Punkte wird korrespondierend zu der sich bewegenden Kante verändert. Dieser Ansatz enthält entsprechend eine Menge Probleme, die J.A. Sethian mit der Level-Set und Fast Marching Methode versucht hat zu umgehen. Statt den Fokus auf die sich bewegenden Kanten zu legen, hat er eine starke mathematische Verbindung zwischen den sich verändernden Kanten und Gleichungen gefunden [Set99].

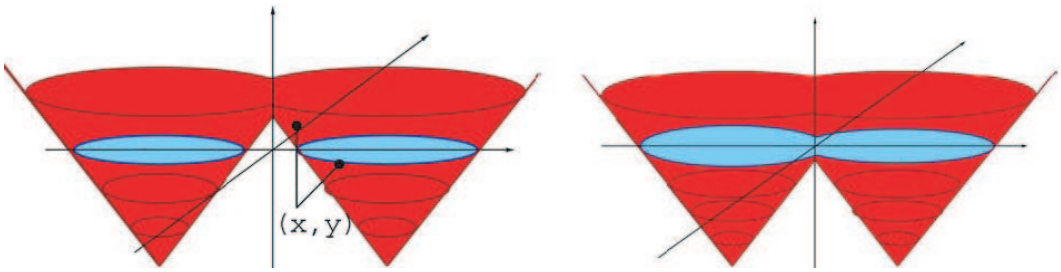


Abbildung 3.11: Eine Level-Set-Funktion (rot), deren Höhe als Abstand eines Punktes zu der Kurve definiert ist, links Zeitpunkt t, rechts Zeitpunkt $t+1$. Hell ist der sogenannte 0-Level-Set.

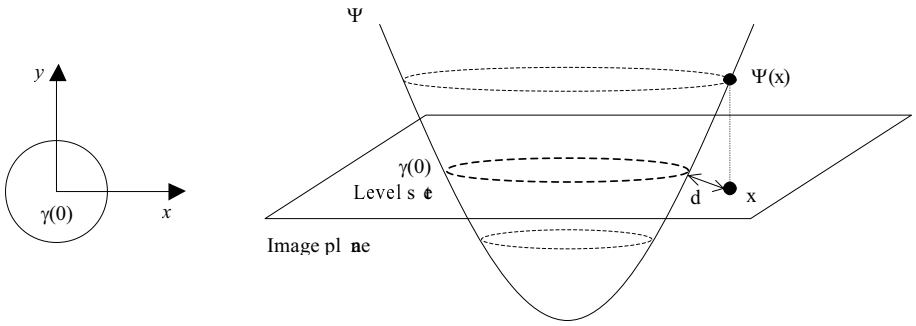


Abbildung 3.12: Level-Set-Funktion

Der Algorithmus

Die Level-Set Methode basiert auf der Idee, ein dynamisches Problem eine Dimension höher als statisches Problem zu betrachten, wobei die neue Dimension die Dynamik des Ausgangsproblems beschreiben muss. Will man also eine dynamische Kurve, die durch (x, y)-Koordinaten beschrieben ist, mit der Level-Set Methode darstellen, so legt man diese in die xy-Ebene eines dreidimensionalen Koordinatensystems und erstellt mit ihr durch die so genannte Level-Set-Funktion

$$\phi(x, y, (t = 0)) = \pm z \tag{3.26}$$

eine dreidimensionale Oberfläche, wobei der Index t den Zeitpunkt festlegt und z die Höhe eines Punktes (x, y) definiert. Der Funktionswert von z wird bestimmt durch den Abstand eines Punktes (x, y) von der Kurve (vgl. Abbildung 3.12), wobei ein positiver Wert gewählt wird, wenn sich der Punkt außerhalb der Kurve befindet. Andernfalls wählt man einen negativen Wert. Man erhält also als Level-Set Funktion eine konische Oberfläche, wie sie in Abbildung 3.11 dargestellt ist, deren Schnitt mit der xy-Ebene exakt durch die Kurve gegeben ist. Dieser Schnitt (die Ausgangskurve) heißt 0-Level-Set, da er alle Punkte beschreibt, deren Funktionswerte 0 sind.

Um mit einer so erzeugten Level-Set-Funktion eine dynamische Kurve zu beschreiben, wird die konische Oberfläche verschoben und verformt. Der Schnitt mit der xy-Ebene simuliert dann die dynamische Kurve. Die Vorteile dieses Verfahrens werden durch diese bildliche Darstellung schnell deutlich: Zum Einen können topologische Veränderungen wie in Abbildung 3.11 ohne Probleme realisiert werden, zum Anderen ist dieses Verfahren ohne Weiteres auch auf höherdimensionale Probleme übertragbar, da man am Prinzip nichts ändern muss.

Die Verformung der Kurve wird also durch die Verschiebung und Verformung der Level-Set-Funktion erzeugt. Durch die so genannte Speed-Funktion F, die jedem Kurvenpunkt (x, y) mitteilt, wie schnell er in die Verformungsrichtung strebt, wird die Verformungseigenschaft der Kurve festgelegt. Eine solche Speed-Funktion kann durch Eigenschaften der Kurve, wie beispielsweise die Kurvenkrümmung, aber auch durch externe Einflüsse, wie beispielsweise einen Bildgradien-

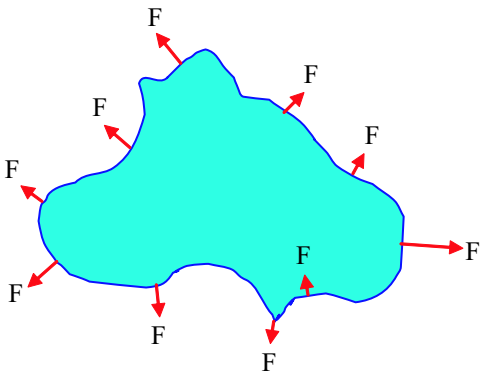


Abbildung 3.13: Eine Kontur, die nach außen drängt. Die Art und Stärke der Verformung wird durch die Speed-Funktion F bestimmt, die senkrecht an jedem Konturpunkt ansetzt. In dieser Darstellung orientiert sich die Speed-Funktion an der Kurvenkrümmung.

ten, bestimmt werden. Hier liegt auch der Ansatzpunkt, mit dem die Level-Set-Methode für die Bildsegmentierung bzw. das Tracking im CellTrack Projekt genutzt werden kann.

Um nun die Verformung einer Kurve darstellen zu können, muss das 0-Level-Set der durch die Speed-Funktion F im Zeitverlauf verformten oder verschobenen Level-Set-Funktion berechnet werden können. Wenn man sich überlegt, dass die Position eines Kurvenpunktes (x, y) zum Zeitpunkt t gegeben ist durch $y = x(t)$, führt dies zu dem Anfangswertproblem

$$\phi(x(t), t) = 0, \tag{3.27}$$

da ϕ die Höhe der Level-Set-Funktion berechnet und man genau an der Höhe 0 interessiert ist. Hier ist der Abstand der Kontur Punkte (welche indirekt durch die Eingabe gegeben sind) zum Schnitt der Ebene mit der Kurve gleich 0. Bildet man nun mit der Kettenregel die partielle Ableitung nach der Zeitvariablen t, so führt dies zu der Gleichung

$$\frac{\partial \phi}{\partial t} + \nabla(\phi(x(t), t)) \cdot x'(t) = 0. \tag{3.28}$$

Da F die Verformungsgeschwindigkeit in Normalenrichtung n nach Außen angibt, gilt

$$x'(t) \cdot m = F \text{ mit } m = \frac{\nabla(\phi)}{|\nabla(\phi)|}. \tag{3.29}$$

Setzt man dies in Gleichung 3.28 ein, so bekommt man die Level-Set-Gleichung

$$\frac{\partial \phi}{\partial t} + F\,|\nabla(\phi)| = 0, \tag{3.30}$$

die das 0-Level-Set eines Kurvenpunktes (x, y) zum Zeitpunkt t berechnet.

Um eine Numerische Lösung für Gleichung 3.30 zu bekommen, ist es nötig die Zeit- und Raum-Komponenten zu diskretisieren. ϕ_{ij}^n ist die Approximation der Lösung von $\phi(ih, jh, n\Delta t)$, dabei ist h der Abstand der Punkte des Gitternetzes, i und j die Koordinaten und Δt ist der Zeitschritt. Hierdurch bekommen wir den Iterations-Ausdruck:

$$\phi_{ij}^{n+1} = \phi_{ij}^n - \Delta t F \left|\nabla_{ij}(\phi_{ij}^n)\right| \tag{3.31}$$

Wie schon erwähnt, können verschiedenste externe und interne Kräfte in die Speed-Funktion eingebunden werden, damit jedoch ein ähnlicher Effekt wie bei den Snakes entsteht, müssen sowohl glättende als auch expandierende Kräfte diese Funktion definieren. Eine übliche Wahl der Speed-Funktion ist

$$F = (\pm 1 - \epsilon\kappa) \cdot I(x, y). \tag{3.32}$$

Hierbei ist κ die Krümmung von f, deren Wirkung durch den Parameter ϵ gewichtet wird. Sie ist definiert durch

$$\kappa = \frac{\phi_{xx}\phi_y^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{yy}\phi_x^2}{(\phi_x^2 + \phi_y^2)^{\frac{3}{2}}}. \tag{3.33}$$

Die vorgestellte Konstante bestimmt die grundsätzliche Verformungsrichtung der Kontur: eine 1 lässt die Kontur expandieren, eine -1 zieht die Kontur zusammen. Die Bestimmung dieses Wertes richtet sich also danach, ob sich die aktive Kontur von außen oder von innen der Objektgrenze nähert. Die Funktion $I(x, y)$ ist vom Bild abhängig und kann beispielsweise durch ein Grauwertgebirge bestimmt werden.

Die Aufgaben der einzelnen Teile dieser Speed-Funktion werden im Vergleich mit dem Snake-Modell schnell deutlich: Der erste Teil soll die Kontur, ähnlich wie die interne Energie der Snake, glatt halten, der zweite Teil wirkt wie die externe Energie der Snake. Natürlich können auch andere, weitaus komplexere Speed-Funktionen definiert werden, anhand des hier vorgestellten Beispiels soll jedoch nur der grundsätzliche Ansatz deutlich werden [LCPL01].

Umsetzung

Der Level-Set-Algorithmus wurde für das CellTrack Projekt in einer kompakten Klasse implementiert. Dies sollte eine leichte Portabilität zwischen den Prototypen sicherstellen. Neben der QT-API verwendet die Klasse noch die Newmat Bibliothek (vgl. Kapitel 4.1.3) und nutzt deren Datenstruktur für die Bereitstellung von 2 Dimensionalen $m \times n$ Matrizen. Auch einfache Rechenoperationen auf den Matrizen werden von der Newmat genutzt und mußten so nicht neu implementiert werden. Einige Rechenoperationen die benötigt werden und von der Newmat nicht bereitgestellt werden wurden als private Funktionen innerhalb der Klasse implementiert. Dies sind z.B. Funktionen für einen zyklischen Shift, die Divergenz oder die Berechnung der Heaviside-Step-Funktion.

Der eigentliche Level-Set-Algorithmus unterteilt sich in einen public *Levelset*-Aufruf, welcher in zwei Kontur-Initialisierungsmethoden, sowie die Funktion *Activecontour* unterteilt ist. Die *Activecontour* Funktion implementiert den eigentlichen ActiveContour-Level-Set-Algorithmus. Die beiden Initialisierungs-Funktionen sind nötig, da der Algorithmus auf zwei verschiedene Arten gestartet werden kann:

1. Am Anfang eines Videos, oder wenn noch nicht mit dem Level-Set-Algorithmus gearbeitet wurde, existiert entsprechend auch noch keine für den Algorithmus als Ausgangspunkt zu verwendende Kontur. Daher wird aus den Koordinaten der Makierung, die in anderen Teilen des Programms erzeugt wird (vgl. Kapitel 4.2.3), ein Kreis berechnet, der innerhalb der (rechteckigen) Makierung liegt und sie maximal ausfüllt. Die Koordinaten des Kreises werden der Initialisierungs-Methode mit den Bilddaten übergeben und es wird daraus eine Initialisierungs-Matrix mit dem Kreis als Ausgangskontur für den Algorithmus erzeugt. Ein Kreis wird verwendet, weil er als geometrische Form der Zell-Kontur relativ am nächsten kommt und damit dem Algortithmus hilft in weniger Iterationen sich der tatsächlichen Zell-Kontur anzugleichen.

2. Wenn der Algorithmus schon einmal (z.B. auf dem vorherigen Bild) gelaufen ist, so kann die Kontur-Matrix vom vorhergehenden Bild wiederverwendet werden. Aufgrund der meist nur geringen Änderungen zwischen zwei Bildern dient sie als gute Ausgangskontur, die der Zell-Kontur wahrscheinlich schon sehr ähnlich ist und damit ein guter Ansatz für die Iterationen des Algorithmus ist.

Ursprünglich wurde dies auch so implementiert. Um aber ein „auslaufen“ der Kontur und dadurch öfter auch ein überspringen auf benachbarte Zellen zu minimieren wurde eine Art Konturkorrektur-Funktion dem Algorithmus nachgeschaltet. Diese schaut sich als erstes alle in sich geschlossenen Flächen an und eliminiert sie, bis auf die Größte, da sich darunter mit höchster Wahrscheinlichkeit die Zelle befindet. Weiterhin werden Ausläufer, die nur durch dünne Bereiche mit dem Hauptteil der Kontur verbunden sind, ebenfalls eliminiert. Die Kontur wird dann als Boolesche Matrix abgelegt (alle Felder mit 1 gehören zu der Fläche auf der die Zelle liegt). Die Umwandlung in Boolesche Werte macht die Nachbearbeitung der Kontur deutlich einfacher.

Diese Kontur Matrix wird dann beim nächsten Aufruf des Algorithmus wieder eingelesen und in eine für den Algorithmus verwendbare Matrix umgewandelt.

Für die Zusammenarbeit der verschiedenen Algorithmen im Projekt CellTrack bestand die Idee jedem Algorithmus sich einen „Vertrauenswert“ geben zu lassen. Hierüber sollte eine Aussage über die Wahrscheinlichkeit eines richtigen Ergebnisses gemacht werden können, so dass bei unterschiedlichen Ergebnissen der verschiedenen Algorithmen eine Abschätzung möglich ist, wessen Ergebnis am ehesten das Richtige ist.

Als Grundlage für einen Vertrauenswert wurde die Fläche, der Umfang und die Relation dieser beiden Werte versucht heranzuziehen, da dies die einzigen Werte sind, die bei diesem Algorithmus die Zelle charakterisieren. Die Ergebnisse sind aber leider nicht besonders gut und variieren je nach verwendetem Video, da die Zellcharakteristika diesbezüglich sehr uneindeutig sind. Grund dafür, dass die Berechnung eines vernünftigen Vertrauenswertes sich als sehr schwierig bzw. eigentlich unlösbar (mit diesem oder ähnlichen Algorithmen) herausgestellt hat, ist eigentlich in den Algorithmen zu suchen, da diese keinerlei eigene Fehlererkennung besitzen. Dies ist auch schwierig zu realisieren, denn würde der Algorithmus selber schon merken, dass sein Ergebnis nicht optimal ist, so würde er dieses Ergebnis vernünftigerweise nicht wählen, sondern ein besseres. D.h. ein Algorithmus wählt immer das für ihn scheinbar beste Ergebnis. Eine Einschätzung zur Güte (im Vergleich zum absolut besten Ergebnis) ist somit sehr schwierig. Interessant wäre sicher die Untersuchung, ob es Algorithmen gibt bzw. ob bestehende so modifiziert werden können, dass eine relativ gute Aussage über die Güte des erzielten Ergebnisses getroffen werden kann [Was04].

Probleme bei der Verwendung von Levelset

Als größtes Problem (neben der fehlenden Aussage über die Güte des Ergebnisses - die auch erst später als wünschenswert erkannt wurde) hat sich die Laufzeit des Algorithmus herausgestellt. Da relativ viele Iterationen zum Finden der Kontur benötigt werden, schlägt die hohe Anzahl an Rechenoperationen, die benötigt wird, zusätzlich zu Buche. Grund ist, dass jeder Bildpunkt in die Berechnung der Kontur mit einfließt. Um dem ein wenig zu begegnen wird nur ein Ausschnitt des Bildes um die Kontur betrachtet. Dadurch ist der Algorithmus deutlich schneller. Eine weitere deutliche Steigerung wäre durch eine Abart des Level-Set-Algorithmus Namens „Narrowband“ möglich. Bei dieser Methode wird anstatt des gesamten Bildes (oder Bildausschnitts) nur ein schmales „Band“ um die Kontur betrachtet. Alle anderen Bildpunkte werden als plus oder minus unendlich angesehen. Ändert sich die Kontur stark in dem Band so wir dieses der Kontur angepasst. Im Rahmen des CellTrack wurde angefangen dieses zu programmieren. Allerdings ist die Implementation nicht ganz einfach, da eine eigene Datenstruktur für das „Narrowband“ implementiert werden muss, auf der dann alle Operationen zur Verfügung gestellt werden müssen, wie sie vom normalen Level-Set auf den Matrizen auch verwendet werden. Bei der Implementierung des normalen Level-Set wird schon viel Arbeit durch die Verwendung der „Newmat“ Bibliothek abgenommen. Durch Zeitmangel gegen Ende der PG ist es dann nicht mehr zu einer funktionierenden Implementierung des „Narrowband-Level-Set“ gekommen.

3.4.5 Kalman-Filter

Der Kalman-Filter wurde an Hand der Beschreibungen von Welsh und Bishop implementiert, so wie er auch im Kapitel Grundlagen beschrieben wurde (vgl. Kapitel 2.6.6). Das Ziel für die Anwendung

des Kalman-Filters war es, die verschiedenen Tracking-Verfahren zu überprüfen, ob ihre Ergebnisse von den geschätzten Ergebnissen abweichen. Dazu wurden zwei Kalman-Filter implementiert: Zum einen der „normale“ Kalman-Filter für zweidimensionale Daten und einen „Multi-Modell“ Kalman-Filter, der auf mehreren Modellannahmen gleichzeitig arbeiten sollte [WB01].
Jedoch erwiesen sich beide Filter als nicht anwendbar auf das CellTrack-Problem, da der Kalman-Filter nicht als Fehlerkorrektur für Modellannahmen verwendet werden kann und vernünftige Modellannahmen für die Bewegung der Zellen nicht ermittelbar waren. Die ersten Tests mit dem Standard-Kalman-Filter zeigten, dass er zwar in der Lage war eine Bewegung zu verfolgen, jedoch konnte er keine Messabweichungen durch fehlerhaftes Tracking korrigieren, obwohl sich die Zelle gleichförmig, und mit der Modellannahme übereinstimmend, weiterbewegte. Er folgte der Zellbewegung mit einer kleinen Zeitverzögerung von wenigen Frames. Dieses Verhalten war auf die anfangs zu gering gewählten Rauschannahmen zurückzuführen. Allerdings brachten auch realistischere Annahmen, wie zwei bis fünf Pixel Messrauschen und bis zu 10 Pixel Systemrauschen keine wesentlichen Verbesserungen des Schätzers. Auch die Anpassung der Modellannahme für die Bewegung der Zellen lieferte nur kurzzeitige Erfolge. Zwar konnte mit einer bestimten Modellannahme eine Zelle gut verfolgt werden, so lange sie sich so bewegte, wie es das idealisierte Modell vorhersagte. Jedoch wichen die Ergebnisse des Trackings und des Kalman-Filters erheblich ab, sobald die Zelle ihre Richtung oder ihre Geschwindigkeit änderte. Da der Kalman-Filter weiterhin versuchte, die Messergebnisse und seine Modellannahmen zu kombinieren, lieferte er Ergebnisse, die zwischen den Tracking-Ergebnissen und dem idealisierten, aber nun falschen Zellpfad lagen.
Somit wurde schnell klar, dass mehrere Modellannahmen nötig waren, um die Bewegung der Zellen beschreiben zu können. Die Modellannahmen wurden auf jede Zelle angewendet, so dass über den Wert des Kalman-Gains entschieden werden sollte, welche Modellannahme am besten zur Bewegung der Zelle im aktuellen Zeitschritt passte. Hier zeigte sich, dass das korrekte Modell nicht durch den Kalman-Gain bestimmt werden konnte, da dieser nur aus den Modellannahmen berechnet wird. War unter den verschiedenen Modellannahmen beispielsweise auch ein Modell, das keine Bewegung vorhersagte, so wurde dieses Modell bervorzugt gewählt, da sich bei gleichen Rauschannahmen ein kleinerer Kalman-Gain aus den Matrizen ergabt, als bei anderen Modellen der Zellbewegungen. Dies führte zur Implementierung des „Multi-Modell“ Kalman-Filters, der mehrere Modellannahmen kombinieren kann und daraus ein besseres Modell erhält [WB01]. Allerdings brachte auch diese Version des Kalman-Filters keine Verbesserung des Tracking-Prozesses, da weiterhin nur die Ergebnisse der anderen Verfahren verfolgt und Abweichungen und Fehler nicht erkannt wurden. Neben der Positionsschätzung wurde auch versucht, einen Schätzer für die Geschwindigkeit der Zelle zu verwenden, doch auch dieser hatte die oben erwähnten Probleme.
Zusammenfassend kann man sagen, dass dem Kalman-Filter die Möglichkeit fehlt, als Werkzeug zur Fehlerüberprüfung und Fehlerkorrektur verwendet zu werden. Da die vom Kalman-Filter geschätzten Werte immer nur die Modellannahme angewandt auf die Messwerte darstellen und sinnvolle Modelle für das Rauschen und die Bewegung der Zellen nicht zu bestimmen waren, wurde beschlossen, den Kalman-Filter nicht für das Tracking zu verwenden.

3.5 SQL - relationale Datenbanken

Im Rahmen des Projektes wurde klar, dass man die Zellen bzw. die Positionen der Zellen in jedem Frame abspeichern muss, um die Ergebnisse nachprüfbar zu halten. Zur Realisierung dieses Problems fiel die Entscheidung, die Daten in einer SQL-Datenbank (SQL = Structured Query Language) zu speichern. Alternativ hätte man die Daten auch in XML-Dateien abspeichern können, dieser Vorschlag aber wurde verworfen, da bei den anfallenden Mengen an Daten Datenbanken die wesentlich stabilere und einfacher zu pflegende Wahl sind.
Die verwendete Datenbank ist eine relationale SQL-Datenbank. Die Daten werden in zweidimensionale Tabellen gefasst, die über Relationen miteinander verbunden sind. Von einer Tabelle kann man verbundene Tabellen mittels entsprechender Schlüssel erreichen. Auf diese Datenbanken erhält man Zugang mittels SQL, eine weitgehend standartisierte eigenständige Sprache für den Zugriff auf

Datenbanken [Mat00]. Der Vorteil hierbei ist, dass sie weitgehend unabhängig von der restlichen Entwicklungsumgebung arbeitet und als Bedingung nur eine relationale Datenbank voraussetzt. Für CellTrack wurde „mySQL“ gewählt, da diese Open-Source-Datenbank zum einen weitverbreitet ist und zum anderen als robust angesehen werden kann. So verwenden viele auf Linux basierten Webserver „mySQL“ als Datenbank. Dass „mySQL“ für fast alle Betriebssysteme verfügbar ist, erklärt zusätzlich die Verbreitung der Datenbank.
Größere Projekte werden mittels des Entity-Relationship-Modells erstellt, da aber die zu speichernden Daten überschaubar blieben, wurde auf ein Modell verzichtet. In dem CellTrack-Projekt beinhaltete der erste Datenbankentwurf alle denkbaren Daten der Zellen, also auch die Daten, die nur in der Statistik auftauchen. Es stellte sich aber heraus, dass diese Datenmenge den Zugriff auf die Datenbank zu langsam machte. Der endgültige Entwurf sah eine schlanke Version der Datenbank nur mit den nötigsten Daten vor. Die Datenbank ist wie folgt aufgebaut: Die zentrale Tabelle ist `TBVideo`, die sowohl die Daten des Videos enthält, als auch den Benutzernamen, damit Videos benutzerbezogenen angezeigt werden können. Die einzelnen Frames sind in einer eigenen Tabelle (`TBFrame`) abgespeichert, wobei diese Tabelle als Verbindung zwischen `TBCells` (Tabelle für die Zellmittelpunkte) und `TBVideo` dient. Da die Zellen für jeden Frame einen Mittelpunkt haben, war diese Verbindung nötig.
Die Tabellen sehen explizit wie folgt aus:

TABLE `TBCCellsFrame` (
    TBFrame_FrameID INTEGER,
    TBCells_CellID INTEGER,
    CenterX INTEGER,
    CenterY INTEGER,
    Fremdschlüssel zur Tabelle `TBCCellsFrame`(TBCells_CellID),
    Fremdschlüssel zur Tabelle `TBCCellsFrame`(TBFrame_FrameID)
);

TABLE `TBCells` (
    CellID INTEGER,
    InitialContourStartX INTEGER,
    InitialContourStartY INTEGER,
    InitialContourEndX INTEGER,
    InitialContourEndY INTEGER,
    TrackedSuccessfully BIT,
    primärer Schlüssel(CellID)
);

TABLE `TBFrame` (
    FrameID INTEGER,
    TBVideo_VideoID INTEGER,
    FrameNumberInVideo INTEGER,
    primärer Schlüssel(FrameID),
    Fremdschlüssel zur Tabelle `TBFrame`(TBVideo_VideoID)
);

TABLE `TBVideo` (
    VideoID INTEGER,
    VideoDate DATE,
    Username VARCHAR(255),
    CellType VARCHAR(255),
    FPS FLOAT,
    FilePath TEXT,
    MediaNumber VARCHAR(255),

```
    NumberOfFrames BIGINT,
    EquipmentDetails FLOAT,
    Comment VARCHAR(255),
    Agent VARCHAR(255),
    TrackedSuccessfully BIT,
    primärer Schlüssel(VideoID)
);
```

Kapitel 4

Implementierung

Inhaltsangabe

4.1 Hilfsmittel

Das Projekt CellTrack wurde komplett unter Linux entwickelt. Da man bei einem solchen Projekt in der vorgegebenen Zeit nicht sämtliche Funktionalitäten selbst von Grund auf programmieren kann, wurden zur Verfügung stehende Tools und Bibliotheken genutzt. Dieser Abschnitt gibt eine kurze Übersicht der verwendeten Hilfsmittel.

4.1.1 KDevelop

Für die Programmierung wurde unter dem Fenstermanager KDE 3.4.0 das Werkzeug KDevelop 3.2.0 genutzt. Dieses Tool bietet die üblichen Features, die man von einer modernen Entwicklungsumgebung erwartet. Dazu gehören ein Klassenbrowser, Syntaxhighlighting, einfache Nutzung des Compilers und ein Debugger. Zudem bietet KDevelop die direkte Einbindung von Qt an.

4.1.2 Qt

Qt ist ein GUI Toolkit für C++ und wurde für verschiedene gängige Plattformen implementiert. Es ist relativ weit entwickelt und stabil. Bei dem Projekt wurde Qt in der Version 3.3.4 benutzt. Qt ist völlig objektorientiert und bietet dem Programmierer die Möglichkeit, Anwendungen mit einem Benutzerinterface zu entwicklen, die state-of-the-art sind. Zur einfacheren Gestaltung der GUI stand der Qt-Designer in der Version 3.3.4 zur Verfügung. Nicht nur für die GUI bot sich die Benutzung von Qt an, sondern auch seine Erweiterung der Standard-C++-Klassen `QString`, `QFile` usw. waren hilfreich.

4.1.3 Newmat

Da bei dem Projekt häufig Matrizen und Standardoperationen auf diesen verwendet werden, bot sich die Matrizenbibliothek Newmat1.1 an. Die Operationen werden effizient von der Bibliothek

durchgeführt und sind im Code einfach zu nutzen und sehr übersichtlich.

4.1.4 Together

Mit dem Tool Borland Together 6.2 wurden Klassendiagramme und Aktivitätsdiagramme für CellTrack angefertig.

4.1.5 CVS (Concurrent Versions System)

Bei der Entwicklung mit einer größeren Gruppe bekommt man schnell Probleme mit der Konsistenz der Dateien. Um dies zu vermeiden wurde das CVS benutzt. Konflikte können leicht bearbeitet werden und man ist jederzeit in der Lage, ältere Versionen wiederherzustellen.

4.1.6 LaTeX

Sämtliche Dokumente wie Seminarband, Endbericht und die Sitzungsprotokolle wurden mit diesem Textsatzsystem erstellt. LaTeX eignet sich sehr gut für die Verfassung wissenschaftlicher Dokumente. Es stehen viele hilfreiche Pakete zur Verfügung und die Einteilung in separate Teile, die einzeln bearbeitet werden können, stellt kein Problem dar.

4.1.7 Dirac

Das Videomaterial, das der Projektgruppe zur Verfügung stand, war recht umfangreich und nahm viel Speicherplatz in Anspruch. Da eine große Anzahl von Videos archiviert werden musste, stellte Speicherplatzmanagment eine weitere Anforderung an das Projekt. Diese Anforderung wird von Dirac erfüllt ohne die Bildqualität stark zu beeinflussen. Dirac wird in der Version 0.5.2 eingesetzt [dir].

4.1.8 SQL

Eine Datenbankanbindung ist notwendig um die gesamten Zellen- und Tracking-Informationen zu archivieren. SQL bietet für diese Zwecke eine effiziente und genügend mächtige Datenbank. SQL ist zudem frei verfügbar und bereitet wenig Aufwand, wenn man die Datenbank administrieren muss. Weitere Informationen über SQL sind dem Kapitel 3.5, [pos] oder [sql] zu entnehmen.

4.2 Klassendokumentation

4.2.1 Paketdiagramm

Abbildung 4.1 zeigt die Beziehungen zwischen den einzelnen Paketen auf. Betrachtet man das Diagramm, so fallen die vielen Beziehungen zwischen dem Paket Main und den anderen Paketen auf. Dieses Paket steht mit dem GUI-, Video-, Statistics-, Database-, Entities- sowie mit dem Tracking-Paket in Verbindung. Es steuert den Kontrollfluss des Projektes. Weiterhin fällt das Paket Datastructures sowie das Tracking-Paket auf. Diese beiden Pakete stehen in enger Verbindung zueinander und sind für die Zellverfolgung hauptsächlich verantwortlich.

4.2.2 Abstraktes Klassendiagramm

Das abstrakte Klassendiagramm in Abbildung 4.2 dient zur Verdeutlichung der Beziehungen zwischen den Klassen. Im abstrakten Klassendiagramm sind keine Attribute und Methoden aufgeführt. Die Vererbungshierarchien innerhalb der Applikation und die Assoziationen kann man so klarer erkennen und nachvollziehen.

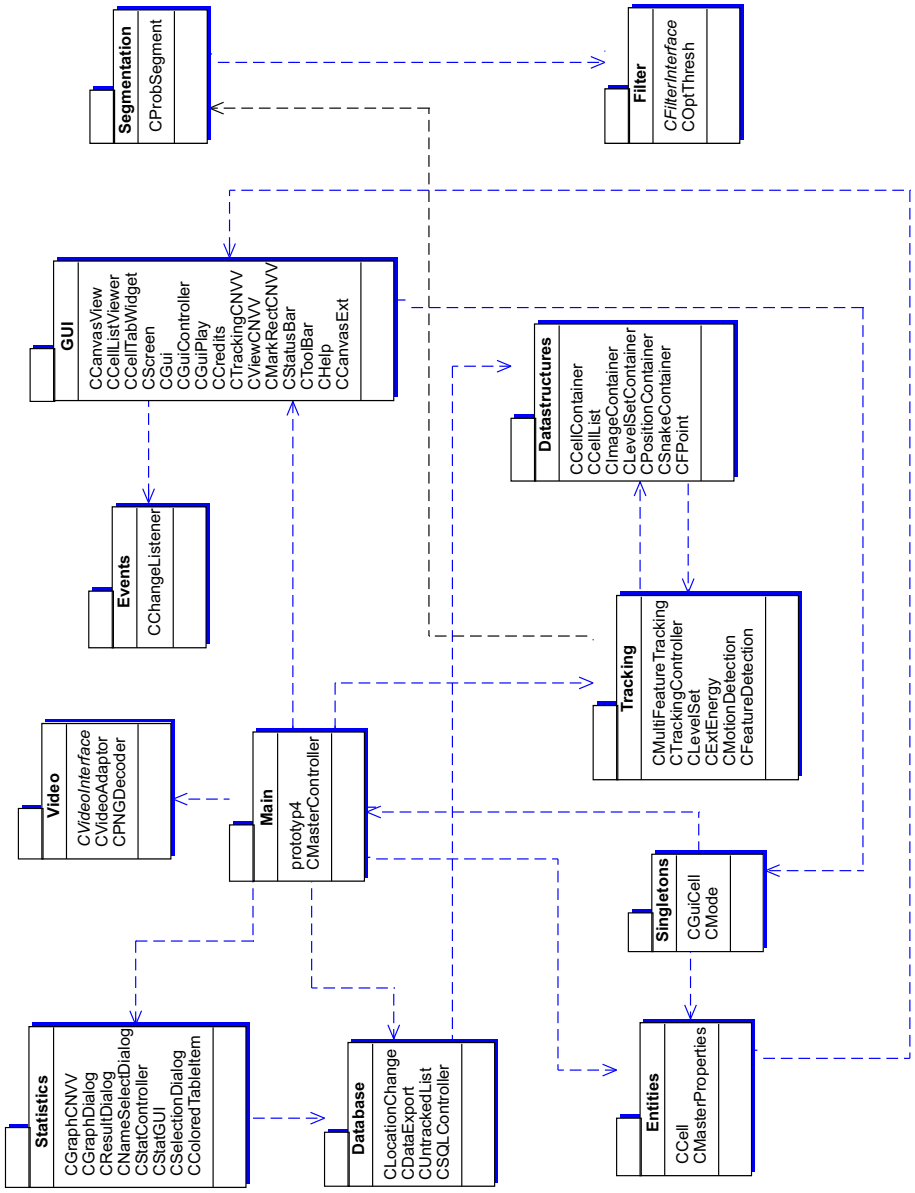


Abbildung 4.1: Paketdiagramm mit allen Klassen

4.2.3 Klassenbeschreibung

Das komplette Klassendiagramm ist für die Darstellung zu groß, deshalb werden im Folgenden die einzelnen Komponenten dargestellt.

Video

Klasse CVideoInterface

Hierbei handelt es sich um ein Interface, welches die Grundmethoden vorgibt.

Klasse CVideoAdapter

Die Klasse CVideoAdapter stellt die Methoden zur Verfügung, die dann von der Benutzer-

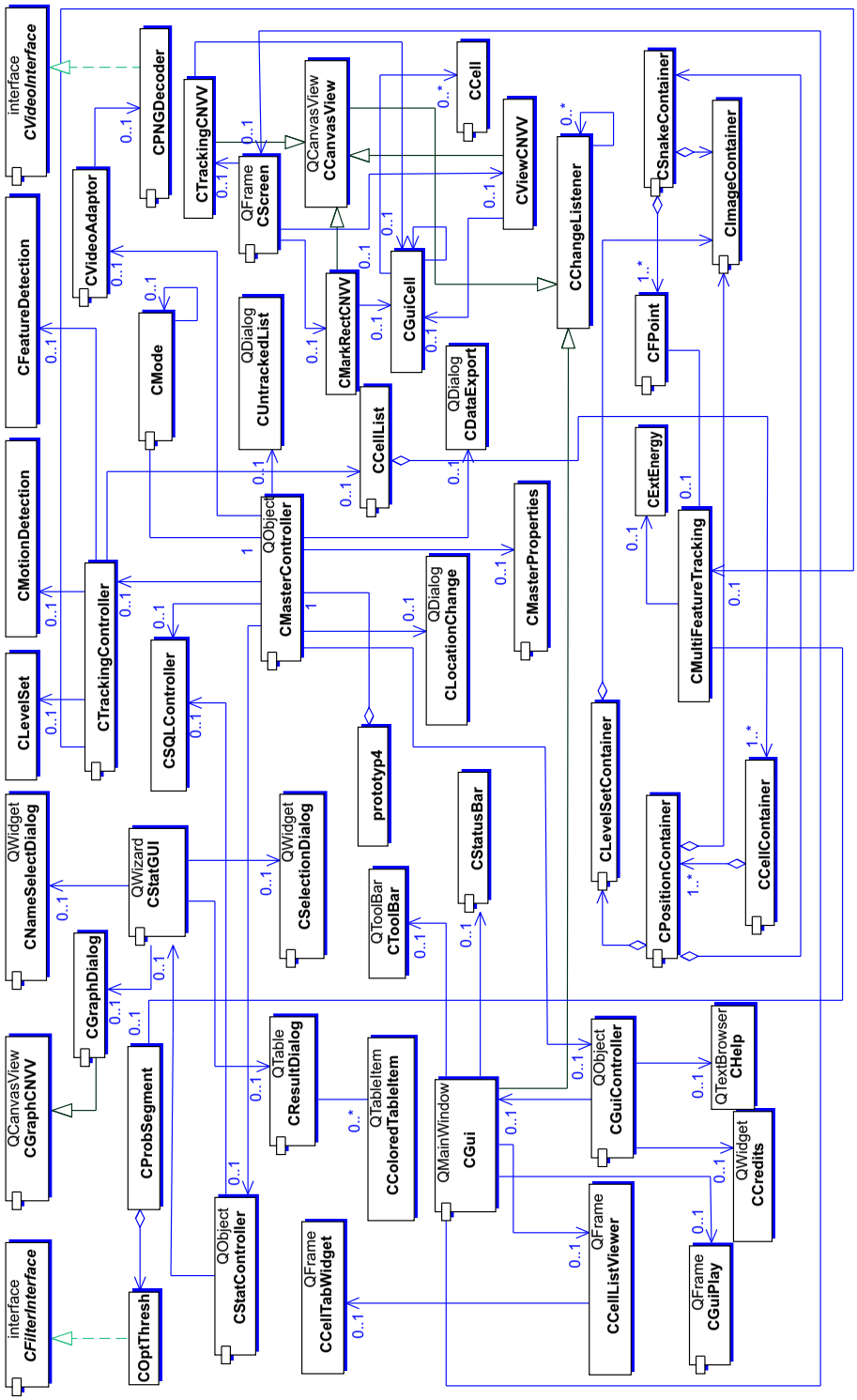


Abbildung 4.2: Abstraktes Klassendiagramm dient der Verdeutlichung der Beziehungshierarchie

oberfläche aus aufgerufen werden. Hierbei handelt es sich um Methoden wie das Laden des Videos.

Klasse CPNGDecoder
Die Klasse CPNGDecoder implementiert das Interface CVideoInterface. Dies ist für die Arbeit mit den PNG-Bildern notwendig.

Benutzerinterface (GUI)

Die Abbildung 4.3 zeigt die Klassen, die zur Benutzeroberfläche gehören. Im Folgenden werden diese genauer beschrieben.

Klasse CGUI
Die Klasse `GUI` umfasst die Methoden, mit denen das Aussehen der Benutzeroberfläche bestimmt wird. Auch die Kontrolle über das Ausblenden und Einblenden der Buttons liegt in dieser Klasse.

Klasse CGuiController
Der `GuiController` überwacht die gesamten Slots und Signals. Jede Aktion des Benutzers wird an die entsprechenden Klassen weitergeleitet.

Klasse CCellListViewer
Betrachtet man die GUI, so fällt die Liste der Zellen oben rechts auf. Die Funktionalität dieser Liste der Zellen wird durch diese Klasse sichergestellt.

Klasse CCellTabWidget
Die Klasse `CellTabWidget` kümmert sich um die Funktionalität der Befehlsschaltfläche unten rechts in der GUI. Der Benutzer hat hier die Möglichkeit über einen Reiter verschiedene Funktionen wie das Autotracking zu aktivieren.

Klasse CCredits
Diese Klasse ist für die Anzeige der Mitwirkenden verantwortlich. Die Namen der mitarbeitenden Studenten, der Betreuer, sowie ein Logo werden mit Hilfe dieser Klasse dargestellt. Diese Klasse bietet sonst keine weiteren Funktionen.

Klasse CGuiPlay
Die Abspielfunktionen werden hier behandelt. Ebenso die Möglichkeiten, die Schaltflächen bei Bedarf auszugrauen oder wieder darzustellen.

Klasse CHelp
Das Programm bietet die Möglichkeit über das Menü die Hilfe zu öffnen. Hierbei wird das Handbuch aufgerufen und diese Funktion ist auch der Hauptbestandteil dieser Klasse.

Klasse CScreen
Die Zeichenfläche `QCanvas` wird in dieser Funktion behandelt.

Klasse CStatusBar
Die Statuszeile und der Fortschrittsbalken werden hier umgesetzt.

Klasse CToolBar
Eine weitere Klasse der GUI, hier werden Funktionen wie „Save as“ oder „Exit“ umgesetzt. Die Klasse hat nur eine Methode, die aufgerufen wird, wenn der Zustand der Buttons sich ändert.

Klasse CCanvasExt
Sollte das Fenster verschoben oder die Größe des Fensters verändert werden, so wird hier die neue gültige Position der Canvas-Zeichenfläche berechnet.

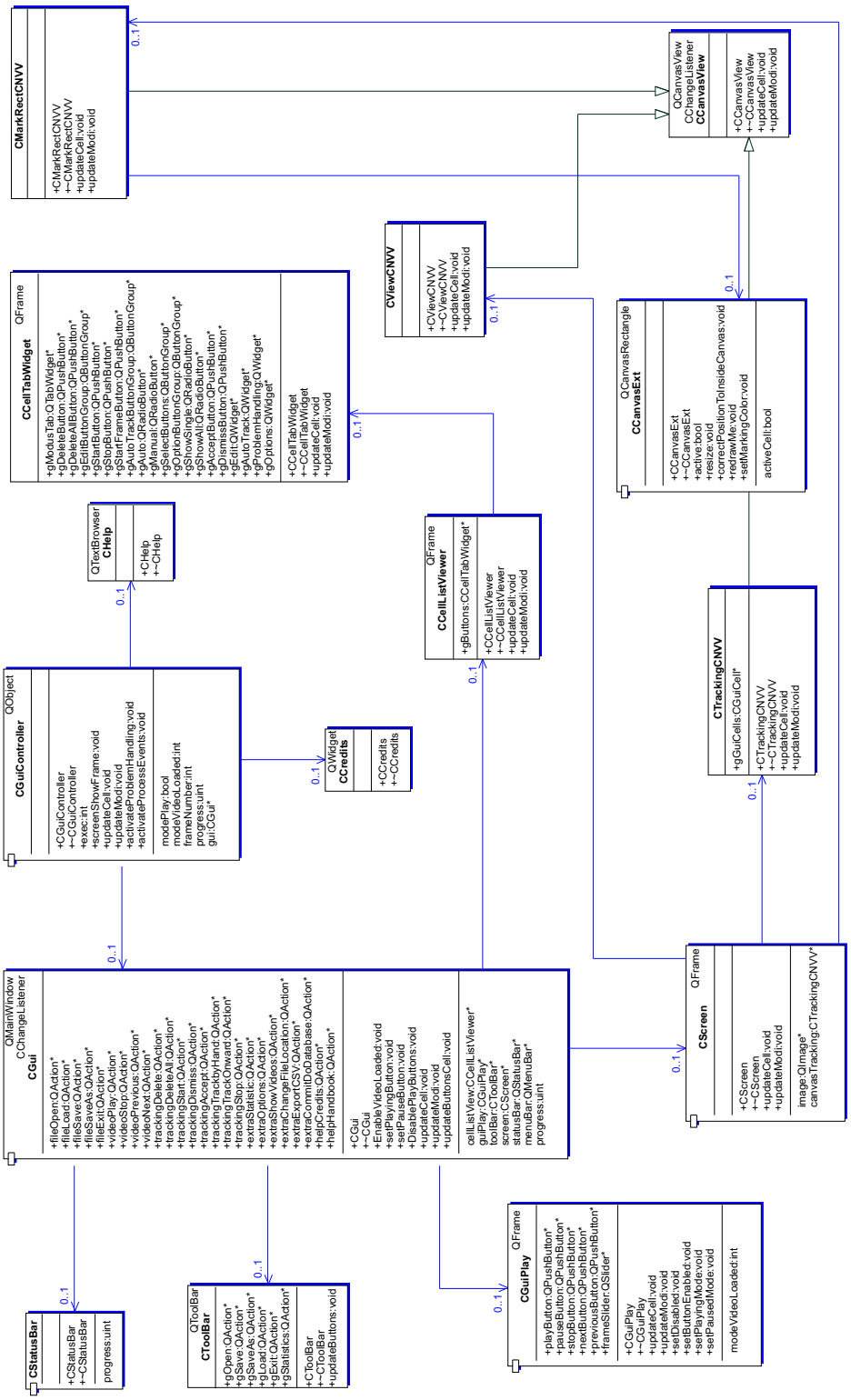


Abbildung 4.3: Klassendiagramm des Paketes GUI

Klasse CCanvasView
Dies ist eine Klasse, die die aktive Mausselektion ermittelt. Diese Klasse bietet wenig Funktionalität, wird jedoch von den folgenden Klassen spezialisiert. Sie selbst erbt von `QCanvasView` und von `ChangeListener`.

Klasse CMarkRectCNVV
Der Benutzer muss in der Lage sein die Zellenmarkierungen zu löschen oder zu verändern. Diese Funktionalität wird in dieser Klasse gewährleistet. Die wichtigste Methode in der Klasse ist wohl die Methode *mousePressEvent*, die das Drücken der Maus im Zeichenfenster behandelt. Die Klasse erbt von `QCanvasView`.

Klasse CTrackingCNVV
Auch diese Klasse erbt von `QCanvasView` und erweitertet diese um die Funktionen, die beim Tracken notwendig werden.

Klasse CViewCNVV
Diese Klasse erbt von der Klasse `QCanvasView` und erweitert die Funktionalität. Die Hauptfunktionen dieser Klasse sind *updateCell* und *updateModi*.

Verwendete Datenstrukturen

Nach den ersten Versuchen mit dem Blockvergleichverfahren wurde schnell klar, dass eine komplexe Datenstruktur benötigt wird. Denn allein dieses eine Verfahren lieferte schon recht viele Tracking-Informationen.
Das Blockvergleichverfahren arbeitet mit einem Featurebild und einem Suchbereich, wobei beide Bereiche Rechtecke sind. Von beiden Rechtecken müssen die Koordinaten der Ecken abgespeichert werden. Schnell wurde die Idee entwickelt, dafür einen eigenen Container zu schreiben, den `ImageContainer`. Dieser enthält die x- und y-Koordinaten der linken oberen und rechten unteren Ecke des Suchbereichs und des Featurebildes.
Dann musste die Frage gelöst werden, wie diese `ImageContainer` für jede Zelle und jedes Bild am besten abgespeichert werden können. Die verwendete Lösung dafür ist, für jede Zelle eine Liste von Bildern zu speichern, in der die entsprechenden `ImageContainer` hängen und somit die Tracking-Informationen für jedes Bild zu einer Zelle. Es wird schneller Zugriff auf die Daten und einfaches Einfügen der Daten in die Liste benötigt. Also wurde die Standard Template Library verwendet und von dieser die Klasse `vector`. Nun ging es daran, die `ImageContainer` vernünftig in diesen Vektor zu schreiben. Dafür wurde ein weiterer Container hinzugefügt, der `CellContainer`. Dieser wird für jede Zelle angelegt und enthält einen Vektor, der aus `ImageContainern` besteht. Die `CellContainer` wurden ihrerseits auch in einem Vektor verwaltet, genannt CellList.
Für einen Tracking-Algorithmus reichte diese Struktur aus. Bald kam aber das Snake-Verfahren hinzu. Die Snake arbeitet mit Konturpunkten, die den Rand der Zelle angeben. Außerdem soll sie zusätzlich zu diesen Daten auch das Featurebild als Rechteck abspeichern, damit die Ergebnisse des Snake-Verfahrens leicht eingezeichnet und die Ergebnisse der einzelnen Verfahren schnell miteinander verglichen werden können. Das heißt, auch das Snake-Verfahren benötigt einen `ImageContainer`. Da später alle Verfahren auf jede Zelle angewendet werden sollen, musste für das Snake-Verfahren ein eigener `ImageContainer` erstellt werden. Weil aber noch zusätzlich zu den Daten, die in dem `ImageContainer` gespeichert werden können, weitere Daten hinzukommen, wurde ein weiterer Container erstellt, der `SnakeContainer`. Dieser enthält den `ImageContainer` und alle zusätzlich notwendigen Daten.
Beide Container, der `ImageContainer` für das Blockvergleichverfahren und der `SnakeContainer` für das Snake-Verfahren sollten nun für jede Zelle und jedes Bild gespeichert werden. Um dies zu erreichen wurde ein weiterer Container erzeugt, der `PositionContainer`. Dieser enthält nun die beiden Container und kann in den Vector eingehängt werden, der für jede Zelle bereits besteht.
Als drittes und letztes Verfahren kam nun das LevelSet-Verfahren hinzu. Dieses benötigt zum Speichern der Tracking-Informationen eine Matrix. Aber zusätzlich zu dieser Matrix werden wie

bei dem Snake-Verfahren wieder die Koordinaten des Featurebildes abgespeichert. Das LevelSet-Verfahren braucht also auch wiederum einen eigenen Container, den `LevelSetContainer`. In diesem `LevelSetContainer` ist also die Matrix und ein `ImageContainer` enthalten. Dieser wird nun zu den anderen Containern auch in den `PositionContainer` geschrieben. Somit können nun alle Tracking-Informationen, die die einzelnen Verfahren generieren, abgespeichert werden.
Da jedes Verfahren einen eigenen `ImageContainer` und noch zusätzlich weitere unterschiedliche Datentypen benötigt, in denen Tracking-Informationen abspeichert werden, ist diese Lösung für das Tracken von Zellen am sinnvollsten. Im Folgenden wird der Aufbau der CellList genauer beschrieben. Außerdem wird auf den genauen Aufbau der verschiedenen Container eingegangen.

Aufbau der CellList

Die CellList ist nun ein Vector, der `CellContainer` enthält. Diese Container werden für jede markierte Zelle erzeugt. Ausserdem enthält jeder dieser Container wiederum einen Vektor, in dem die Tracking-Informationen für jedes Bild und jedes Tracking-Verfahren abgespeichert werden. Man hat also letztendlich eine Liste von Listen. Abbildung 4.4 soll dies verdeutlichen.

Ein `PositionContainer` wird genau dann erstellt, sobald Tracking-Informationen für das aktuelle Bild vorhanden sind. Dieser wird dann in den Vektor gehängt, der im `CellContainer` vorhanden ist. Die Container der einzelnen Verfahren werden entsprechend dann gesetzt und in den `PositionContainer` geschrieben, wenn die Verfahren aufgerufen wurden und Tracking-Informationen existieren. Es kann z. B. passieren, dass der `LevelSetContainer` nicht für jedes Bild vorhanden ist. Dies hängt mit dem Aufbau des TrackingControllers zusammen (vgl. Kapitel 4.5), der nicht für jede Zelle jedes Verfahren aufruft. Somit sind auch nicht für jedes Bild die Informationen aller Verfahren vorhanden.
Der `PositionContainer` seinerseits besteht nun aus den einzelnen Containern der verschiedenen Verfahren, wie einem `ImageContainer` für das Blockvergleichverfahren, einem `SnakeContainer` für das Snake-Verfahren und einem `LevelSetContainer` für das LevelSet-Verfahren.
Wie diese einzelnen Klassen zusammenhängen, zeigt das Klassendiagramm in Abbildung 4.5 . Die Klasse `CellList` ist die Kontrollklasse, sie kontrolliert alle Operationen, die auf den verschiedenen Containern ausgeführt werden können. D. h. man hat keinen direkten Zugriff auf die Container, sondern muss immer über die CellList gehen, um an Informationen zu kommen oder Informationen zu schreiben oder zu ändern. Dabei reicht die `CellList` die verschiedenen Aufrufe an die entsprechenden Klassen weiter.
Dazu ein kleines Beispiel: Wenn der `ImageContainer` für das Blockvergleichverfahren ausgelesen werden soll, wird auf der CellList die Operation `getImageContainer` aufgerufen. Diese Funktion ruft aber nur die Funktion `getImageContainer` im `CellContainer` auf und dieser wiederum gibt den Aufruf an den `PositionContainer` weiter. Erst dort wird durch die Methode `getImageContainer` der gewünschte `ImageContainer` zurückgegeben.

Klasse CImageContainer

Der `ImageContainer` wird für alle drei Tracking-Verfahren benötigt. In diesem werden die Koordinaten des Featurebildes und des Suchbereichs als Rechtecke abgespeicher. d. h. für jedes Rechteck gibt es vier Integer-Werte, in denen jeweils die x- und y-Koordinaten der linken oberen Ecke und der rechten unteren Ecke des Rechtecks abgespeichert werden.
Zusätzlich zu diesen Koordinaten wird noch ein Vertrauenswert abgespeichert, der für jedes Verfahren angeben soll, inwieweit diesem vertraut werden darf. Dabei ist der Vertrauenswert für das Snake-Verfahren entweder 1 oder 0, wobei 1 dafür steht, dass das Verfahren gute Werte liefert, d. h. die Zelle wurde optimal wiedergefunden, der Wert 0 dagegen zeigt an, dass das Verfahren versagt hat. Das Blockvergleichverfahren liefert Werte für den Vertrauenswert zwischen 0 und 1. Werte nahe bei 1 geben dabei an, dass das Verfahren gut funktioniert hat und die Zelle mit sehr großer Wahrscheinlichkeit wiedergefunden wurde. Dagegen geben Werte kleiner $0,5$ an, dass das Verfahren mit ziemlicher Sicherheit die Zelle nicht optimal

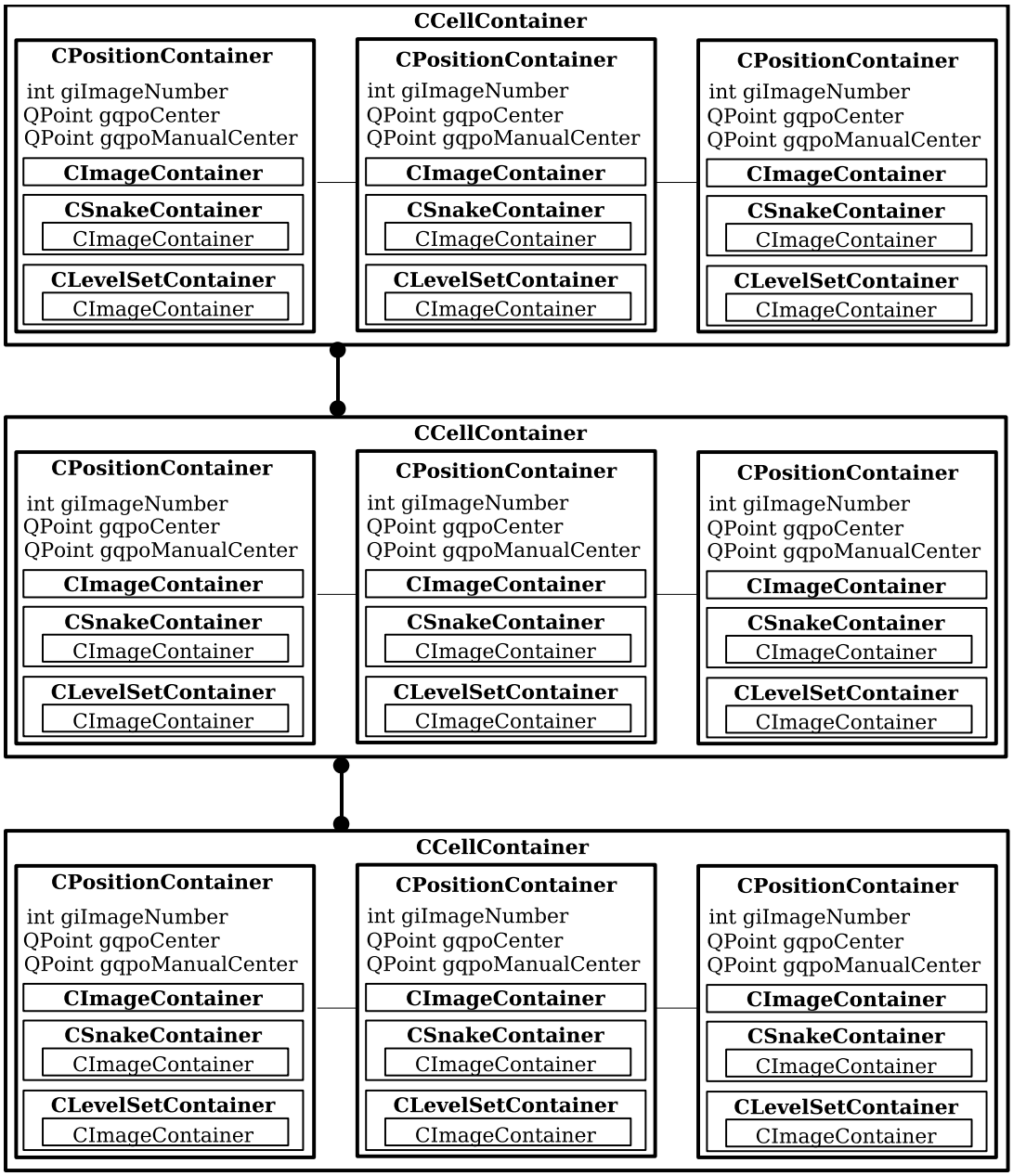


Abbildung 4.4: Schematische Darstellung der CellList

wiedergefunden hat. Dies ist z. B. der Fall, wenn nur noch ein Teil der Zelle im Featurebild liegt. Und Werte nahe bei 0 geben an, dass die Zelle sehr wahrscheinlich verloren wurde und nur noch Teile vom Hintergrund im Featurebild liegen. Auch das LevelSet-Verfahren berechnet Vertrauenswerte. Auch hierbei gibt der Wert 1 an, dass die Zelle mit sehr großer Wahrscheinlichkeit optimal wiedergefunden wurde.
Diese Werte sind nun für den TrackingController (vgl. Kapitel 4.5) wichtig, weil dieser anhand dieser Werte den einzelnen Verfahren vertraut oder nicht. Vertrauen heißt hierbei, dass das Ergebnis des Verfahrens als richtig betrachtet wird und vollständig in die weiteren Be-

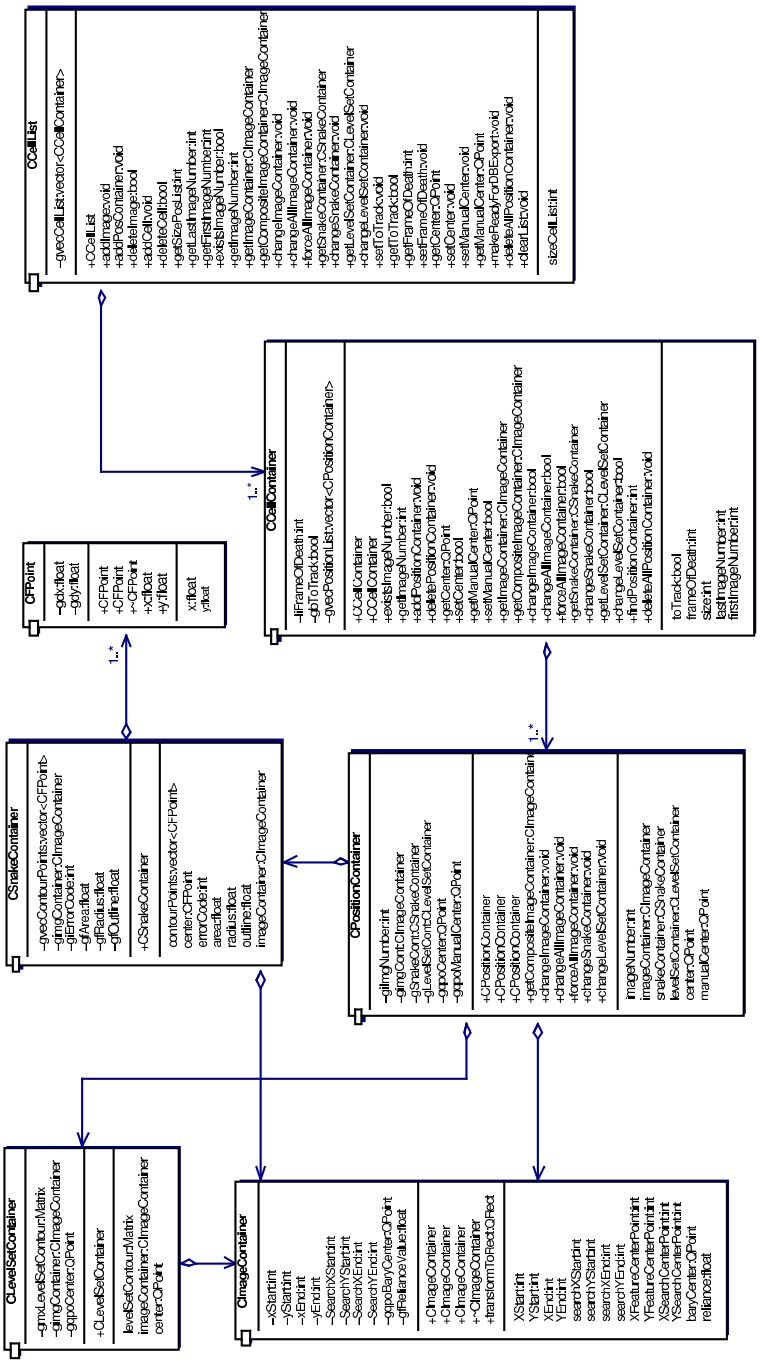


Abbildung 4.5: Klassendiagramm der CellList

rechnungen mit einfließt.
Zusätzlich dazu wird noch ein Mittelpunkt und der Zellschwerpunkt gespeichert. Der Mittelpunkt gibt den Mittelpunkt des Featurebildes an und wird berechnet, sobald der ImageContainer gesetzt wird. Der Zellschwerpunkt dagegen wird von den einzelnen Verfahren berechnet, dieser stimmt nicht zwangsläufig mit dem Mittelpunkt des Featurebildes überein.

Klasse CSnakeContainer

Der `SnakeContainer` stellt die Datenstruktur für die Speicherung der Tracking-Informationen der Snake dar. Für das Snake-Verfahren reichen alleine die Daten, die im `ImageContainer` gespeichert werden nicht aus, um eine Zelle zu tracken. Damit dieses Verfahren vernünftig arbeiten kann, braucht es die Konturpunkte der Snake. Die Konturpunkte werden in einem Vektor gespeichert, dies garantiert schnellen Zugriff auf die einzelnen Konturpunkte. Die Konturpunkte selber werden als `CFPoint` gespeichert. Ein `CFPoint` enthält eine x- und eine y-Koordinate vom Typ float. Somit können die Konturpunkte durch diesen Datentyp in dem Vektor gespeichert werden.
Um das Verhalten der Snake beurteilen zu können, benötigt man weitere Angaben über die Snake. Das sind der Umfang der Snake, der Flächeninhalt der selektierten Zelle und der Radius. Diese Werte werden auch in dem `SnakeContainer` gespeichert.

Klasse CLevelSetContainer

Der `LevelSetContainer` ist ein spezieller Container für das LevelSet-Verfahren. Dieses Verfahren benötigt für das Segmentieren der Zelle und das anschließende Tracken dieser Zelle eine Matrix, in der alle relevanten Informationen über die Zelle abgespeichert werden. Der `LevelSetContainer` enthält also zusätzlich zu dem `ImageContainer`, der wieder die Koordinaten des Feauturebildes speichert, die von diesem Verfahren berechnet werden, eine Matrix. In dieser Matrix wird die Kontur der Zelle abgespeichert. Diese Matrix, die in einem Frame berechnet wird, ist Grundlage für die Berechnung der Kontur im nächsten Frame.
Zusätzlich dazu wird noch der Mittelpunkt der Zelle abgespeichert. Jedes Verfahren berechnet unabhängig von den anderen Verfahren einen Mittelpunkt der Zelle. Deshalb enthält jeder spezielle Container einen eigenen Mittelpunkt für die Zelle.

Klasse CPositionContainer

Der `PositionContainer` fasst alle Informationen der einzelnen Tracking-Verfahren in einem Container zusammen. In ihm enthalten ist also der Container für das Blockvergleichverfahren, der Container für das Snake-Verfahren und der Container für das LevelSet-Verfahren. Ein `PositionContainer` wird für jedes Bild, in dem die entsprechende Zelle getrackt wird, erstellt. Er ist über die Bildnummer eindeutig zu identifizieren. Somit hat man für jedes Bild einen einfachen Zugriff auf alle drei Verfahren. Die Tracking-Informationen aller drei Tracking-Verfahren werden an einer Stelle gespeichert, dies vereinfacht das Auslesen der Daten.
Zusätzlich enthält dieser Container einen Mittelpunkt, der für das Tracken per Hand benötigt wird. Tracken per Hand bedeutet, dass der Benutzer mit dem Mauszeiger die Zelle am Bildschirm verfolgt. Die Position des Mauszeigers wird abgespeichert. Dafür wird der Datentyp QPoint verwendet, in dem die x- und y-Koordinate der Position gespeichert wird.

Klasse CCellContainer

Wird eine Zelle von dem Benutzer markiert, so wird für diese ein `CellContainer` angelegt. Dieser `CellContainer` wird in der CellList an der letzten Stelle eingefügt. Zusätzlich dazu wird ein `PositionContainer` erstellt und die Koordinaten des markierten Bereichs in den `ImageContainer` geschrieben.
Die CellList ist ein Vektor, Grundlage hierfür ist die Standard Template Library. Die Benutzung eines Vektors erlaubt schnellen Zugriff auf die einzelnen in ihm gespeicherten Elemente, in diesem Fall auf die `CellContainer`.

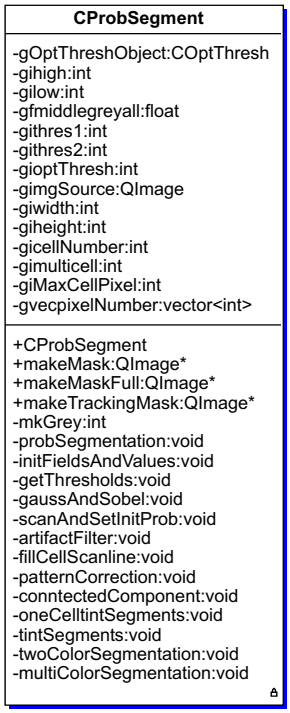


Abbildung 4.6: Klassendiagramm des Paketes Segmentierung

Insgesamt besteht also die CellList aus einer Liste von `CellContainern`. In diesen `CellContainern` ist jeweils eine Liste an `PositionContainern` enthalten und diese `PositionContainer` enthalten letztendlich die Tracking-Informationen der einzelnen Verfahren, die getrennt nach den Verfahren in den jeweiligen Containern gespeichert werden.

Klasse CFPoint
Diese Klasse enthält einen Vektor mit zwei Fließkommazahlen, die die x- und y-Koordinate eines Punktes darstellen. Die Koordinaten können gesetzt und zurückgegeben werden.

Klasse CCell
In dieser Klasse werden die zellbezüglichen Daten gespeichert, die in der GUI angezeit werden sollen. Dazu zählt eine Liste von Linien, die den Pfad der Zelle darstellen, und die rechteckige Markierung um die Zelle.

Tracking und Segmentierung

Das Herzstück des Projektes sind die Trackingklassen, die in Abbildung 4.7 abgebildet werden. Abbildung 4.6 zeigt die einzige Klasse der Segmentierung.

Klasse CProbSegment
Die Klasse enthält Methoden, die eine Segmentierung auf einem Eingabebild zurückliefern. Dieser Segmentierungsansatz vereinigt viele Segmentierungsideen in einer Klasse. So gibt es beispielsweise Methoden, um die Anfangswahrscheinlichkeit einer Zelle mittels der Scanline zu bestimmen. Vergleiche dazu Kapitel 3.3.2.

Klasse CExtEnergy
Die Klasse kapselt die Methoden, die für die externe Energie benötigt werden. Zentral ist die

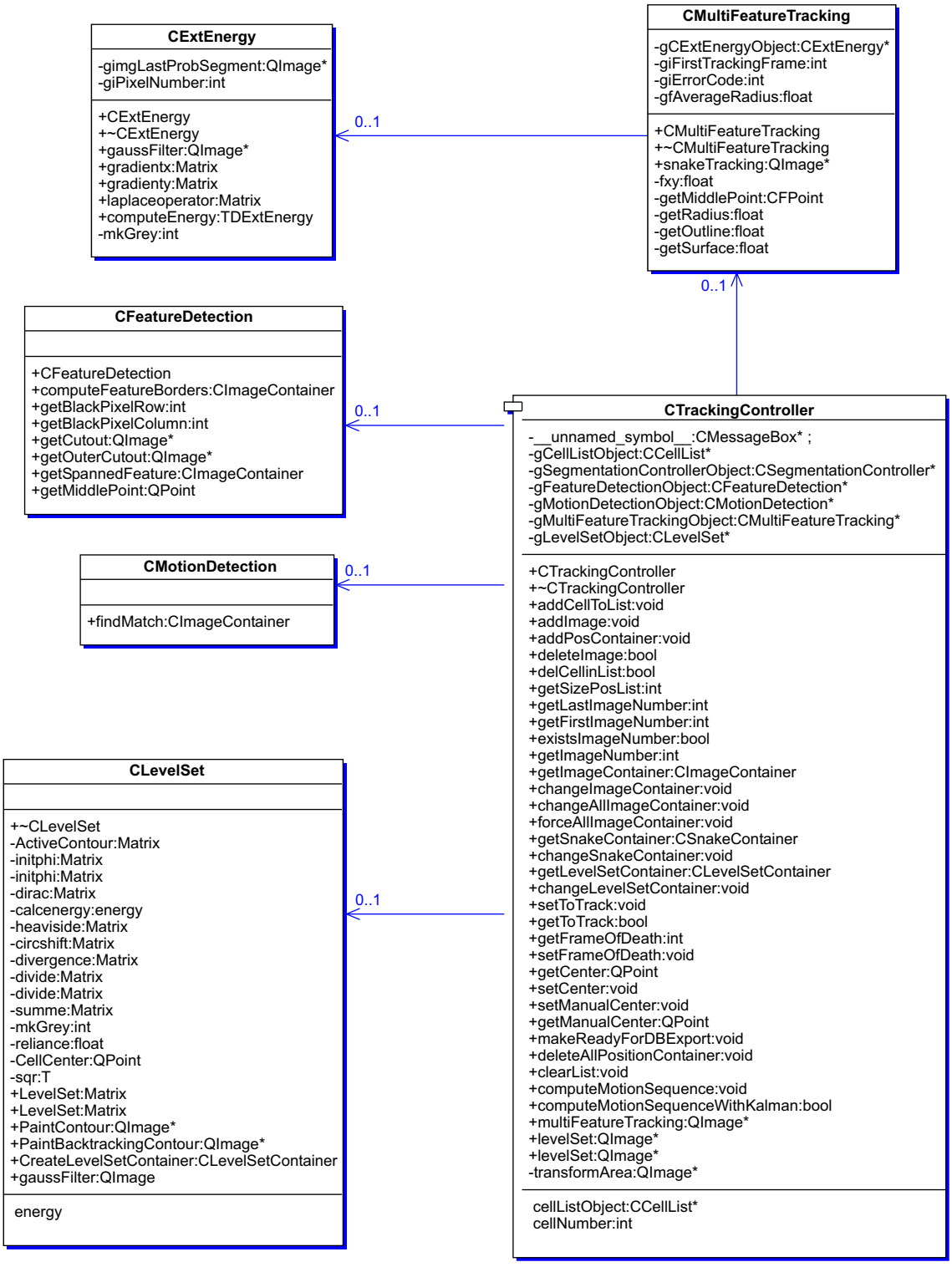


Abbildung 4.7: Klassendiagramm des Paketes Tracking

Methode *computeEnergy*, die die Berechnung einer externen Energie anstößt. Eine Methode für die Berechnung des Gaussfilters und des Laplace-Operators finden sich ebenfalls in dieser Klasse, da sie als Hilfsmethoden für die externe Energie gebraucht werden.

Klasse CFeatureDetection
Die wichtigste Methode in `FeatureDetection` ist *computeFeatureBorders*. Diese Methode berechnet die Ausdehnung des Suchbereichs in alle vier Richtungen. Ziel ist es, schwarze Zeilen und Spalten zu finden, um so die Ausdehnung eingrenzen zu können.

Klasse CLevelSet
`LevelSet` stellt eines der drei Haupttrackingverfahren dar. Die wichtigste Methode dieser Klasse ist *activeContour*. In der Methode wurde der Algorithmus von Level-Set umgesetzt. Die Energien werden direkt in dieser Klasse berechnet.

Klasse CMotionDetection
Die Klasse `MotionDetection` realisiert das Blockvergleichverfahren und besteht lediglich aus der Methode *findMatch*. In diesem Verfahren wird versucht, den Featurebereich im Suchbereich wiederzufinden. Dazu werden alle Bildpunkte des Featurebereichs mit den Bildpunkten des Suchbereichs verglichen. Das Blockvergleichverfahren ist das Haupttrackingverfahren.

Klasse CMultiFeatureTracking
Bei dem dritten Trackingverfahren handelt es sich um die Snake. In `MultiFeatureTracking` wird der Algorithmus der Snake umgesetzt. Die Hauptmethode der Klasse ist *snakeTracking*, in der der Algorithmus angestoßen wird.

Klasse CTrackingController
Die Aufrufe der Trackingalgorithmen werden in dieser Klasse gesteuert.

Datenbank

Die Datenbankklassen werden in Abbildung 4.8 dargestellt. Da diese vier Klassen alle direkt mit der Klasse `MasterController` verbunden sind, haben sie untereinander keine Verbindung.

Klasse CDataExport
`DataExport` stellt eine Exportmöglichkeit für Trackingdaten zur Verfügung. Die Initialkonturen einer Zelle sowie ihre Positionen in den Frames werden in eine Datei geschrieben.

Klasse CLocationChange
Diese Klasse ermöglicht es, die Angabe des Speicherorts eines Experiments in der Datenbank zu ändern. Die Experimente liegen als Liste vor. Die Mediennummer und der Pfad können geändert werden.

Klasse CSQLController
Diese Kontrollklasse stellt die Verbindung zur SQL-Datenbank her.

Klasse CUntrackedList
Mit dieser Klasse können aufgenommene Videos angezeigt werden, die bisher noch nicht getrackt wurden.

Statistik

Die Statistikklassen werden in Abbildung 4.9 aufgezeigt.

Klasse CColoredTableItem
Je nach Ergebnis der Auswertung werden die Ergebnisse farbig dargestellt. Dies übernimmt diese Klasse, daher ist die einzige Methode dieser Klasse die Methode *paint*.

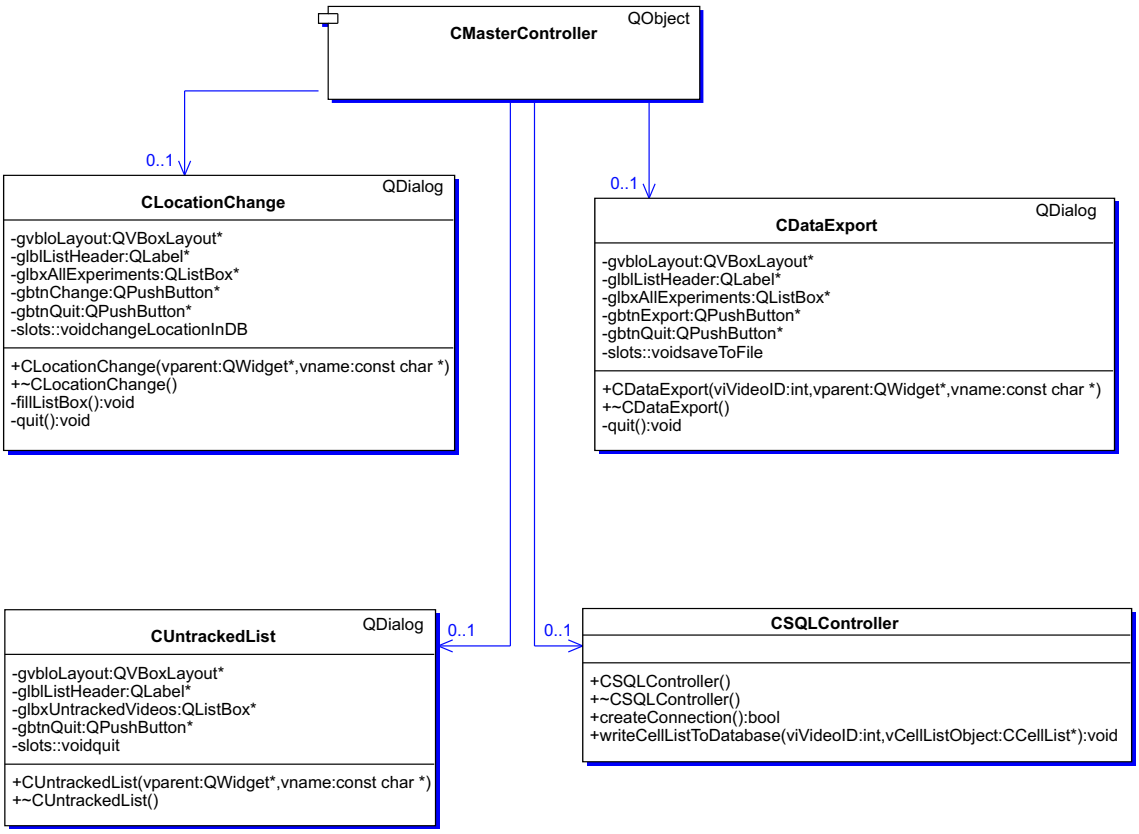


Abbildung 4.8: Klassendiagramm der Datenbankklassen

Klasse CGraphCNVV
Diese Klasse erbt von `CanvasView`, überschreibt die Mouse-Events und stellt zwei Linien für die Markierung zur Verfügung.

Klasse CGraphDialog
Auf dieser Seite wird der steady-state-Graph angezeigt. Der Benutzer bestimmt dann per Hand den steady-state. Die Klasse erbt von `GraphCNVV`.

Klasse CNameSelectDialog
Diese Klasse erbt von `QWidget` und ist die erste Seite des Statistik-Wizards. Hier kann der Benutzer wählen, ob er nur Experimente unter einem bestimmten Namen oder Experimente aller Benutzer sehen will.

Klasse CResultDialog
`ResultDialog` erbt von `QTable` und ist die dritte Seite des Statistik-Wizards. Hier werden in Form einer Tabelle die berechneten Werte angezeigt.

Klasse CSelectionDialog
`SelectionDialog` erbt von `QWidget` und ist die erste Seite des Statistik-Wizards. Hier werden in Form von Tabellen dem Benutzer die Möglichkeit gegeben Versuche aus der Datenbank auszuwählen. Ebenfalls werden hier wichtige Parameter der Auswertung gesetzt.

Klasse CStatController
Die Klasse `StatController` übernimmt die Koordination der Statistik-Methoden.

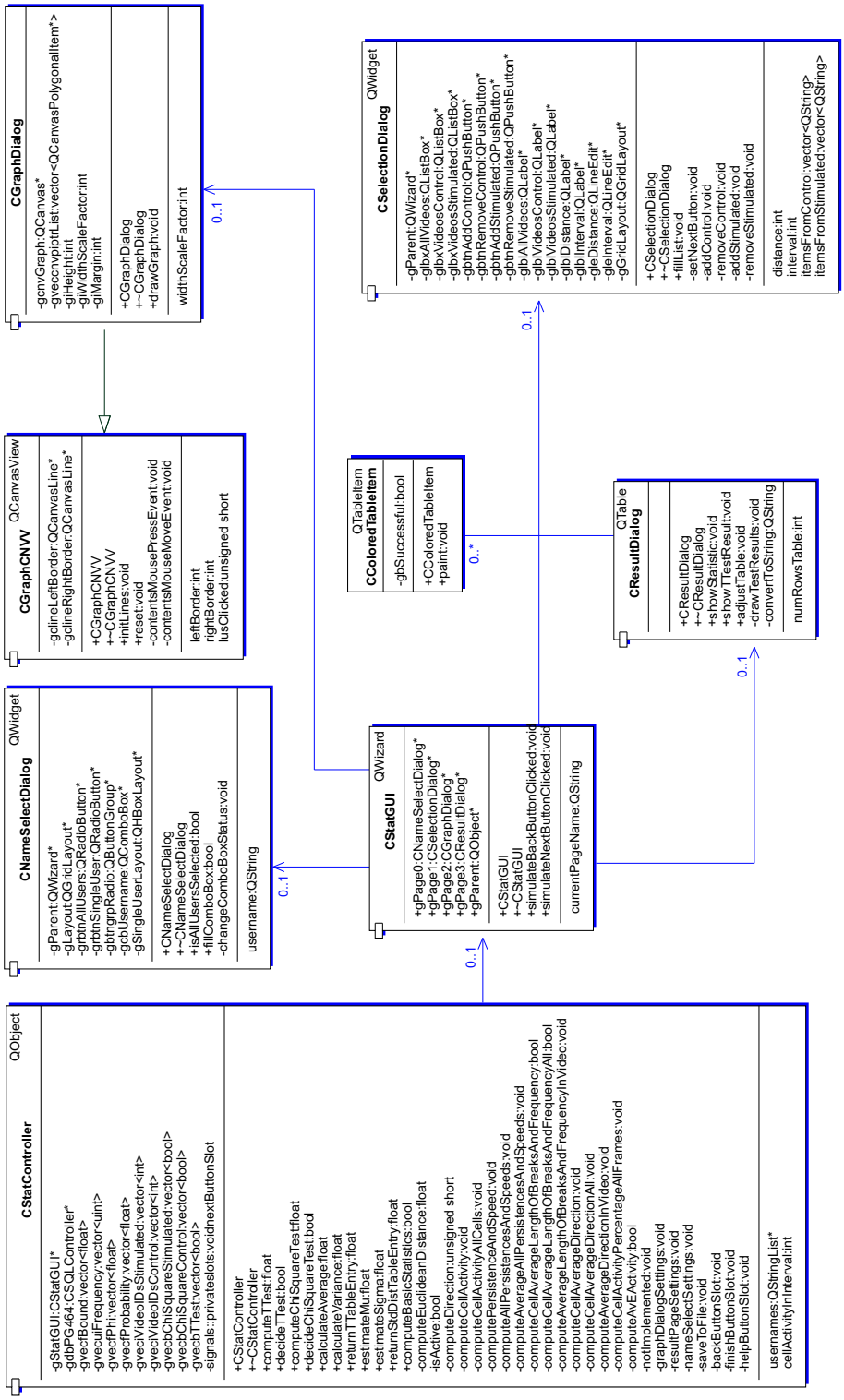


Abbildung 4.9: Abbildung der Klassen des Statistikteils

Klasse CStatGUI
Diese Klasse stellt den Wizard für die statistische Datenauswertung zur Verfügung.

Die einzelnen Dialoge werden in Kapitel 4.6 näher erläutert.

4.3 Aktivitätsdiagramme

4.3.1 Gesamtübersicht

Abbildung 4.10 stellt die grundsätzliche Funktionalität des Projektes dar. Nach dem Start des Programms muss ein Benutzer ausgewählt werden. Danach kann der Benutzer grundsätzlich wählen, ob er eine statistische Auswertung starten möchte, oder ein aufgenommenes Video laden will. Ein geladenes Video kann entweder nur abgespielt oder es können Zellen im Video markiert werden. Die markierten Zellen können dann verfolgt werden.

4.3.2 Tracking

Möchte der Benutzer ein Video tracken, muss er zunächst ein Video laden und darin eine oder mehrere Zellen markieren. Dann hat er die Wahl zwischen manuellem und automatischem Tracking. Wenn das automatische Tracking gestartet wird, kann der Benutzer das Video pausieren und entweder weiter manuell tracken oder automatisch fortfahren, bis das Video vollständig getrackt ist. Beim manuellen Tracking muss erst eine Zelle aus der Liste ausgewählt werden und durch einen Mausklick in die Mitte dieser Zelle wird das manuelle Tracking gestartet. Nun wird das Video abgespielt und der Benutzer muss den Cursor in der Mitte der Zelle halten. Nachdem eine Zelle vollständig manuell getrackt wurde, kann entweder die nächste Zelle getrackt oder das Tracking beendet werden. Diesen Ablauf zeigt das Aktivitätsdiagramm in Abbildung 4.11.

4.3.3 Statistik

Wenn der Benutzer eine statistische Auswertung startet, muss er zunächst auswählen, ob alle getrackten Videos angezeigt werden sollen oder nur die eines einzelnen Benutzers. Die auszuwertenden Videos können dann in die Gruppen „stimulated“ und „control“ eingeordnet werden. Nun kann ein Start- und ein Zielzeitpunkt eingestellt werden. Nach der Auswertung kann die Statistik abgespeichert oder beendet werden. Das Aktivitätsdiagramm in Abbildung 4.12 zeigt diesen Ablauf.

4.4 Kontrollfluss

Die Hauptkontrolle über das Projekt hat die Klasse `MasterController`. Von dort aus werden alle Aktivitäten angestossen, die der Benutzer ausführen kann. Dazu gehören z.B. das Abspielen des Videos, das Markieren der Zellen und das Initialisieren der Datenbank. Es werden Slots zur Verfügung gestellt, die im weiteren Programmablauf genutzt werden. Von der Klasse `MasterController` wird der Fluss an die anderen Kontrollklassen weitergeleitet. Dies sind die Klassen: `GuiController`, `SQLController`, `TrackingController`, `StatController`. Z. B. werden die Eingaben des Benutzers mittels Signals aus dem `GuiController` mit den Slots der Klasse `MasterController` verbunden. Dadurch das an die Slots nur sinnvolle Parameter übergeben werden, erübrigt sich eine Ausnahmebehandlung für Benutzereingaben. Fehl- und NULL-Eingaben können bei diesem Modell fast nicht entstehen. Anders verhält es sich beim `TrackingController`. Die Trackingmethoden können allerdings Fehleingaben produzieren. Diese werden mit einer TrackingExecption im `TrackingController` abgefangen. In der Testphase des Projekts wurden alle Methoden auf ihr Verhalten bei ungewöhnlichen Eingaben getestet. Jedes Verhalten bei Fehleingaben wurde so korrigiert, dass Programmabstürze fast nicht auftreten können.

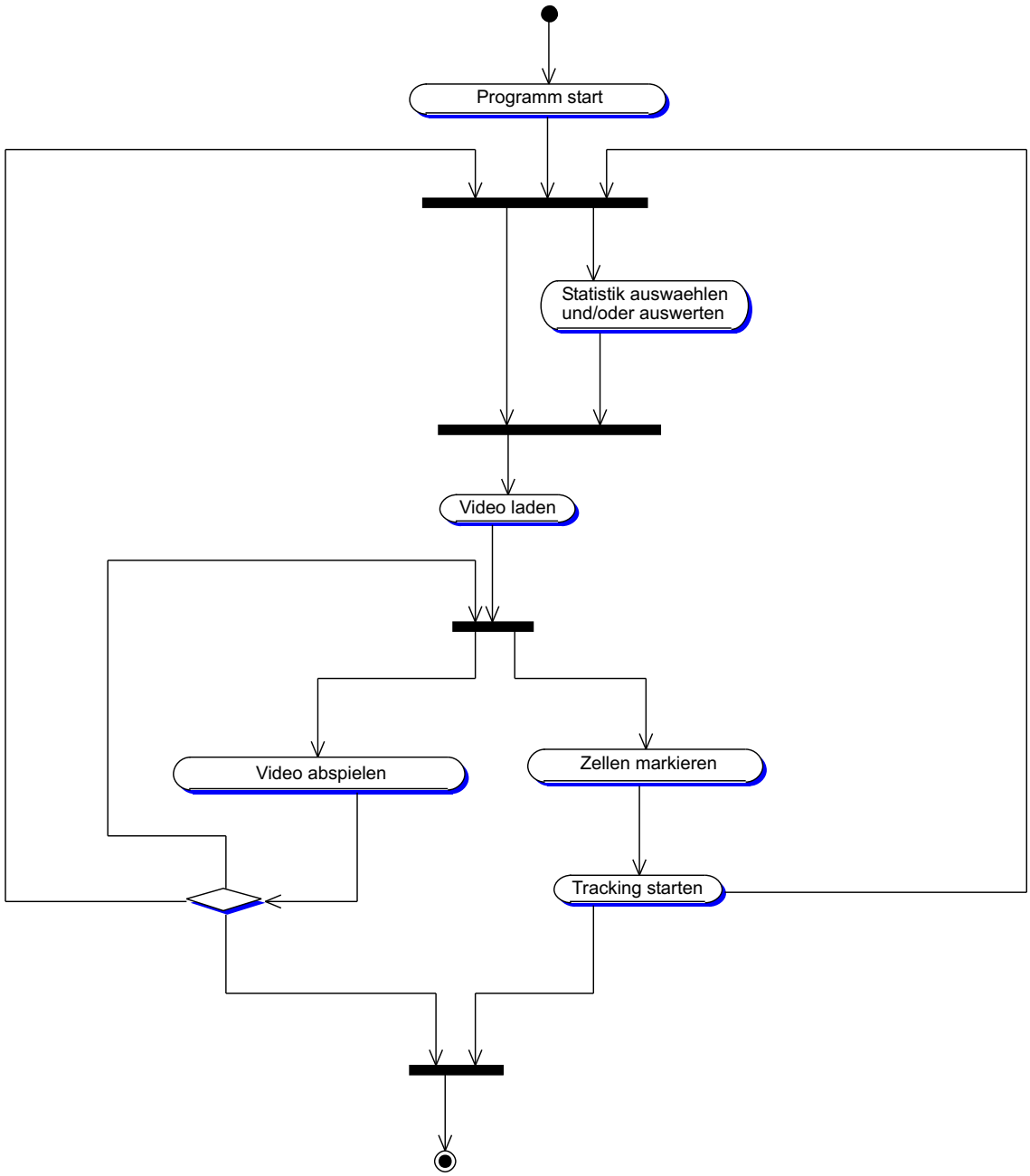


Abbildung 4.10: Aktivitätsdiagramm um die grundsätzliche Funktionalität darzustellen

4.5 TrackingController

Der `TrackingController` ist die zentrale Kontrollklasse des Trackings. Eine solche Klasse ist notwendig, da das Tracking auf drei verschiedenen Verfahren basiert, dem Blockvergleichverfahren, dem Snake-Verfahren und dem Level-Set-Verfahren. Er kombiniert die einzelnen Ergebnisse zu einem Mittelpunkt, da zur Beschreibung der Bewegung einer Zelle die Position des Mittelpunktes ausreicht. Die einzelnen Verfahren liefern beim Tracken eine Menge an Informationen, aus denen der `TrackingController` den Mittelpunkt berechnet. Neben der Berechnung ist auch die Überprüfung

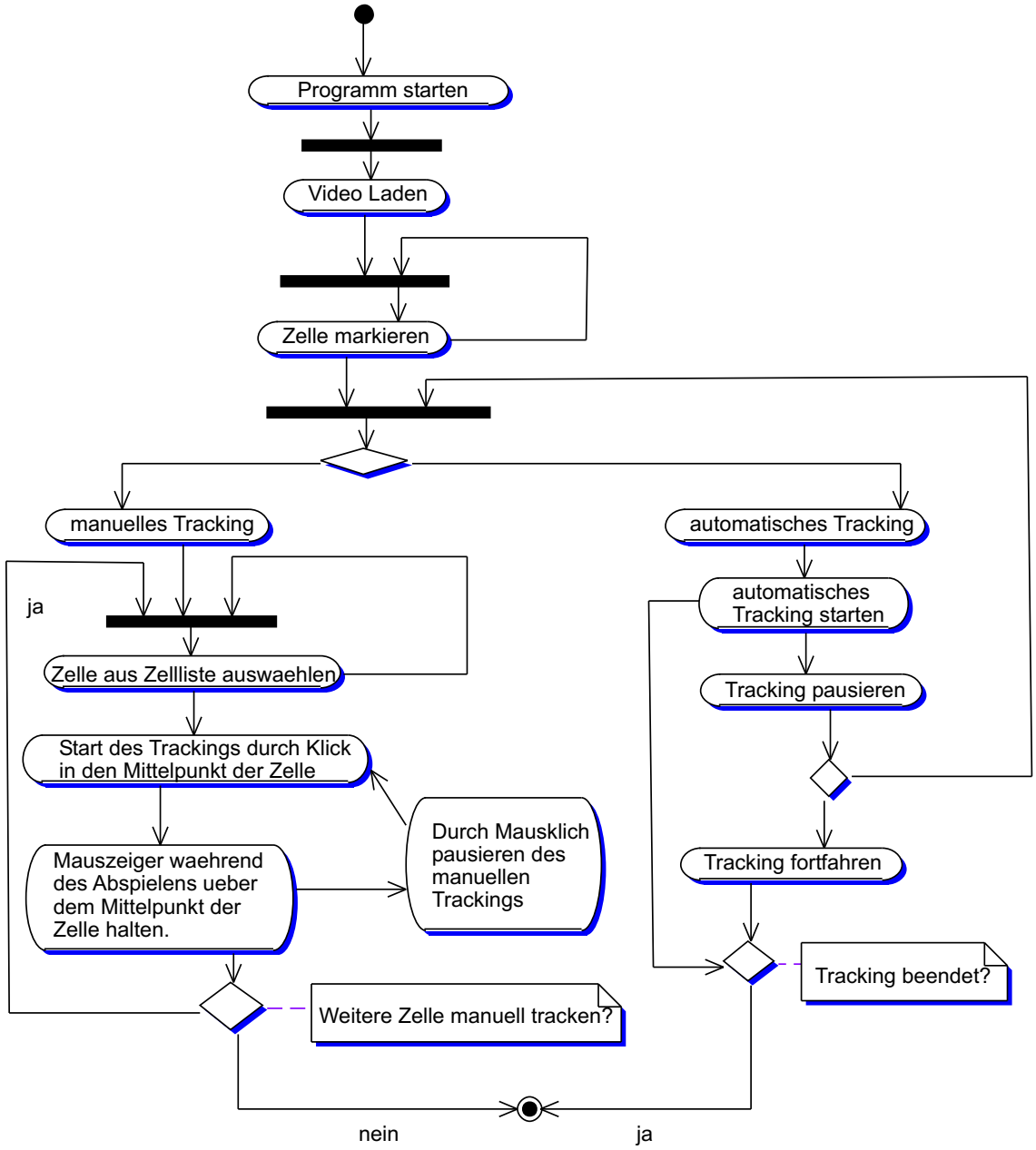


Abbildung 4.11: Aktivitätsdiagramm zum Tracking

der Korrektheit der Verfahren eine zentrale Aufgabe des `TrackingControllers`. Zusätzlich hat der `TrackingController` im Laufe der Zeit die Aufgabe erhalten, möglichst effizient zu arbeiten und keine überflüssigen Informationen zu erzeugen. Da vor allem das Level-Set-Verfahren erhebliche Laufzeitprobleme verursacht, ist der `TrackingController` dazu angehalten, dieses Verfahren möglichst selten aufzurufen.

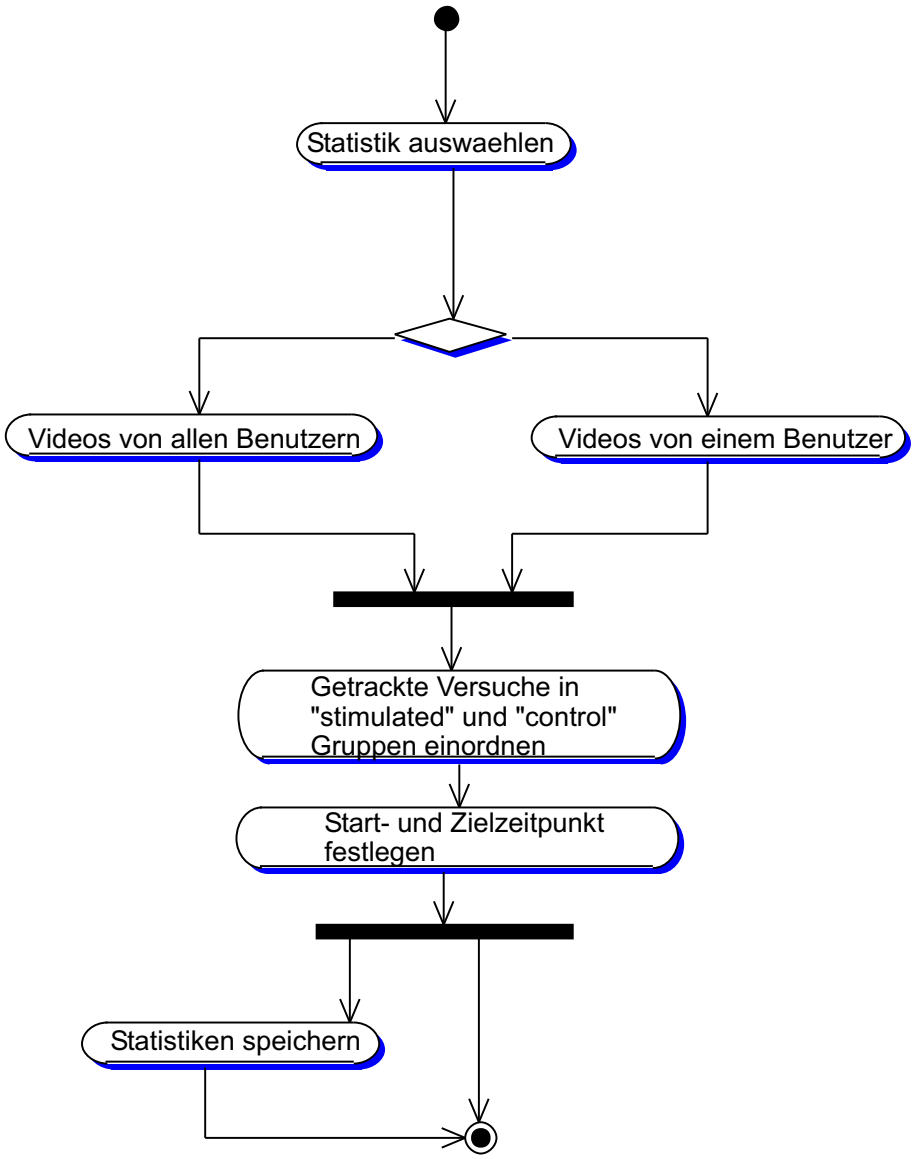


Abbildung 4.12: Aktivitätsdiagramm zur statistischen Auswertung

4.5.1 Bewertung der Tracking-Verfahren im `TrackingController`

Der `TrackingController` beurteilt die einzelnen Verfahren grundsätzlich nach deren Vertrauenswerten. Die Vertrauenswerte werden für jedes Verfahren gesetzt und geben an, in wie weit man den Ergebnissen der Verfahren vertrauen kann. Jedes Verfahren setzt seinen Vertrauenswert auf Grund der spezifischen Tracking-Informationen, des aktuellen und der vorhergehenden Zeitschritte. Die Werte hierfür liegen zwischen 0 und 1, wobei ein Wert nahe bei 1 angibt, dass das Ergebnis mit großer Wahrscheinlichkeit richtig ist. Ein Wert nahe bei 0 dagegen gibt an, dass das gelieferte Ergebnis nicht brauchbar ist und verworfen werden sollte. Dabei liefert das Blockvergleichverfahren viele differenzierte Werte zwischen 0 und 1 (vgl. Kapitel 3.4.3) und das Snake-Verfahren liefert, ähnlich wie das Level-Set-Verfahren (vgl. Kapitel 3.4.4), nur wenige diskrete Werte (vgl. Kapitel 3.4.1).

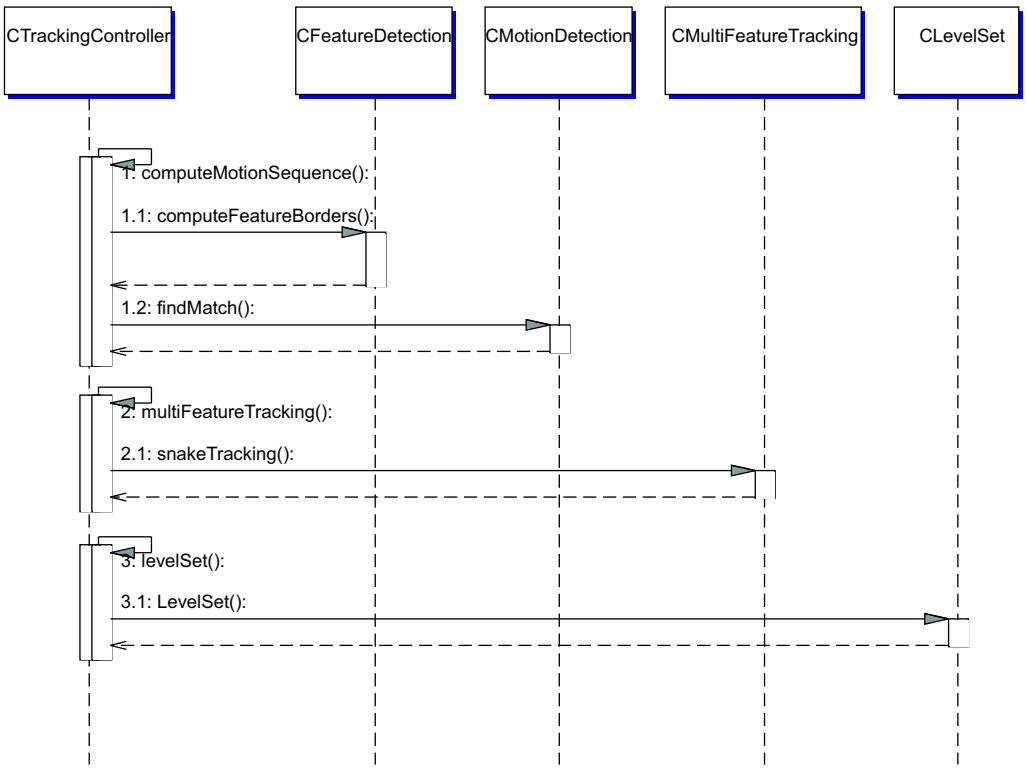


Abbildung 4.13: Sequenzdiagramm für den TrackingController.

Die Verwendung dieser Werte erwies sich als schwierig, da kein einheitliches Kriterium für das Versagen verwendet werden konnte. Für jedes Verfahren musste ein eigenes Versagenskriterium gefunden werden. Bei dem Snake-Verfahren z. B. war dies einfach, da der Wert 1 Erfolg und Wert 0 Versagen angibt, wohingegen die Grenze für Versagen beim Blockvergleichverfahren schwieriger zu bestimmen war und letztendlich auf $0,4$ festgelegt wurde.

4.5.2 Kombination der Verfahren

Auf Grund der Laufzeitschwierigkeiten des Level-Set-Verfahrens mussten zwei Kombinationsmethoden entwickelt werden, wobei eins nur das Snake-Verfahren und das Blockvergleichverfahren miteinander kombiniert, und das andere alle drei Verfahren kombiniert. Somit konnten die Aufrufe des Level-Set-Verfahrens auf ein Minimum reduziert werden, ohne dabei allzu große Genauigkeitsverluste der getrackten Position zu erhalten. Das Sequenzdiagramm in Abbildung 4.13 zeigt die Arbeitsweise des TrackingControllers. Das Blockvergleichverfahren und das Snake-Verfahren werden in jedem Fall aufgerufen, das Level-Set-Verfahren dagegen nur, wenn bestimmte Voraussetzungen erfüllt sind.
Zusätzlich wurde ein Mittelpunktschätzer implementiert, der aus den Bewegungen der vorhergehenden Zeitschritte eine Mittelpunktposition schätzt. Dieser wird für die Beurteilung der einzelnen Ergebnisse verwendet, indem die Berechnung wie ein zusätzliches Trackingverfahren behandelt wird, mit dem die Ergebnisse verglichen werden können.

Um entscheiden zu können, welche dieser beiden Kombinationen verwendet wird, wird zuerst der Abstand der errechneten Mittelpunkte des Snake-Verfahrens und des Blockvergleichverfahrens berechnet. Danach erfolgt eine Berechnung der Abstände zum geschätzten Mittelpunkt. Die Kombination des Snake- und des Blockvergleichverfahrens wird dann aufgerufen, wenn folgende

Bedingungen erfüllt sind:

- Die Differenz zwischen den Mittelpunkten der beiden Verfahren ist höchstens so groß wie der Zellradius,
- die Mittelpunkte der Verfahren weichen höchstens um den Zellradius vom geschätzten Mittelpunkt ab und
- die Vertrauenswerte liegen über einem Wert von 0,4.

Sind diese Bedingungen erfüllt, wird das Ergebnis der beiden Verfahren zu einem Ergebnis zusammengefasst, indem ein gewichteter Mittelpunkt berechnet wird. Um die Gewichtung γ zu bestimmen, werden die Differenzen der beiden errechneten Mittelpunkte mit dem geschätzten Mittelpunkt verglichen. Ist die Differenz zwischen dem errechneten Mittelpunkt des Blockvergleichverfahrens und dem geschätzten Mittelpunkt kleiner als die entsprechende Differenz des Snake-Verfahrens, so wird γ auf 0,85 gesetzt. Ist darüber hinaus der Vertrauenswert des Blockvergleichverfahrens gleich 0, so wird γ auf 0 gesetzt. Dies bewirkt, dass das Ergebnis des Blockvergleichverfahrens nicht in die weitere Berechnung einfließt. Ist die Differenz des Blockvergleichverfahrens dagegen größer als die Differenz des Snake-Verfahrens, so wird γ auf 0,15 gesetzt. Zusätzlich wird auch hier überprüft, ob der Vertrauenswert des Blockvergleichverfahrens gleich 0 ist. Ist dies der Fall, wird γ auch auf 0 gesetzt.
Eine größere Differenz heißt, dass das Ergebnis weit vom geschätzten Mittelpunkt abweicht und nach unserer Definition dementsprechend schlechter ist. Deshalb fließt das Ergebnis, das einen kleineren Abstand zum geschätzten Mittelpunkt liefert, stärker in die weitere Berechnung ein. Um den gewichteten Mittelpunkt nun zu berechnen, wird der errechnete Mittelpunkt des Blockvergleichverfahrens mit γ multipliziert und darauf wird der errechnete Mittelpunkt des Snake-Verfahrens, multipliziert mit $1 - \gamma$, addiert. Man erhält somit den Mittelpunkt der Zelle, der durch Kombination dieser beiden Verfahren errechnet wurde.

Ist eine der drei aufgeführten Bedingungen nicht erfüllt, so wird zusätzlich das Level-Set-Verfahren aufgerufen und ein Mittelpunkt als Kombination der errechneten Mittelpunkte der drei Verfahren bestimmt. Dabei wird als erstes überprüft, ob einer der drei Vertrauenswerte gleich 0 ist. Ist dies bei einem Verfahren der Fall, so wird das Ergebnis dieses Verfahrens in den weiteren Berechnungen nicht mehr berücksichtigt. An Stelle dieses Mittelpunktes tritt dann der vorher geschätzte Mittelpunkt. Als nächstes werden, ähnlich wie bei der Kombination des Blockvergleichverfahrens und des Snake-Verfahrens, die Abstände der drei Mittelpunkte zueinander berechent. Wenn nun alle drei Differenzen null sind, so heißt das, dass alle drei Verfahren den gleichen Mittelpunkt berechnet haben und eine gewichtete Berechnung des Mittelpunktes nicht durchgeführt werden muss. In diesem Fall wird der Mittelpunkt auf den Mittelpunkt eines der drei Verfahren gesetzt.
Ist dies nicht der Fall, müssen geeignete Gewichte für die drei Verfahren gefunden werden. Dafür werden die vorher berechneten Abstände verwendet. Die Abstände werden mit $\Delta_j, j \in \{0, 2\}$ bezeichent, wobei Δ_0 für den Abstand des Mittelpunktes des Blockvergleichverfahrens zum Punkt zwischen den Mittelpunkten der beiden anderen Verfahren steht. Δ_1 steht analog für den Abstand des Mittelpunktes des Snake-Verfahrens zum Mittelpunkt zwischen den Mittelpunkten vom Blockvergleich- und Level-Set-Verfahren und Δ_2 entsprechend für den Abstand des Mittelpunktes des Level-Set-Verfahrens. Wie diese Abstände nun miteinander verrechnet werden, zeigt Formel 4.1.

$$\gamma_i = \frac{\sum\limits_{j=0, j\neq i}^{2} \Delta_j}{\sum\limits_{j=0}^{2} \Delta_j} \tag{4.1}$$

Auf diese Weise erhält man nun die Gewichtung für das Ergebnis jedes einzelnen Verfahrens und kann nun damit den gewichteten Mittelpunkt bestimmen.

4.5.3 Korrektur der Verfahren

Nach den Tracking-Berechnungen können die Verfahren korrigiert werden, so dass diese im nächsten Schritt die Möglichkeit haben, wieder brauchbare Ergebnisse zu liefern. Diese Korrektur wird allerdings nur dann aufgerufen, wenn mindestens eins der Verfahren den Vertrauenswert 1 liefert oder die Summe der drei Vertrauenswerte größer als 1,3 ist. Diese Einschränkung ist sinnvoll, da nur dann ein Verfahren korrigiert werden kann, wenn mindestens ein Verfahren vernünftige Ergebnisse liefert und diese für die Korrektur verwendet werden können. Haben alle drei Verfahren versagt, so ist eine Korrektur nicht möglich.
Wenn das Snake-Verfahren einen Vertrauenswert von 0 liefert, wird dieses mit den Werten des Level-Set-Verfahrens korrigiert. Dies geschieht allerdings nur, wenn das Level-Set-Verfahren einen Vertrauenswert größer 0,5 liefert, sonst werden zum Korrigieren die Ergebnisse des Blockvergleichverfahrens verwendet. Dabei wird der `ImageContainer` ausgelesen und die Werte in den `SnakeContainer` geschrieben. Zusätzlich dazu wird ein leerer Vektor für die Konturpunkte erzeugt und in den `SnakeContainer` gehängt. Dies veranlasst das Snake-Verfahren dazu, die Snake im nächsten Schritt neu aufzubauen.
Das Blockvergleichverfahren wird nur dann korrigiert, wenn der Vertrauenswert kleiner als 0,5 ist. Zum Korrigieren werden die Ergebnisse des Snake-Verfahrens verwendet, sofern dieses Verfahren einen Vertrauenswert von 1 liefert. Ist dies nicht der Fall, werden die Werte des Level-Set-Verfahrens verwendet. Dafür werden jeweils die `ImageContainer` ausgelesen und die Werte in den `ImageContainer` des Blockvergleichverfahrens geschrieben.
Auch das Level-Set-Verfahren wird korrigiert, wenn sein Vertrauenswert gleich 0 ist. Ist dies der Fall, wird die Matrix auf 0 gesetzt, also eine leere, dimensionslose Matrix übergeben. Dies ist das Zeichen für das Level-Set-Verfahren, das Tracken mit einer neuen Initialkontur fortzusetzen. Zusätzlich dazu wird der `ImageContainer` entweder aus dem `SnakeContainer` ausgelesen, wenn dieses Verfahren den Vertrauenswert 1 liefert, sonst wird der `ImageContainer` des Blockvergleichverfahrens verwendet.

4.6 Statistik

Bei der Implementierung der Statistik wurden im Wesentlichen die Verfahren aus Kapitel 2.7.4 berücksichtigt. Es wurde auf die Implementierung eines nicht-parametrischen Tests verzichtet und nur der t-Test als parametrischer Test und der χ^2-Test zur Überprüfung von Normalverteilungen verwendet.

4.6.1 Die Statistik-GUI

Der Benutzer wird mit Hilfe eines `QWizard`-Dialogs durch die statistische Datenauswertung geführt. Er bekommt nacheinander vier Dialogfelder mit diversen Einstellungsmöglichkeiten angezeigt und kann mit Hilfe von `Next>`- und `Back>`-Buttons zwischen diesen Feldern hin- und herschalten. Nach der Datenbank-Verbindung erfolgt die Auswahl eines Benutzernamens basierend auf der Menge der bereits in der Datenbank vorhandenen Nutzer. Anschließend wird eine dreigeteilte Liste angezeigt. Der Benutzer kann sich hier aus allen verfügbaren Videos eine Gruppe von Videos mit stimulierten Zellen und eine Kontrollgruppe zusammenstellen, wobei die wählbaren Videos abhängig vom Benutzernamen aus der Datenbank eingelesen werden. In der nun folgenden Steady-State-Auswahl werden zwei Graphen angezeigt, in denen die durchschnittliche Migrationsaktivität der beiden Gruppen über der Zeit aufgetragen wird. Mit Hilfe der Maus wählt der Benutzer nun den Zeitraum aus, für den er die statistische Datenanalyse ausführen möchte. Das letzte Dialogfeld des Wizards zeigt in Form einer Tabelle für jedes Video die durchschnittlichen Werte für Migrationsaktivität, Pausenlänge, Pausenfrequenz, Migrationsgeschwindigkeit, Migrationsrichtung und

Persistenz an. Außerdem wird der Benutzer darauf hingewiesen, inwiefern bei den Ergebnissen statistische Signifikanz vorliegt. Um die ermittelten Daten weiterverarbeiten zu können, besteht die Möglichkeit, diese in eine CSV-Datei zu exportieren.

4.6.2 Die Datenstruktur für die statistische Auswertung

Testläufe haben ergeben, dass die für die statistische Auswertung benötigten Daten nicht einzeln aus der Datenbank ausgelesen werden sollten, da sich ansonsten auf Grund der Latenzzeit erhebliche Laufzeitprobleme ergeben. Zur Lösung dieses Problems wurde eine Datenstruktur entwickelt, die den internen Aufbau der Datenbank nachbildet. Zu Beginn der Auswertungsphase erfolgt ein einmaliger Datenbankzugriff, um die relevanten Daten auszulesen und in die Datenstruktur zu schreiben. Die gesamte Auswertung operiert dann auf der Datenstruktur, so dass die Datenbankzugriffe auf ein Minimum reduziert werden konnten.

4.6.3 Ermittlung der Migrationsparameter

Die Berechnung der Migrationsparameter erfolgt auf Grundlage der Angaben von Seiten der Biologen. Lediglich bei der Migrationsaktivität wird davon abgewichen, da hier ein Verfahren benutzt wird, das eine höhere Genauigkeit verspricht. Für die Berechnung der Parameter wird nur der Teil des Videos herangezogen, den der Benutzer zuvor als Steady-State markiert hat. Im Einzelnen erfolgt die Berechnung folgendermaßen:

Migrationsaktivität
: Grundlage für die Berechnung der Migrationsaktivität sind zwei vom Benutzer eingegebene Parameter: Zum einen die Distanz d, um die sich eine Zelle mindestens bewegen muss, um als aktiv zu gelten und zum anderen die Zeitspanne t, innerhalb der diese Bewegung stattfinden muss. Jeder Zelle in jedem Videoframe wird nun ein boolesche Wert zugeordnet, der angibt, ob die Zelle aktiv ist. Für den Frame zum Zeitpunkt t_i erfolgt die Aktivitätsbestimmung innerhalb des Intervalls $[t_i, t_i + t]$. Für jedes Video wird dann innerhalb des Steady-State-Intervalls die prozentuale Aktivität über die Frames gemittelt.
  Bei dem zuvor von den Biologen benutzten Verfahren wurde das Video zunächst in disjunkte Intervalle $I_0, ..., I_n$ der Länge t unterteilt und dann für jede Zelle in jedem Intervall die Aktivität untersucht. Dies birgt jedoch die Gefahr, dass vorhandene Aktivität nicht erkannt wird, wenn z.B. die notwendige Distanz d zur Hälfte in Intervall I_i und zur Hälfte in Intervall I_{i+1} zurückgelegt wird: Insgesamt ist dann Aktivität vorhanden, obwohl die Zelle in jedem einzelnen Intervall als inaktiv gelten würde.

Pausenlänge
: Wenn eine Zelle nicht gemäß der obigen Definition aktiv ist, pausiert sie. Zellen, die über den gesamten Betrachtungszeitraum inaktiv sind, werden bei der folgenden Berechnung nicht berücksichtigt. Zur Ermittlung der durchschnittlichen Pausenlänge wird für jede Zelle die Anzahl der Frames bestimmt, in denen die Zelle inaktiv ist. Diese Anzahl wird in Sekunden umgerechnet. Ausgegeben wird dann für jedes Video das arithmetische Mittel aller Pausenlängen.

Pausenfrequenz
: Neben der durchschnittlichen Pausenlänge ist für die statistische Auswertung auch die Pausenhäufigkeit interessant. Wiederum wird in jedem Video jede nicht komplett inaktive Zelle betrachtet, wobei nun die Anzahl der Intervalle bestimmt wird, in denen die Zelle inaktiv ist. Diese Intervalle liegen entweder am Anfang oder am Ende des Videos oder werden von zwei aktiven Intervallen „eingerahmt“. Für jedes Video wird nun das arithmetische Mittel der Pausenfrequenzen bestimmt.

Migrationsgeschwindigkeit
: Die Migrationsgeschwindigkeit bezieht sich auf die Intervalle, in denen eine Zelle aktiv ist. Es

wird jeweils die zurückgelegte Weglänge zwischen zwei Frames ermittelt. Die Summe dieser Weglängen wird durch die Gesamtdauer der aktiven Beobachtungsintervalle dividiert, so dass sich für jede Zelle eine Durchschnittsgeschwindigkeit ergibt. Das arithmetische Mittel aller Durchschnittsgeschwindigkeiten eines Videos wird ausgegeben.
Eine andere Möglichkeit, Migrationsgeschwindigkeit zu definieren, ist, inaktive Intervalle mit in die Berechnung einzubeziehen. Die Biologen haben früher beide Versionen der Geschwindigkeit berechnet, verwenden heute aber nur noch die o.g. Definition. Daher wurde auf eine Implementierung der anderen Version verzichtet.

Migrationsrichtung
Die Migrationsrichtung einer Zelle kann über einen Winkel ausgedrückt werden. Dabei befindet sich der Startpunkt der Migration im Ursprung eines gedachten Polarkoordinatensystems. Wenn der Winkel ϕ des Migrationsendpunkts in $[0, 90) \cup (270, 360)$ liegt, bewegt sich die Zelle nach rechts, anderenfalls nach links. Für die Auswertung wird pro Video der Prozentsatz an Zellen ermittelt, der sich nach rechts bewegt, da in den Experimenten der Stimulus von rechts hinzugefügt wird.

Persistenz
Persistenz ist ein Maß dafür, wie zielstrebig eine Zelle sich bewegt. Sie ist definiert als Quotient aus dem euklidischen Abstand von Migrationsstart- und -endpunkt und der tatsächlich zurückgelegten Weglänge. Je zielstrebiger sich eine Zelle bewegt, desto mehr konvergiert die Persistenz gegen 1. CellTrack ermittelt die durchschnittliche Persistenz je Video, indem es den arithmetischen Mittelwert der einzelnen Persistenzen bestimmt.

4.6.4 t-Test

Die statistische Datenauswertung verwendet zur Analyse der Tracking-Daten einen ungepaarten, ungerichteten t-Test. Zur Durchführung dieses Tests werden die Formeln aus 2.7.4 verwendet. Aus Gründen der Übersichtlichkeit wurden Nebenrechnungen wie z.B. Bestimmung von Mittelwert und Varianz in private Hilfsmethoden ausgelagert. Für zwei gegebene Datenreihen wird zunächst die Teststatistik berechnet. Dann wird dieser Wert mit dem entsprechenden Quantil der t-Verteilung verglichen, um so die Testentscheidung zu ermitteln. Da CellTrack mit einem festen Testniveau von $\alpha = 0,05$ arbeiten, ist dieser Quantilswert lediglich von dem Umfang der Messreihen abhängig. Somit ergab sich die Entscheidung, eine vorgenerierte Quantil-Tabelle in den Programmcode zu integrieren, da die Berechnung dieses Werts zur Laufzeit zu ineffizient wäre.

4.6.5 χ^2-Test

Da der t-Test die Normalverteilung der Messreihen voraussetzt, testet CellTrack diese Bedingung mittels eines χ^2-Anpassungstests. Streng genommen darf ein t-Test nicht angewandt werden, wenn die Normalverteilungsannahme verletzt ist. Stattdessen müsste dann ein nicht-parametrischer Signifikanztest benutzt werden. Aus Zeitgründen wurde darauf verzichtet, einen solchen Test zu implementieren, allerdings erscheint eine Warnung, wenn der χ^2-Test fehlschlägt.
Die Implementierung des χ^2-Tests ist etwas komplizierter als die Umsetzung des t-Tests, da die Messwerte hier zuvor in Klassen eingeteilt werden müssen. Im Rahmen des CellTrack-Kontexts scheint eine Einteilung in 6 Klassen sinnvoll. Begründungen für diese Einteilung sollen hier nicht weiter erörtert werden, weiterführende Informationen sind z.B. in [Käh95] zu finden. Die Klasseneinteilung wird auf Grundlage der geschätzten Werte für Mittelwert und Standardabweichung vorgenommen. Klasse i umfasst dabei die Messwerte im Intervall $[\mu + (i-3) \cdot \sigma, \mu + (i-2) \cdot \sigma)$. Sollte es Messwerte geben, die außerhalb dieser Klassen liegen, so werden sie für die weitere Berechnung nicht berücksichtigt. Nach der Klassierung der Daten verfährt unser Algorithmus so, wie in Kapitel 2.7.4 angegeben, d.h. er ermittelt die Teststatistik und vergleicht diesen Wert mit dem entsprechenden Quantilswert der χ^2-Verteilung. Im Gegensatz zum t-Test ist dieser Quantilswert nicht vom Umfang der Messreihe sondern von der Anzahl der Freiheitsgrade sowie dem Testniveau

abhängig. Da das Testniveau konstant $\alpha = 0,05$ beträgt und sich die Anzahl der Freiheitsgrade nach der Formel $r - k - 1$ berechnet, wobei r die Anzahl der Klassen und k die Anzahl der geschätzten Parameter bezeichnet, ist ein einziger Quantilswert für die Durchführung des χ^2-Tests ausreichend.

4.6.6 Sicherheitsabfragen und Benutzerfreundlichkeit

Da die statistische Datenauswertung in der Regel linear abläuft, wurde zur Implementierung der GUI ein `QWizard` verwendet. Weil z.B. der Steady-State-Graph auf Grundlage der zuvor ausgewählten Videos und der eingegebenen Schwellenwerte für die Zellaktivität ermittelt wird, müssen hier zwei getrennte Dialoge angezeigt werden (Video-Auswahl und Steady-State-Graph). Der Benutzer bekommt zu jedem Zeitpunkt genau die Informationen angezeigt, die er für den nächsten Schritt benötigt.
Bei den einzelnen Dialogen wurde durch sorgfältige Prüfung der eingegebenen Daten für eine angemessene Benutzerfreundlichkeit gesorgt. Es wird im Folgenden kurz für jeden Dialog beschrieben, inwiefern dieses umgesetzt wurde.

Username Dialog
: Es werden lediglich die Benutzernamen angezeigt, zu denen Videos existieren, die erfolgreich getrackt wurden. Erfolgreich heißt dabei, dass für mindestens eine Zelle pro Frame Positionsdaten vorliegen. Somit wird verhindert, dass unvollständig ausgeführte Tracking-Vorgänge in die statistische Auswertung aufgenommen werden.

Selection Dialog
: Der Auswahl-Dialog zeigt die Experimente nach Benutzername, Datum, Zelltyp und Stimulus sortiert an. Mehrfaches Hinzufügen eines Experiments zu einer Gruppe wird verhindert. Der Benutzer kann erst dann zum nächsten Dialog weiterschalten, wenn sich sowohl in der Kontroll- als auch in der Stimulus-Gruppe mindestens ein Experiment befindet, wobei alle ausgewählten Videos mit derselben Geschwindigkeit aufgenommen worden sein müssen. Außerdem sind die Werte für die Bestimmung der Zellaktivität obligatorisch. Sollte der Benutzer Experimente mit unterschiedlichem Zelltyp und/oder Stimulus zu einer Gruppe hinzufügen, so wird er darauf hingewiesen, kann aber – nach Wunsch – mit dieser Auswahl weiterarbeiten.

Graph Dialog
: Durch eine vertikale Linie durch die Position des Mauszeigers wird es dem Benutzer ermöglicht, eine präzise Auswahl des Steady-States zu treffen. Die Aktivitätsgraphen werden zur besseren Unterscheidbarkeit verschieden eingefärbt. Wenn der Benutzer versehentlich zuerst die rechte Grenze des Steady-States setzt und erst danach die linke Grenze, dann werden diese Markierungen intern automatisch in die richtige Reihenfolge gebracht. Darüber hinaus gibt es die Möglichkeit, nur eine oder gar keine Markierung zu setzen: Wird nur eine Markierung gesetzt, so wird diese als linke Grenze interpretiert und der Steady-State erstreckt sich bis zum Ende des Videos. Markiert der Benutzer nichts, so wird das gesamte Video als Steady-State betrachtet.

Results
: Dem Benutzer wird hier farbig signalisiert, ob die Ergebnisse der t-Tests verwendbar sind. Die jeweilige Tabellenzelle wird grün hinterlegt, wenn der χ^2-Test erfolgreich ist, bzw. rot, wenn er fehlschlägt. Sind weitergehende statistische Auswertung erwünscht, gibt CellTrack dem Benutzer die Möglichkeit, sämtliche Zellpositionen sowie die bereits berechneten Werte aus der Tabelle in dem Microsoft-Excel-kompatiblen CSV-Format abzuspeichern.

Kapitel 5

Evaluierung

In diesem Kapitel wird eine Übersicht über die erzielten Ergebnisse und Problematiken gegeben, die das Programm CellTrack aufweist. Untersucht wurde dabei der Programmablauf, die Bedienung, sowie das Laufzeitverhalten und die Qualität der Ergebnisse beim Erfassen des Bildmaterials sowie beim Tracking.

5.1 Programmablauf

Um den Versuchsablauf in einem digital vorliegenden Video zu haben, wurde für das Projekt eine kleine Videograbber Software geschrieben, welche Bilder von einem Videoeingang einer TV-Karte aufzeichnet und als PNG Bilder oder dirac-Video abspeichert. Die Daten zu dem Versuch in einer mySQL Datenbank ablegt. Das Trackingprogramm greift dann auf die mySQL Datenbank zurück und speichert dort die beim Tracking gewonnenen Daten ab. Im zweiten Programmteil, der Statistik, werden die beim Tracking gewonnenen Daten ausgewertet.

5.1.1 Videograbber

Der Grabber wurde geschrieben, um möglichst schnell Testdaten zu erhalten. Deshalb ist die GUI einfach gehalten und nur das Wichtigste wurde implementiert. Der Grabber speichert die Videodaten in der vollen PAL-Auflösung (720x576 Pixel) und nicht nur die 320x240 Pixel, die die Videokameras liefern. Ein Problem stellt die Codierung in das Wavelettformat dar, wenn das Aufnahmeintervall der Bilder zu gering ist. Bei einem Intervall von weniger als 40 Sekunden können keine Videos mit Differenzbildern erzeugt werden, da diese bis zu 30 Sekunden für die Codierung benötigen. In dieser Zeit wird auch das Vorschaubild nicht aktualisiert.
Um ein Video zu erzeugen sind nur einige Schritte notwendig. Zuerst muss der Speicherort und ein Name für das Video angegeben werden. Danach wählt man Dauer der Aufzeichnung und Intervall der Aufnahme (alle n Sekunden ein Bild). Anschließend sollten noch Angaben zu dem Versuch, wie Name des Benutzers, Zellart, zugegebene Reagenzien etc. eingetragen werden. Ein Klick auf den `Start`-Knopf startet die Aufnahme. Nach Ablauf der eingestellten Zeit wird die Aufnahme selbstständig beendet.

5.1.2 Tracking

Für das Tracking werden die Daten aus der Datenbank geholt und das Video automatisch geladen. Danach Zellen markiert und das Tracking gestartet werden. Als letzter Schritt muss das Tracking überprüft werden. Die ersten beiden Schritte sind intuitiv realisiert. Die Korrektur ist zwar relativ einfach möglich beinhaltet aber sehr viele manuelle Schritte, so das dies die meiste Zeit in Anspruch nimmt.

5.2 Testergebnisse

Als Testmaterialien standen fünf Videos aus unterschiedlichen Versuchen mit unterschiedlichen Zelltypen zur Verfügung, anhand derer die Güte des Programms ermittelt wurde.

Das Bildmaterial in den Videos besteht aus Graustufenbildern, die von einer VHS-Kamera mit einer 320x200 Pixel Auflösung stammen. Aufgrund der fehlenden Farbinformation sowie der geringen Auflösung sind die Zellen teilweise schlecht vom Hintergrund abgegrenzt. Es besteht nicht die Möglichkeit für die Biologen Zellen farblich zu markieren, um eine Erkennung zu erleichtern.

Beim Testen wurde versucht die Vorgehensweisen der Biologen beim Benutzen des Programms nachzuvollziehen. Es wird davon ausgegangen, dass die Biologen besonderen Wert auf einfache, logische Benutzbarkeit, sowie gute und schnelle Trackingergebnisse legen.

Es wurden in fünf Videos möglichst viele (zwischen 16 und 30) Zellen markiert, die sich nach den Kriterien der Biologen für das Tracking eignen. Nach dem automatischen Tracking auf diesen Zellen wurden folgende Ergebnisse beobachtet: Das Tracking dauert im Maximalfall bis zu 18 Minuten, kann aber durch Einstellungen im Programm beschleunigt werden, die allerdings evtl. sich auf die Qualität des Trackings auswirken können. Daher wurden diese Einstellungen bei den Tests nicht berücksichtigt.

Das Programm CellTrack versucht selbständig zu erkennen, ob eine Zelle noch richtig getrackt wird oder bereits verloren wurde. In diesem Fall wird das dem Benutzer angezeigt und das Tracking für diese Zelle gestoppt. Der Benutzer hat dann die Möglichkeit die Zelle manuell von dieser Stelle an weiterzutracken bzw. die Zellmarkierung zu korrigieren und das automatische Tracking erneut aufzunehmen. Allerdings kann es auch vorkommen, dass Zellen falsch weiterverfolgt werden und dies vom Programm nicht bemerkt wird. Deshalb muss der Benutzer die Ergebnisse des Programms überprüfen. In der Regel muss der Benutzer eine halbe Stunde mit einkalkulieren, in der er die verloren gegangenen Zellen manuell nachtrackt und falsch getrackte Zellen korrigiert.

Bei den Tests ergaben sich folgende Durchschnittswerte: Der Prozentsatz an erfolgreich getrackten Zellen betrug im Durchschnitt aller getesteten Videos 38%. Von den restlichen Zellen wurden im Durchschnitt 30% aller Zellen vom Programm nicht zu Ende getrackt, da sie verloren gingen und dies vom Programm erkannt wurde. 32% aller Zellen wurden vom System unerkannt falsch getrackt.

Gut erkennbare Zellen, die klar von anderen Zellen abgegrenzt sind, sich vom Hintergrund gut abheben und nicht zu klein sind, werden in der Regel erfolgreich getrackt. Schwierigkeiten bereiten vor allem Zellen, die sehr klein sind, sich überlagernde Zellen, sowie aus dem Fokus abtauchende Zellen. In diesen Fällen kann das Programm nicht immer erfolgreich erkennen, wie sich die Zelle verhält.

CellTrack ist nicht die erste Software, die sich mit der Aufgabe der Zellverfolgung beschäftigt. Allerdings scheint es die erste Software zu sein, die auf Zellbildern arbeitet, die von einer Gelatinematrix stammen. Ein direkter Vergleich mit schon existierender Software konnte nicht durchgeführt werden, da keine Software gefunden wurde, die ebenfalls auf Bildern auf Basis einer Gelatinematrix, also mit zum Teil genauso schlechtem und stark variierendem Bildmaterial, arbeitet.

5.2.1 Stabilität

Das Programm läuft in den meisten Fällen stabil. Abstürze sind nur in sehr seltenen Fällen zu verzeichnen. Bisher konnten Abstürze dann auch auf Fehler zurückgeführt werden, die dann ausgebessert wurden. Etwas häufiger kommt es noch zu Fehlverhalten in Situationen die nicht bedacht wurden. Zum Beispiel wenn das Programm intern bestimmte Werte nicht gesetzt hat. Dies ist größten teils darauf zurück zuführen, dass noch kurz vor Ende des Projektes Grundlegende-Programm-Abläufe geändert wurden, wie zum Beispiel die Möglichkeit beim Tracken nur jedes x-te Frame (x darf von 1-5 gewählt werden) zu betrachten. Dies beschleunigt zwar das Tracken, warf aber Probleme auf, da nun nicht mehr Daten für jedes Frame vorliegen. So mussten dann auch an

anderer Stelle wieder Änderungen vorgenommen werden, die potenziell auch wieder Fehler nach sich zogen.

Kapitel 6

Fazit

Dieses Kapitel geht noch einmal auf die wesentlichen Punkte der Projektarbeit ein. Neben einer kurzen Zusammenfassung erfolgt auch eine kritische Beleuchtung des Resultats. Insbesondere die Unzulänglichkeiten des Software-Systems und ihre Gründe sollen hier erörtert werden. Der Bericht schließt mit einigen Vorschlägen für die weitere Arbeit an dem Projekt CellTrack.

6.1 Zusammenfassung und Bewertung

Am Anfang der Projektarbeit stand eine Phase zur präzisen Zieldefinition, aus der sich sämtliche angestrebten Funktionen des Software-Systems ergeben haben. Diese Zieldefinition wurde in Form eines Pflichtenhefts (vgl. Anhang A) niedergeschrieben, das neben den obligatorischen Musskriterien auch optional zu erfüllende Wunschkriterien enthält. Es schloss sich eine experimentelle Phase an, in der die Möglichkeiten zum automatischen Tracken getestet wurden. Auf Grundlage der dadurch gewonnenen Informationen begann anschließend die Modellierung und die prototypische Entwicklung des Software-Systems. Inwiefern die Vorgaben des Pflichtenhefts umgesetzt wurden, soll im Folgenden untersucht werden.

In Bezug auf die Datenhaltung und die statistische Datenauswertung wurden die Zielvorgaben von der Projektgruppe erreicht. Die für den praktischen Einsatz erforderliche Funktionalität – also Einlesen von Videodaten, Durchführen statistischer Berechnungen, visuelle Ausgabe und Persistenz der Auswertungsdaten – wird von CellTrack in vollem Umfang angeboten. Auch die direkte Übernahme des Videodatenstroms von der Kamera mittels des Grabber-Tools und eine Exportmöglichkeit der Auswertungsdaten in ein von Microsoft Excel lesbares Datenformat wurden verwirklicht.
Der Aspekt der Bildverbesserung spielt in dem Endprodukt keine Rolle mehr, da Versuche gezeigt haben, dass durch Bildverbesserungsmaßnahmen keine nennenswerten Verbesserungen erzielt werden können. Auch dieses Resultat steht in Einklang mit der Zieldefinition, da ein praktischer Nutzen durch die Anwendung von Bildfiltern von Beginn an bezweifelt wurde.
Auch die grafische Benutzeroberfläche von CellTrack wurde im Wesentlichen nach den zuvor aufgestellten Gesichtspunkten gestaltet. So lag ein besonderer Schwerpunkt bei der Entwicklung auf der Benutzerfreundlichkeit und der intuitiven Bedienbarkeit des Systems. Aussagekräftige Icons und Menüeinträge sowie eine während des Programmlaufs verfügbare Hilfedatei im HTML-Format erleichtern die Benutzung von CellTrack.
Der Schwerpunkt bei der Entwicklungsarbeit lag jedoch auf dem Segmentieren und Tracken von Zellen. Obwohl die meiste Zeit in diese beiden Aspekte investiert wurde, liegen hier auch die meisten Probleme. Da sich die Schwierigkeiten des vollautomatischen Trackens bereits in der Anfangsphase zeigten, wurde von Beginn an nur ein semi-automatisches Trackingverfahren angestrebt. Doch auch mit dieser Einschränkung konnte leider kein deutlich überzeugendes Verfahren entwickelt werden.

Die Zielsetzung für das Tracking wurde im Pflichtenheft allerdings bewusst vorsichtig formuliert und somit konnten die Vorgaben auch auf diesem Sektor erreicht werden.
Über die Gründe, die zu diesem Resultat führten, soll nun kurz reflektiert werden.

Unter Berücksichtigung der Tatsache, dass keiner der Projektgruppen-Teilnehmer zu Beginn der Arbeit Erfahrungen auf dem Gebiet der Bildverarbeitung hatte, fiel die experimentelle Phase länger aus als erwartet. Viele verschiedene Ansätze mussten zunächst auf ihre Verwendbarkeit hin untersucht werden, letztendlich kristallisierten sich jedoch nur das Blockvergleichverfahren, der Snake-Ansatz und das Level-Set-Verfahren als nutzbar heraus. Andere Ansätze, insbesondere das Split-And-Merge-Verfahren zur Segmentierung, mussten verworfen werden, obwohl schon viel Zeit darin investiert worden war.
Die brauchbaren Verfahren mussten nun geeignet parametrisiert werden, um auf den vorliegenden Daten arbeiten zu können. Lange Zeit lagen für Testzwecke allerdings nur wenige und qualitativ minderwertige Videodaten vor. Mit einer geeigneten Optimierung der verwendeten Verfahren konnte somit erst relativ spät begonnen werden.
Weitere Faktoren, wie z.B. der erfolglose Versuch, Videodaten im MNG-Format zu speichern und einzulesen, oder auch die von den Projektgruppen-Betreuern relativ offen formulierte Herangehensweise, führten ebenfalls dazu, dass sich ein ungünstiges Zeitmanagement innerhalb der Projektgruppe ergab.
Schließlich sollte auch berücksichtigt werden, dass der aktuelle Stand der Forschung offensichtlich noch kein optimales Resultat zulässt. Viele Forschungsansätze zum Thema Tracking sind auf das spezielle Gebiet des Zell-Trackings nicht anwendbar oder müssen geeignet konfiguriert und kombiniert werden, um halbwegs verwertbare Ergebnisse zu erzielen. In die Kombination der drei o.g. Verfahren wurde ebenfalls viel Zeit investiert, in der Hoffnung, dadurch deutliche Verbesserungen erzielen zu können. Leider hat sich aber auch diese Hoffnung nicht erfüllt.

Trotz der (abhängig von der betrachteten Zellkultur) relativ geringen Tracking-Erfolgsquote kann CellTrack jedoch als erfolgreiches Projekt bezeichnet werden. Aus der Perspektive der Biologen stellt CellTrack eine eindeutige Verbesserung zum Ist-Zustand dar: Müssen Zellvideos bislang komplett manuell getrackt werden, so bietet CellTrack zumindest eine semi-automatische Tracking-Variante an. Während der automatischen Phase erwartet das Software-System keinerlei Interaktion von Seiten des Benutzers, so dass nach dem Laden eines Videos und dem Markieren der zu beobachtenden Zellen zunächst keine Arbeitszeit in Anspruch genommen wird. Erst nach Abschluss des Tracking-Vorgangs ist eine manuelle Korrektur des Ergebnisses nötig, die jedoch weniger zeitintensiv ausfällt als ein vollständig manuelles Tracking. Des Weiteren wird dem Benutzer die statistische Auswertung der Messdaten komplett abgenommen, was ebenfalls zu einem Zeitgewinn führt.
Nicht zu vernachlässigen ist neben der reinen Zeitersparnis auch die Erleichterung der Videodatenarchivierung. Bislang wird das Bildmaterial auf handelsüblichen VHS-Kassetten archiviert, was einen enormen Platzbedarf sowie Qualitätsverlust bei längerer Lagerungszeit mit sich führt. CellTrack ermöglicht nun die zeitgemäßere digitale Erfassung und Speicherung von Videodaten.

6.2 Ausblick

Auch wenn die Arbeit der Projektgruppe beendet ist, besteht die Möglichkeit, CellTrack frei weiterzuentwickeln, da das Software-System unter der GNU Public License (GPL) veröffentlicht wird. Sicherlich gibt es sehr viele Ansätze für Verbesserungen und Weiterentwicklungen, daher sollen die aus Sicht der Entwickler wichtigsten Punkte kurz aufgelistet werden:

- Beim Level-Set-Verfahren gibt es zwei Ansätze für die Verbesserung. Da es dem Verfahren vor allem an Recheneffizienz fehlt, zielen beide Ansätze auf dieses Problem ab. Zum einen könnte die für das Level-Set verwendete Datenstruktur auf Narrow-Band-Matrizen umgestellt werden und zum anderen könnten die Berechnungen durch einen Grafikprozessor gemacht

werden. Die erste Methode wurde ansatzweise auch in CellTrack implementiert. Da jedoch die Fertigstellung des Verfahrens im Vordergrund stand, wurden die Bemühungen an diesem Ansatz recht schnell eingestellt. Der zweite Ansatz ist sinnvoll, da es beim Level-Set vor allem um die schnelle Berechnung von Matrizenoperationen geht und diese von heutigen Grafikkartenchips sehr effizient durchgeführt werden können.

- Neben der Verbesserung des Level-Set könnte aber auch die Entwicklung weiterer Tracking-Verfahren sinnvoll sein. Diese könnten bei qualitativ besseren Videos den Hintergrund als Referenzbild benutzen und so das Tracking der Zellen deutlich vereinfachen, da sich der Hintergrund im Laufe eines Videos nur minimal ändert.

- In CellTrack ist nur ein Tracking möglich, nachdem das Video bereits aufgezeichnet wurde. Es wäre aber auch denkbar schon während des Grabbings die Zellen zu verfolgen, also ein Tracking „on-the-fly“. Hierdurch ließe sich erhebliche Rechenzeit sparen, da nur alle paar Sekunden ein Bild aufgenommen wird und in der Zwischenzeit die Rechenleistung für das Tracking genutzt werden könnte. Allerdings dürfte das Tracking bei einem solchen Ansatz nicht versagen, da der Benutzer das Programm nicht die ganze Zeit kontrollieren könnte. Der gesamte Trackingvorgang würde dann so lange dauern, wie ein gesamter Versuch.

- Um das Tracking stabiler zu machen, so dass es nicht mehr so oft versagt, wäre es unter anderem möglich, nachdem ein Mittelpunkt für eine Zelle bestimmt wurde, den Bereich um diesen Mittelpunkt nochmals zu segmentieren. Dadurch könnte überprüft werden, ob sich der Mittelpunkt wirklich über einer Zelle befindet, oder ob gegebenenfalls eine Korrektur erfolgen muss.

- Eine weitere Verbesserung des Trackings könnte darin bestehen, anhand der Form, Größe und Versuchsbeschreibung zu überprüfen, welche Art von Zellen getrackt wird. So könnten Probleme, die beispielsweise das Snake-Verfahren mit kleinen Zellen hat, umgangen und unnötige Aufrufe von Verfahren, bei denen von vorne herein klar ist, dass sie versagen, vermieden werden.

- Auch eine integrierte Bewertung der Tracking-Verfahren wäre sinnvoll. So könnte CellTrack selbst erkennen, ob eine Zelle korrekt getrackt wurde.

- Denkbar ist auch ein Multi-Agenten-System, das jeder Zelle einen Agenten zuweist. Diese Agenten könnten sich dann gegenseitig austauschen, was eine Behandlung von Zellhaufen vereinfachen würde.

- Verbesserungen in der Statistik können sich z.B. an den Diagrammen orientieren. So könnte ein Spider-Diagramm gezeichnet werden, aus dem ersichtlich wird, wie sich jede einzelne Zelle vom Startpunkt aus bewegt hat.

- Die Statistik kann auch in Bezug auf die Signifikanztests erweitert werden. Nichtparametrische Tests sowie einstellbare Testniveaus sind hier denkbar.

6.3 Danksagung

Die Projektgruppe 464 dankt allen Menschen, die zum Fortschritt von CellTrack beigetragen haben – sei es direkt durch persönliche Unterstützung oder indirekt durch Hilfestellungen aller Art. Expliziter Dank geht an Dipl.-Inform. Martin Wawro und Dipl.-Inform. Frank Weichert vom Lehrstuhl für Graphische Systeme (Fachbereich Informatik der Universität Dortmund) für die gute Betreuung und an Priv.-Doz. Dr. Frank Entschladen und sein Team vom Institut für Immunologie der Universität Witten/Herdecke für die reibungslose Kooperation.

Anhang A

Pflichtenheft

In diesem Dokument findet der Auftraggeber eine Zusammenfassung aller fachlichen Anforderungen, die das zu entwickelnde Software-Produkt erfüllen muss. Es werden alle angestrebten Zielkriterien aufgeführt. Ebenso erfolgt eine Auflistung der Funktionen und Qualitätsanforderungen. Die Beschreibung findet in verbaler Form statt, da diese Form übersichtlicher ist als das IEEE SRS-Schema, welches sonst häufig in Pflichtenheften benutzt wird [T+94].

A.1 Zielbestimmung

Die Zielbestimmungen lassen sich in vier verschiedene Bereiche unterteilen, dabei werden jeweils die Musskriterien, die Wunschkriterien und die Abgrenzungskriterien einzeln ausgearbeitet. Die vier Bereiche sind

- Daten und Statistik
- Bildverbesserung
- Segmentierung und Tracking
- GUI

Es folgt eine detaillierte Auflistung aller Kriterien.

A.1.1 Datenhaltung und Statistik

- Musskriterien
  - Videodaten im PNG-Format können eingelesen werden.
  - Film- und Auswertungsdaten können gespeichert werden, dazu wird eine Datenbank benutzt.
  - Zusatzinformationen (Kommentare etc.) können gespeichert werden.
  - Tracking-Kurven werden graphisch ausgegeben.
  - Messdaten werden tabellarisch ausgegeben.
  - Migrationsparameter werden berechnet und tabellarisch ausgegeben.
- Wunschkriterien
  - Eine direkte Aufzeichnung von der analogen Videoquelle ist möglich, die Pinnacle PCTV Stereo-Karte wird unterstützt.

  - Die Migrationsparameter werden grafisch aufbereitet und ausgegeben.
  - Tabellen, Diagramme etc. können ausgedruckt werden.
  - Versuchsdaten werden automatisch archiviert.
  - Datenexport nach MS Excel ist möglich.
  - Die Parameter für den Signifikanztest (Art des Tests, Testniveau etc.) bzw. für die Auswertung allgemein können modifiziert werden.

- Abgrenzungskriterien
  - Andere als die angegebene Hardware wird nicht unterstützt (digitale Camcorder etc.).
  - Andere Videoformate werden nicht unterstützt.
  - Es werden neben dem Signifikanztest keine weiterführenden statistischen Methoden implementiert.
  - Nur fest programmierte Parameter werden ausgegeben / ausgewertet (z.B. Migrationsaktivität), es gibt in dieser Hinsicht keine Erweiterungsmöglichkeiten.
  - Datenexport ist nur nach MS Excel möglich.
  - Datenspeicherung ist nur auf der Festplatte möglich, es gibt keine Optionen wie z.B. Daten auf DVD brennen.

A.1.2 Bildverbesserung

- Musskriterien:
  - Es gibt in diesem Bereich keine Musskriterien, da die Bilder unter Umständen nicht verbessert werden müssen. Bildverbesserungselemente werden daher nur bei Bedarf eingesetzt. Zu erwarten ist, dass Deinterlacing und eine Shading-Korrektur zum Einsatz kommen können, wenn die Bilder verbessert werden müssen. Daher werden diese auch als Wunschkriterien aufgenommen.

- Wunschkriterien
  - Deinterlacing
  - Shading-Korrektur

- Abgrenzungskriterien
  - (keine)

A.1.3 Segmentierung und Tracking

- Musskriterien
  - Eine Segmentierung der Zellen durch den Benutzer muss möglich sein.
  - Das Tracking einer Zelle erfolgt automatisch, wenn es sich um eine Zelle auf einem einfachen Weg handelt. Ein einfacher Weg liegt vor, falls die Zelle die ganze Zeit über in der Fokusebene bleibt und sich ihre Bewegung nicht mit anderen Zellen überschneidet.
  - Das Tracking wird automatisch bis zu kritischen Stellen durchgeführt, danach muss eine Benutzereingabe erfolgen.
  - Das Tracking einer jeden Zelle wird automatisch dokumentiert, dabei wird der zurückgelegte Weg der Zellen gespeichert. Sollten Zellen absterben oder sich aus der Fokusebene entfernen, wird dieses ebenfalls dokumentiert.

- Wunschkriterien
  - Alle Zellen werden automatisch segmentiert, dabei wird automatisch erkannt, ob es sich um eine gut oder schlecht zu sehende Zelle handelt und der Benutzer wird gegebenenfalls aufgefordert einzugreifen.
  - Alle Zellen werden automatisch getrackt.
  - Im Startframe werden zu trackende Zellen automatisch vorgeschlagen. Es erfordert jedoch eine Bestätigung durch den Benutzer ob diese auch tatsächlich getrackt werden sollen.
- Abgrenzungskriterien
  - (keine)

A.1.4 GUI

- Musskriterien
  - Alle Funktionen müssen in der GUI zugänglich gemacht werden.
  - Zu jeder Funktion wird eine Hilfe angeboten.
  - Allgemeine Richtlinien zur GUI-Entwicklung werden eingehalten [Sut02].
  - Fehlermeldungen werden allgemein verständlich formuliert.
  - Es besteht die Möglichkeit, den Tracking-Vorgang zu stoppen und neu zu starten.
  - Der Arbeitsfortschritt / Status wird angezeigt.
- Wunschkriterien
  - Einstellbare Parameter werden visuell und inituitiv verständlich dargestellt.
  - Bei Bedarf ist eine Zoomfunktion auswählbar.
  - Das laufende Verfahren kann manipuliert werden.
  - Rücksprünge zu einzelnen Schritten sind möglich.
  - Einstellungen können manipuliert und abgespeichert werden.
- Abgrenzungskriterien
  - Individuelle Benutzereinstellungen wird es nicht geben, d.h. die GUI wird nicht veränderbar sein.

A.2 Produkt-Einsatz

A.2.1 Anwendungsbereich

Das Software-System soll im Bereich der Zellforschung eingesetzt werden. Dort wird es zur Untersuchung der Bewegung von Zellen benutzt. Ein Einsatz der Software außerhalb dieses Bereichs ist nicht vorgesehen.

A.2.2 Zielgruppe

- Personengruppen, für die das Produkt vorgesehen ist:
  - Biologen,
  - medizinisches Personal.
- Personengruppen, für die das Produkt nicht vorgesehen ist:
  - Informatiker,
  - Personen ohne medizinisches oder biologisches Hintergrundwissen.

A.2.3 Betriebsbedingungen

- physikalische Umgebung des Systems
  - Das System wird in einem medizinischen Labor eingesetzt, wo konstante Temperaturverhältnisse herrschen. Die Lichtintensität im Labor ist konstant, jedoch werden die Zellen von einer Wärmelampe auf einer Temperatur von 37°C gehalten. Diese Wärmelampe schaltet sich von Zeit zu Zeit ein bzw. aus. Das Labor befindet sich in der Nähe einer viel befahrenen Straße, daher ist mit gelegentlichen Erschütterungen durch das Verkehrsaufkommen zu rechnen. Im Labor herrschen sterile Bedingungen, so dass keine Verschmutzungen auf der Kameralinse oder auf dem Versuchsaufbau entstehen können.
- Betriebszeit
  - Das System wird ganztägig benutzt.
- Aufsicht des Systems
  - Das System wird nicht permanent beaufsichtigt, allerdings besteht die Notwendigkeit, mehrmals während des Programmablaufs den Fortschritt zu überprüfen. Außerdem steht vor Ort kein Wartungspersonal für das System zur Verfügung.

A.3 Produkt-Umgebung

- Hardware: Bei dem Zielrechner handelt es sich um einen PC mit x86-Architektur, einer handelsüblichen Grafikkarte und einer Festplatte mit einer Mindestkapazität von 80GB. Soll der Rechner auch zum Grabben eingesetzt werden, ist eine vom Linux-Kernel unterstützte TV-Karte erforderlich.
- Software: Die Zielmaschine ist ein Linux-System mit einer MySQL-Datenbankanbindung.

A.4 Produkt-Funktionen

/F10/

- Beschreibung: Die Zellbewegungen können nicht direkt abgefilmt werden, daher wird die Beobachtung durch ein Mikroskop erfolgen. Das Bild, welches durch das Mikroskop zu sehen ist, wird abgefilmt. Um mit dem Bildmaterial arbeiten zu können, wird es im PNG-Format eingelesen.
- Funktion: Einlesen der Videodaten im PNG-Format.

/F20/

- Beschreibung: Um Versuchsergebnisse nachvollziehen zu können und um Vergleichsdaten zu haben, ist die Archivierung der Bilddaten notwendig. Die Videodaten werden in einer Datenbank abgelegt. Zusätzlich wird dem Benutzer die Möglichkeit gegeben, weitere Informationen in dieser Datenbank abzulegen.
- Funktion: Abspeichern der PNG-Videodaten sowie zusätzlicher Informationen in einer MySQL-Datenbank.

/F30/

- Beschreibung: Die Zellbewegungen stehen im Mittelpunkt des Interesses der Anwender. Die Bewegungen der ausgewählten Zellen müssen daher nach dem Programmdurchlauf ausgegeben werden.
- Funktion: Tracking-Kurven ausgeben.

/F40/

- Beschreibung: Unabhängig von der Tracking-Kurve sind noch weitere Messdaten sowie Migrationsparameter gegeben, welche tabellarisch ausgegeben werden.
- Funktion: Messdaten und Migrationsparameter tabellarisch ausgeben.

/F50/

- Beschreibung: Die Zellen sind in der Collagen-Matrix schwer zu entdecken. Sollte eine Zelle falsch oder unvollständig als Startzelle erkannt werden, so ist die Auswertung des Weges dieser Zelle nicht möglich. Daher muss der Benutzer im ersten Frame eine oder mehrere Zellen markieren.
- Funktion: Möglichkeit, die Zellen vor dem Tracking zu selektieren bzw. zu markieren.

/F60/

- Beschreibung: Die Zellbewegungen sind Gegenstand der Untersuchung der Biologen. Es gilt „gut sichtbare“ Zellen und Zellen auf einfachen Wegen automatisch zu tracken. Gut sichtbare Zellen sind Zellen, die das menschliche Auge sofort ohne Mühe als Zellen erkennen kann. Diese Zellen heben sich eindeutig vom Hintergrund ab und befinden sich in der Fokalebene
- Funktion: Bewegungen von zuvor markierten Zellen werden automatisch verfolgt, sofern es sich um Zellen auf einfachen Wegen handelt.

/F70/

- Beschreibung: Während eines Durchlaufs kann es vorkommen, dass dem Benutzer ein Fehler auffällt oder er den Vorgang aus anderen Gründen stoppen möchte. Diese Möglichkeit muss das Programm bieten. Insbesondere beim Tracking kann es vorkommen, dass nur Zellen markiert wurden, die ungünstig für den Tracking-Algorithmus sind. Auch hier ist die Abbruch-Funktion sinnvoll.
- Funktion: Stoppen des Programmdurchlaufs.

/F80/

- Beschreibung: Das Videomaterial wird vor dem Versuch nicht durchgeschaut. So kann es vorkommen, dass bestimmte Parameter falsch eingestellt wurden. Möchte der Versuchsleiter während der Ausführung bestimmte Parameter ändern, ohne die bisherigen Trackingergebnisse zu verwerfen, so muss es die Funktion des Pausierens geben.
- Funktion: Unterbrechen und Fortsetzen des Versuchs.

A.5 Wunschfunktionen

/WF10/

- Beschreibung: Wie in Funktion /F10/ beschrieben, wird das Videosignal im PNG-Format eingelesen. Wünschenswert wäre ein direktes Einlesen von der Videodatenquelle.
- Wunschfunktion: Videodaten werden direkt eingelesen.

/WF20/

- Beschreibung: In kritischen Fällen wird das Tracking pausiert und der Benutzer muss bestimmte Eingaben vornehmen. Um eine Entscheidung treffen zu können, werden Folgezustände berechnet und der Benutzer muss dann aus den Alternativen entscheiden, wie fortgefahren werden soll.
- Wunschfunktion: Folgezustände anzeigen, um eine Entscheidung zu ermöglichen.

/WF30/

- Beschreibung: Die tabellarische Darstellung der Migrationsparameter ist ausreichend, eine visuelle Darstellung ist jedoch aussagekräftiger und somit erstrebenswert.
- Wunschfunktion: Grafische Darstellung der Migrationsparameter.

/WF40/

- Beschreibung: Sollte der Benutzer eine manuelle Korrektur des Trackings vornehmen müssen, so ist dieses anhand des aktuellen Bildes kaum möglich. Dem Benutzer müssen mehrere Bilder zu Verfügung stehen. Denkbar ist auch, dass ein Zoom auf die fragliche Stelle das Problem vereinfacht.
- Wunschfunktion: Übersicht über die letzten Frames anzeigen; bei Bedarf eine Zoommöglichkeit anbieten.

/WF50/

- Beschreibung: Der Benutzer hat die Möglichkeit Zellen zu selektieren, allerdings können einige Zellen auch vom Rechner selektiert werden. Diese automatisch selektierten Zellen werden dem Benutzer angezeigt und falls die Wahl zu seiner Zufriedenheit ausgefallen ist, muss er die Auswahl bestätigen.
- Wunschfunktion: Automatische Selektion der Zellen.

/WF60/

- Beschreibung: Der Benutzer wird unter bestimmten Umständen aufgefordert, einzugreifen. Sollte dem Benutzer auffallen, dass einige Parameter in früheren Arbeitsschritten geändert werden müssen, so sollte er die Möglichkeit haben, in vorherige Schritte zu springen.
- Wunschfunktion: Nach Korrekturaufforderung gibt es die Möglichkeit, zu zurückliegenden Arbeitsschritte zu springen.

/WF70/

- Beschreibung: Im Bereich Segmentierung und im Bildverbesserungsbereich sind Parameter voreingestellt. Möglicherweise existieren bessere Startwerte für dieses Parameter als die vorgegebenen. Die neuen Parameter sollten also als neue Standardwerte gespeichert werden können.
- Wunschfunktion: Speicherung der eingestellten Parameter.

A.6 Produkt-Daten

A.6.1 Bilddaten

Das Programm muss in der Lage sein, die Videodaten und dazugehörige Informationen zu speichern. Dies ist eine der Hauptfunktionen des Programms. Die Videodaten liegen im PNG-Format vor.

A.6.2 Versuchsdaten

Gespeichert werden relevante Versuchsdaten wie Benutzername, Datum, Zelltyp und Stimulus, außerdem das Video selbst, die durch das Tracking ermittelten Zellmittelpunkte und statistische Auswertungsdaten.

A.7 Benutzeroberfläche

Auf dem Großteil des Bildschirmes wird das zu trackende Video dargestellt. Das Bildschirmlayout wird in allen Arbeitsschritten ähnlich aussehen, um schnelle Einarbeitung und einen hohen Wiedererkennungswert zu garantieren. Über ein Hauptmenü können alle Funktionen angewählt werden. Diese sind selbsterklärend und zusätzlich auch über Buttons ansprechbar.

A.8 Globale Testszenarien bzw. Testfälle

Bereits abgefilmte Videodaten werden zum Testen verwendet. Dabei werden möglichst verschiedene Videodaten herangezogen.

A.9 Entwicklungsumgebung

A.9.1 Software

Für die UML-Modellierung wird Together von Borland verwendet. Die Softwareentwicklung findet unter Linux mit dem KDE-Tool KDevelop statt. Als Programmiersprache wird C++ und die Qt-Klassenbibliothek von Trolltech verwendet.

A.9.2 Hardware

Die Entwicklungshardware stimmt im Wesentlichen mit der in Kapitel A.3 genannten Zielhardware überein.

Anhang B

Manual

Contents

B.1 Getting started

B.1.1 Introduction

Some scientists explore the behavior of cells under special circumstances, for example under the influence of noradrenaline. To analyse the motion, direction and speed of these cells, they have to be tracked.

The project group CellTrack developed a tool to assist the Institute of Immunology at the University of Witten/Herdecke in analysing the migration, invasion, and extravasation of leukocytes, lymphocytes and tumor cells within a 3D collagen matrix. CellTrack is intended to let users track cells contained in a video either automatically or manually. If the results do not satisfy the requirements, the user can change them manually. Is the video of low quality (for example too dark or too noisy) for CellTrack to track cells accurately, the user can also track all cells of the video manually.

The paths of one or all tracked cells can be displayed to illustrate the motion of the cells. Finally, the whole video with all paths can be verified in order to review the results of the tracking process. An integrated statistics tool enables a simple access to statistical information.

CellTrack is based upon a database, in which all gathered information about the different experiments are automatically saved. There is a lot of data specifying the video (for example: Who is the experiments conductor? What agents are used in the collagen matrix? ...). The results of the tracking are saved in the database. The user can record the initial cell markings to verify the automated cell tracking progress, or simply redo it.

B.1.2 System requirements

- Pentium IV 1000 Mhz or above. AMD Athlon XP 1400+ or above.
- 32 megabytes or above. More memory is recommended, but not required.
- Linux (right now CellTrack is compilable on UNIX).
- Access to a SQL database via network.
- The total disc space should be at least 80 gigabytes, allowing to save a lot of videos.
- A client requires about 100 megabytes of free disc space.

B.1.3 Installing the program

CellTrack is delivered in form of a binary .tar.gz package, meaning no install procedure is required. The user can simply extract the release files into a separate folder by typing `tar -xvzf` *`name`*`.tar.gz` and executing the program from command line.

B.2 How to use the main program

B.2.1 User interface - a brief introduction

Figure B.1 shows the frontend as seen by the user directly after starting the program, creating a database connection and opening a video file. The numbers marked in red are explained below.

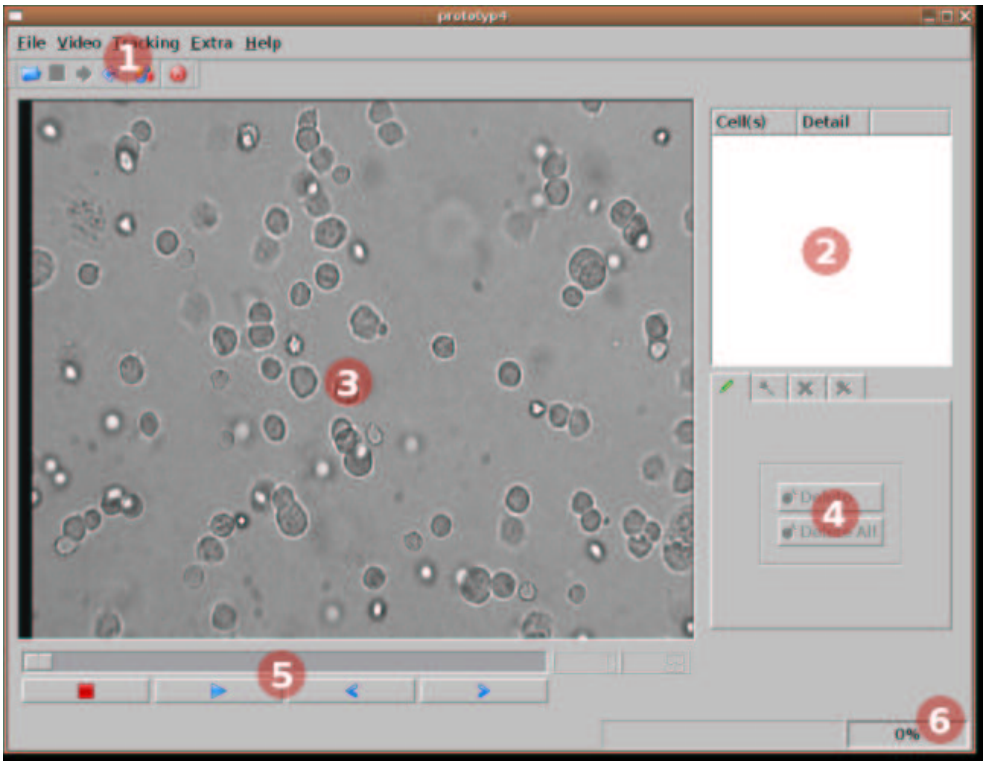


Figure B.1: Userinterface of CellTrack

1. Menu Bar
   The menu bar provides control over videos, file export and the tracking process. It also offers access to the database.

2. Cell List
   Cells that have been selected will be listed here. Expanding an entry leads to a more complete information about the cell. A cell adopts several states, represented by different colors of the LEDs. Check marks can be set or removed indicating the tracking status.

3. Screen
   The interaction with cells takes place on the screen most of the time. The user has to mark the cells to be tracked. Once those cells are selected, the positions can still be edited manually.

4. Control Field
   This field provides control over the most important features during the process of tracking. Four views are available:

Available Modes	
Editing Mode	The Editing mode lets the user select, move, resize and delete cells.
Tracking Mode	In this mode are two submodes encapsulated. First is a mode supporting automatic tracking of cells. Second is a mode allowing the user to track a cell manually by pointing on it with the mouse.
Error Correction Mode	If tracking failed, the user has the option to use some functionality of this mode to correct the tracked path. One example is to dismiss a path, which will remove the tracked path from the actual frame forward.
Option Mode	Some settings can be changed here by the user.

5. Video Control
   This button group controlls the playback of the video, similar to most common video players.

6. Status Bar
   At the bottom of the frontend there is a status bar. Pointing on buttons with the mouse will show a tooltip. There is also a percentage bar. This bar will show the percentage of all succesfully tracked cells.

B.2.2 Quick guide to tracking

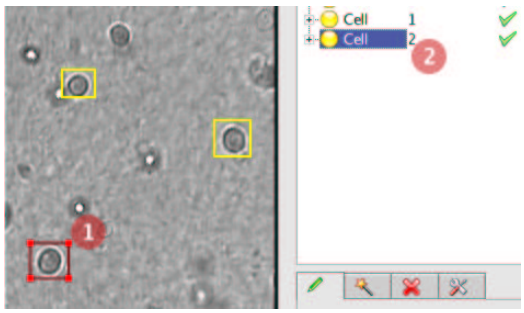


Figure B.2: Screen, cell list and several marked cells

To track a video the user has to open a videofile, mark cells and press [Start] in the tracking menu. However, there are no guarantees that this will result in perfect tracking results. Achieving better results might require manual corrections.
Figure B.2 shows the cell list to the right (red "1") and some marked cells (red "2"). The cell list displays all required information about the marked cells. Every marked cell is listed here. For more detailed information see paragraph B.2.3. On the screen, cells can be marked and will automatically appear in the cell list. More reading about the screen is available in paragraph B.2.3.

Marking cells and editing markings

In CellTrack marking of cells is done by dragging a rectangle around the cell. This is done by clicking on the upper left corner of a cell, holding the mouse button while moving to the lower right corner of the cell and releasing it.
Once the mouse button is released the cell will be added to the list. The spacing of the surrounding rectangle can still be edited. To do so, the user first selects a cell (see figure B.2 red "1") which will be highlighted. The corners of the spanning rectangle will appear enlarged. Dragging them to a different location will change the shape of the rectangle. Click inside the rectangle to drag the whole marker to another place.

Automatic Tracking

There are a few steps to complete to get automatic tracking to work:

- Mark cells
  This can be accomplished by switching to the Editing Mode and dragging proper selections around a cell.
- Press [Start]
  Switch to the automatic Tracking Mode and press the [Start] button. The video screen will update only every fifth frame to save CPU time. Depending on your computers speed the tracking will be faster.
- Wait for tracking to finish
  The tracking is finished when the last frame of the video is shown or when the [Stop] button is pressed.

Manual Tracking

In case of insufficient automatic tracking results the user can track manually via mouse. Figure B.3 shows a situation, where tracking of the last cell was just completed by hand. Hence all cells are marked green.

- Mark cells
  This can be accomplished the same way as in automatic tracking. The user needs to switch to the Editing Mode and select cells (figure B.3 red marking number "2").
- Track by Hand
  The next step requires to switch to the Tracking Mode and select manual tracking (figure B.3 red marking number "1"). Now a cell has to be selected for tracking by clicking on it in the cell list (figure B.3 red marking number "3"). Tracking starts by clicking on the cells position - the video will begin to play.
- Finish Tracking
  Tracking is finished when the last frame of the video is shown or when the mouse button is pressed again.

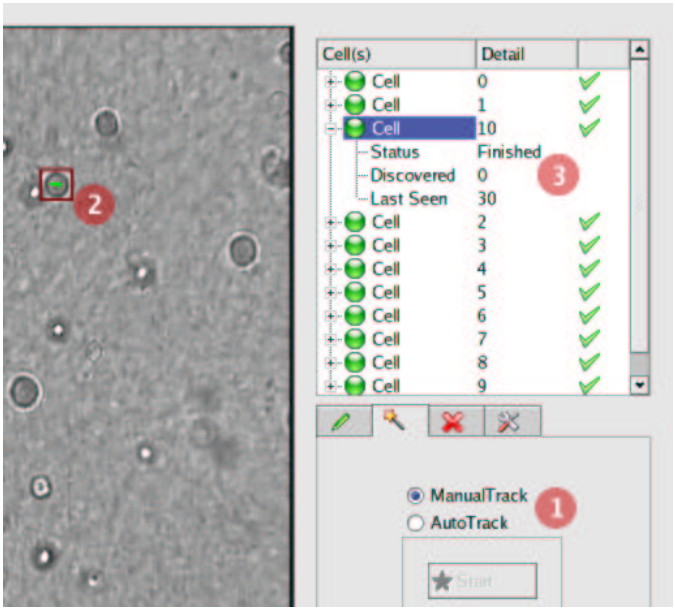


Figure B.3: Completed tracking manually

Correcting wrong paths

Paths can be wrong because of various factors. For example some cells may have collided and a wrong cell was followed from that point of time. Another example is, that cells sometimes vanish from the focus. Once tracking errors have been discovered the user should correct them. There is a simple way to do so:

- Finding the critical point in time
  The user first has to find the frame where the tracking failed. He/She can do so by dragging the slider contained in the player elements, or in case of lost cells by double clicking on the cell in the cell list.

- Dragging and resizing
  First the user selects the Edit Mode. Once the correct frame has been located the user can select the cells rectangle and drag it to its proper location, the ongoing path will be removed and this cell is ready to be tracked again.
  Alternatively, the user can select the `Dismiss` button to remove the path from this frame forward.

- Go on
  The user now has two options while returning to the Tracking Mode. The first one is to press the `Continue` button and track onward automatically. The second one is to track the cell manually from now on.

B.2.3 User Interface Components

Since the user interface is composed of many elements accomplishing their own operations, they will all be explained separate.

Menu Bar

As seen in figure B.1 the menu bar is the element marked with a red "1".

- `File`
  With this menu the user can open a new video file. He/She can export and import the initial rectangles of cells and also quit the program.

- `Video`
  This menu gives the user control over the video. It plays and stops the video. It allows to jump back to the first frame and to step forward or backward one frame relative to the actual frame.

- `Tracking`
  This menu gives control over the most important tracking features. On top of the menu the user can select a cell and click `Delete` to remove it; so it is not marked anymore and will not be tracked. Pressing `Delete All` removes every cell in the list.
  `Start` will start the automatic tracking. It always discards all previously collected tracking paths and retracks the whole video. In case the user does not want to restart, he/she can select `Continue`. The process of tracking can be interrupted by pressing `Stop`.
  If a cell is completed and should not be considered anymore it can be unlocked (the green check mark in the cell list will disappear) and locked by pressing `(Un-)Lock` consecutively.
  `Dismiss` will remove the retrieved path from the actual image and the highlighted cell in all following frames. The user can now retrack automatically or correct the path manually.

- `Extra`
  In this menu the user has different options.
  `Statistics` opens the statistics tool for analysing tracked experiments.
  `Show Videos in Database` will present a list of all untracked videos and their current location.
  `Change Video File Location` is a simple tool to change the location and the media number of a video in the database. It is useful when moving the files from one computer to another or to DVD.
  `Export Data to CSV` allows to write the cell positions of a tracked video to a Microsoft Excel®-compatible file. The current video is preselected on a list of all tracked videos, if it has already been tracked.
  `Commit to Database` is only available after completly tracking an opened video. This will store all the tracking information in the database and make the video eligible for statistical analysis.

- `Help`
  This menu entry offers a look at the credits of our project members and also brief instructions on how to track.

Cell List

The cell list is marked with a red "2" in figure B.1. If the user has opened the video, cells can be marked by dragging a rectangle around it. Once a cell has been marked it will appear in the cell list. A cell list entry consists of an LED, a name, a number and a check mark. If the user expands a cell entry in the list, more options will be shown.

- Meaning of an LEDs color
  There are three colors: green, yellow and red. Once a cell has been added it will contain a yellow LED meaning this cell is ready to be tracked, it has not been lost and is not completed yet. If this cell is tracked throughout the whole video without getting lost it will turn green in the end, meaning this cell is now completed. The user should now check the path for correctness. If the tracking procedure starts and the cell gets lost, the LED will turn red meaning this cell has not been tracked correctly and its position must be corrected or the cell must be deleted.

Note that red and green marked cells will not be tracked.
If cell positions of red or green marked cells are corrected their color will change back to yellow.

- Number of a cell
  Cells have a unique number, while a certain video is loaded.
- Green check mark
  The green check mark is just a help for the user. He can mark as many cells as required and remove the check marks (by clicking on the check mark or the `(Un-)Lock` button) at those cells which should not be tracked. Cells not having a check mark will be ignored by the automatic tracking. Tracking cannot be completed in this case.
- Hidden options
  Clicking on the expand icon will show the options which are normally hidden. "Last seen" tells the user in which frame the cell was last found by the tracking algorithm. Also the cells status is given in textual form.

Screen

The screen is the main component of the program since it shows most information. There are several modes the screen can change to and which should be explained step by step. The screen is marked with a red "3" in figure B.1 and is tightly coupled to the control fields modes. Three modes are available: editing, viewing and manual tracking.

Editing Screen

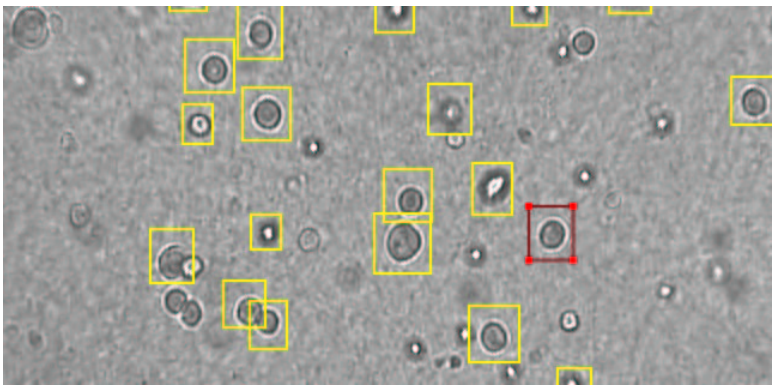


Figure B.4: Example on how to mark cells

Directly after starting the program and loading a video, the user faces the editing screen. Figure B.4 shows a possible screen with marked cells. This screen lets the user perform the following actions:

- Marking a cell
  Marking a cell can be performed by clicking somewhere (near) outside the cell and dragging the rectangle, so that the cell fits inside.
- Deleting a cell
  Directly after marking a cell or after clicking at a cells rectangle it is highlighted (only one cell at a time can be highlighted). Pressing the `Delete` button will cause the cell to disappear, meaning this marking does not exist anymore.

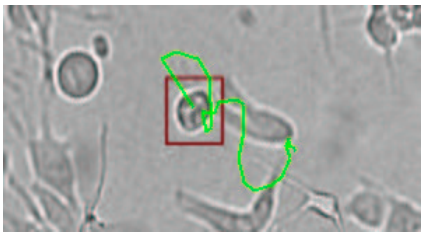


Figure B.5: Example on how to track manually

- Dragging and resizing
  A highlighted cell is marked specially. It contains four enlarged edges, which can be used to resize the marking. Clicking inside the large rectangle will enable the user to drag the marking to a new location.

Tracking Screen

This screen lets the user track the cell with the mouse cursor. Selecting manual tracking will automatically switch to this screen. This might be the preferred option if the video is very noisy and difficult to track automatically. Figure B.5 shows an example how a manual tracked path might look like. If the user wants to track with the mouse, several steps have to be performed:

1. Mark cells and switch to manual tracking (see section B.2.3 for details).
2. Highlight a cell in the cell list (see section B.2.3 for details).
3. Once a cell is highlighted, the program knows which cell will now be tracked (of course only one cell at a time is trackable by the user). Now click inside the rectangle that marks the appropriate cell – the video will start and the mouse pointer will change.
4. The cell should now be followed by the mouse cursor. The path is recorded and displayed.
5. Click again to stop the video and also manual tracking. The path has been saved and can be reviewed. Continue by clicking on the cell again and repeat the steps from there on.

Viewing Screen

Very limited interaction is allowed on the viewing screen. Users can only mark (highlight) and unmark a cell by clicking inside or outside a marking. This screens purpose is to show the already gathered information, for example a cell's path. One possible use is after the user has selected to track cells automatically.

Control Field

This is the field where the user can gain control over the tracking processes. It is marked with a red "4" in figure B.1.

Editing Mode

The Editing Mode (see figure B.6) is represented by a green marker. The program will be set to Editing Mode at the start by default. All other modes are disabled, because they do not work without having marked cells. There are only two buttons; one to delete a highlighted cell and the other to delete all cells.

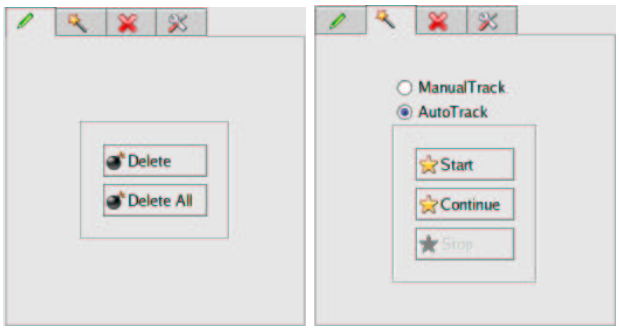


Figure B.6: Edit interface and tracking interface

Tracking Mode

Once the user has marked cells, he/she can switch to other modes, for example the Tracking Mode (see figure B.6). In the Tracking Mode, the user has several options. An important possibility is the selection between automatic and manual tracking. This can be done by clicking a radio button. Also if automatical tracking is selected the `Start` and `Continue` buttons are enabled. Pressing `Start` will begin the process of automatic tracking from the first frame on. If there is any additional tracking information beyond that frame, it will be discarded. `Continue` will only continue to track all cells from their yet found positions onward. If auto tracking is running, the stop button is enabled and clicking on it will abort the process at once.

Problem Handling Mode

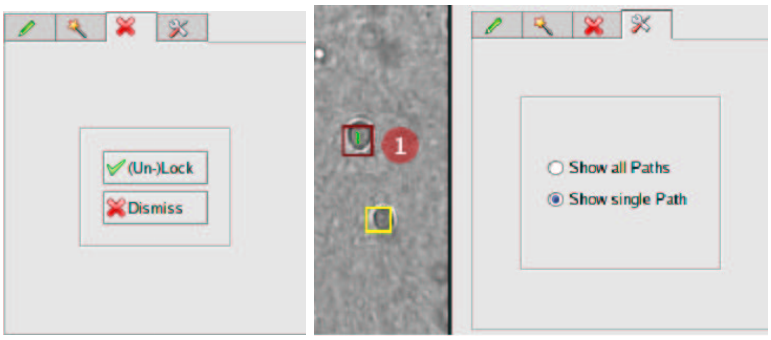


Figure B.7: Problem handling interface and options interface

Pressing `(Un-)Lock` will remove the check mark in the cell list. This unchecked cell will not be examined any more (the cell is disabled). Clicking again will re–enable the cell. If a cell is disabled, it will not be automatically tracked. The `Dismiss` button will dismiss a tracking path from the actual frame onward. A screenshot can be found in figure B.7.

Option Mode

Figure B.7 shows the options dialog. Here, the user can select if all or one path is to be displayed. Clicking will lead to the appropriate result. The red marking "1" shows path of the selected cell. Another option for the user is to decide on the tracking speed. Choosing higher tracking speeds might result in less precise data.

Video Control

As seen in most video player software and even on most hardware players there are buttons which let the user control the video. As seen in figure B.1 this program also provides some buttons. The user can play, pause and stop the video. There are arrow buttons pointing left and right, these let the user jump to the next or previous frame. The slider at the bottom allows to jump to a position anywhere in the video.

Status Bar

Figure B.8: Status Bar

The status bar is at the bottom of the frontend (figure B.1 number "6 " and figure B.8). It displays useful information, for example how many cells are marked right now and how many percent of the overall tracking process has been completed sucessfully.

B.2.4 Tutorials

This section contains some examples of how tracking might proceed.

Tracking complete video automatically with inline error correction

The following case shows how to track all cells automatically and to correct appearing errors.

1. **Start the program and log in the database**
   At the start of the program a database connection has to be established. There are two options: the first option is to use a database.opt file in the main folder of the program. If that file exists and contains correct data it will connect to the database automatically. The data should look like this:
   Host:[name of the databases host]
   DB:[name of the database]
   User:[username for the database]
   PW:[the user's password for the database]
   If this file does not exist or its data is incomplete, the program will ask for the missing data as the second option.

2. **Load a video**
   To load the video either use the `File` menu and click `Open` or press the button underneath the file menu.

3. **Mark a couple of cells**
   Use the left mouse button to drag some rectangles across the cells you want to track. See also section B.2.2.

4. **Track automatically**
   Click on the panel with the wand on the lower right of the window. The radio button "AutoTrack" is checked per default, so you can press the `Start` button. The program will begin to track the selected cells. In the cell list on the right side of the window one can see the status of every cell: A yellow LED means, that the program still tracks this cell, a green LED means that this cell is completed, and a red LED means that this cell has been lost.

5. **The red LEDs**
   There are some options for handling cells marked with a red LED. If for example the cell descented in one of the first frames and never emerged again, this cell should not be further considered. So it can be deleted by pressing the `Delete` button in the edit panel. Other cells may have been collided. In this case, it is advisable to track by hand past this problem. With a double click on the cell entry in the cell list, the video jumps to the frame, where the cell was last seen. After clicking the radio button "Manual Track", the user can step a few frames backwards to begin the manual tracking at a frame before the cell tracking has failed. Then the mouse cursor should hover on top of the cell. By clicking the mouse button, the video will start to play and the user can track the cell with the mouse cursor. To stop the video and the tracking, the user has to press the mouse button again. Clicking again will repeat the process. When the critical situation has passed and the cell is in a good position to track automatically, the manual tracking can be stopped. The LED will turn yellow to show that this cell is ready to be tracked onward. If there is another cell, which shall be tracked manually, the user can repeat this process. If all cells with red LEDs are revised, the radio button "Auto Track" and then the `Continue` button have to be pressed.

6. **End of tracking**
   The tracking ends when all remaining cells are marked with a green LED.

7. **Commit to database**
   Once the tracking is finished the data has to be committed to the database. This can be done in in the menu `Extra`. If no data is commited, the data cannot be used in the future.

Tracking a cell manually

This case shows how the user can mark and track a cell throughout the whole video. See section B.2.2 for a rough overview.

1. **Starting the program and log in the database**
   At the start of the program a database connection has to be established. There are two options: the first option is to use a database.opt file in the main folder of the program. If that file exists and contains correct data it will automatically connect to the database. The data should look like this:
   Host:[name of the database's host]
   DB:[name of the database]
   User:[username for the database]
   PW:[the user's password for the database]
   If this file does not exist or its data is incomplete, the program will ask for the missing data as the second option.

2. **Starting the program and load a video**
   To load the video either the `Open` item from the `File` menu or the button underneath the `File` menu can be used.

3. **Marking a couple of cells**
   The user has to switch to the correct mode for marking cells. This mode is marked by a green pen and should be selected by default. See also section B.2.2.

4. **Setting up the mode and the cell**
   Now the user has to switch to the Tracking Mode. The mode is marked with a wand. By default automatic tracking is selected, consequently the user has to switch to manual tracking. Before he can start tracking by hovering on top of the cell, a cell must be selected. Clicking on the cell list selects a cell – only this cell will be highlighted.

5. **Tracking by hovering with the mouse cursor**
   Now clicking on the screen will start the video, clicking again will stop the video. Once clicked the shape of the cursor changes. Now manual tracking is active. The user has to follow the cell with the cursor. Clicking again will stop the video and the cursor will change back to normal.

B.3 How to use the grabber

B.3.1 The grabbing tool

The grabbing tool was designed to store the video data to a compressed video format, e.g. the dirac format (.drc). Additionaly the number of frames, the file path, the user name, the cell type and other data needed for statistic analysis are stored to a database.

The user interface

Figure B.9 shows the grabbing tool as seen by the user directly after starting the program.

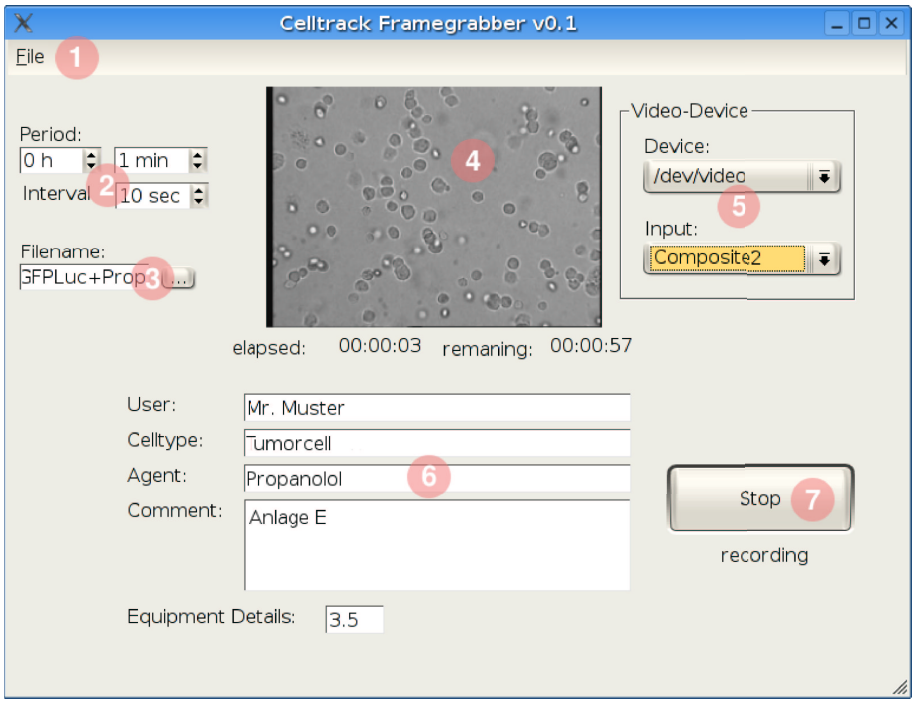


Figure B.9: The user interface

1. Menu Bar
   The menu bar provides selecting the name and path of the output video.

2. Time Selection
   Here the user can select the period and the interval of the experiment.

3. The New Dialog
   At the beginning, the user needs to choose a filename and a place where the video should be saved. The dialog is shown in figure B.10.

4. The Screen
   The screen shows the actual picture of the video camera.

5. The Device Selection
   From the drop-downmenues the user can select the video device and the input channel of the video device (see figure B.10).

6. The Experiment Data
   The textboxes can be filled with information about the experiment.

7. The Start Button
   The `Start` button starts recording the video.

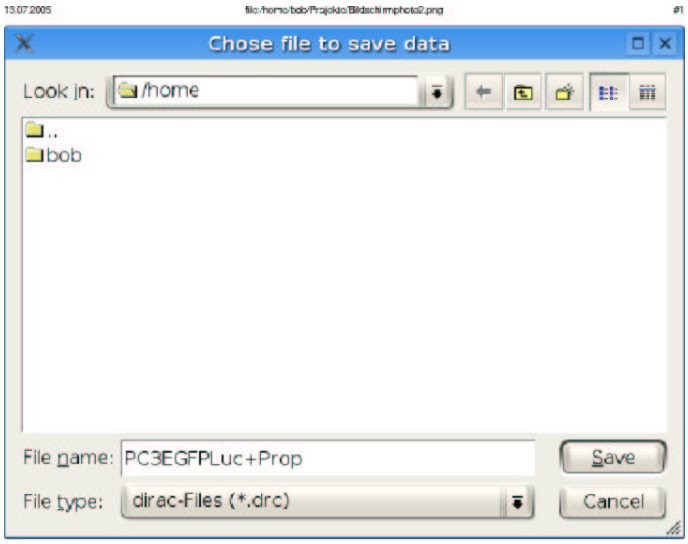


Figure B.10: Selecting filename and path

Quick guide – How to grab a video

After setting the video device and input channel (see figure B.9, red "5") the user has to select a filename and the path the video should be stored to. He/She can open the `New` dialog by pressing the `...` button, next to the filename (see image B.9 red "3"), or using the `File` menu.
The next step is to select the period and interval of the video (see figure B.9, red "2").
The last step before starting the grabbing process is to collect some information about the experiment. Here the user can provide information about the username, cell type, agent, equipment details (micrometers per pixel) and comments (see figure B.9, red "6").
Finally, the `Start` button starts recording (see figure B.9, red "7") and then becomes a `Stop` button. After the chosen period recording stops automatically. If the user wants to stop recording before, he/she has to press the `Stop` button.

B.4 How to use statistics

B.4.1 Statistics

The statistics tool was designed to reduce the workload when analysing the videos. It can be accessed through the item `Statistics` in the `Extra` menu. For running the statistics, the video

data has to be available. That means, that a sufficient number of videos need to be completely tracked and saved in the database.

At the beginning, if the database connection is not available, the user needs to log in to the database with the host, database name, username and password. If the database is running and data is present the statistics wizard will open.

Username Dialog

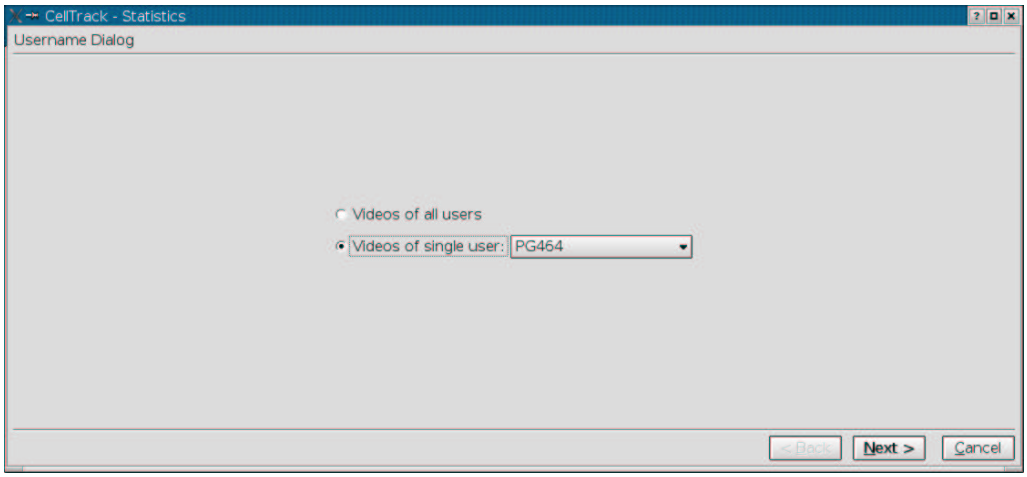


Figure B.11: Selecting the username

In this window the user can choose, if he/she wants to see the videos of all users or only by a specific user. The list in the drop down menu holds all users with at least one sucessfully tracked video. Once the user has made a choice he/she can continue to the next page using the `Next>` button.

Selection Dialog

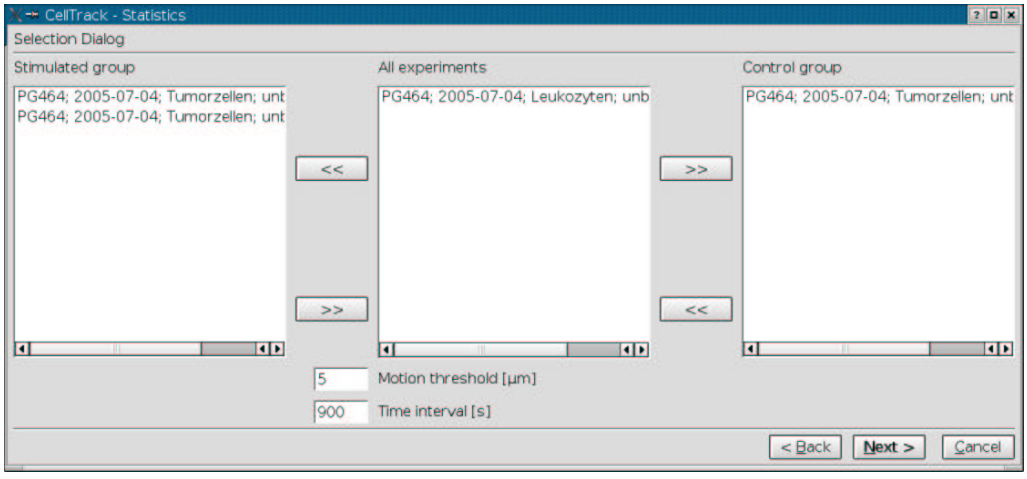


Figure B.12: Selecting the videos to be analyzed

In the middle of the page a list is shown containing all the experiments with sucessfully tracked cells. They are ordered by user name, date, cell type, the agent used on the cells, and an internal video number. The user can now use the [<<] and [>>] buttons to move videos to the control group or stimulated group. A recommendation is to use at least three videos in each section. It is mandatory to use at least one video in each group to enable the [Next>] button and to continue.

The user must also fill in two numbers in the boxes on the bottom of this page. He/She needs to decide the distance a cell needs to travel to be considered active. He/She also needs to decide on the time interval offered to the cell for moving this distance.

Once the [Next>] button is pressed warnings may come up. If the stimulated or the control group holds experiments with different cell types or agents, a warning will be shown. If this warning is exited with [Ok] the program continues. Elsewise the user has the chance to change his selection. If the videos have different recording speeds the video analysis will fail and the user must change his selection before he can continue.

Graph Dialog

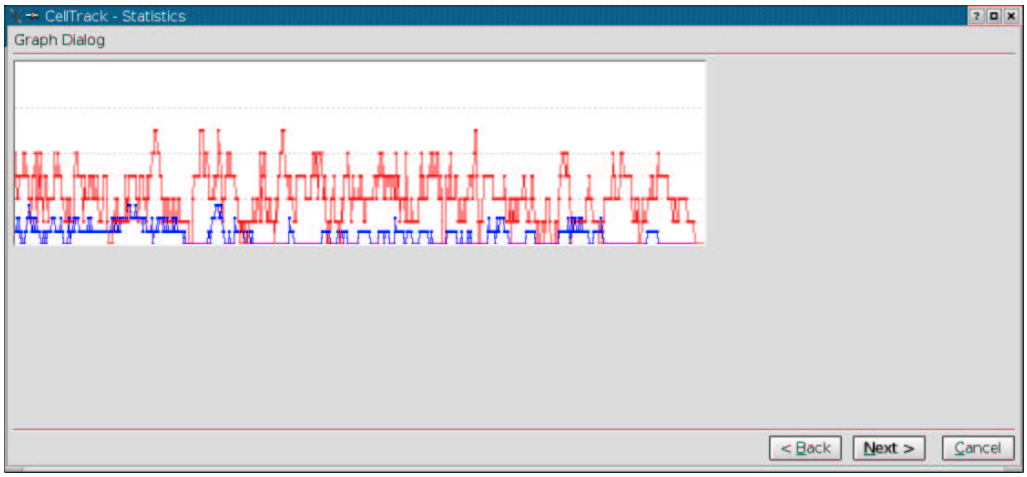


Figure B.13: The steady state graph

This page presents the two graphs of the cell activities. On the x-axis the time is plotted and on the y-axis the percentage of the active cells at that time is plotted. The control group is symbolised by the blue line and the stimulated group is symbolised by the red line. At this point the option is given to define a steady state interval on the video. This interval will be the basis for all the further calculations. The starting and ending point of this interval can be selected by moving the mouse on the corresponding parts of the graph. By doing so, the selection will be visualised with two vertical green lines. Clicking the left mouse button will fix these lines. If the [Next>] button is pressed without fixing lines, the program will use the whole video for future calculations. If the steady state reaches from one point to the ending of the video, only the initial line needs to be fixed.

Result Dialog

Now the results of the statistical analysis are given. The results of the statistics are presented in a table. In the first rows the average numbers of the control videos are presented followed by the data of the stimulated videos. The values presented are:

Activity
This number represents the average percentage of cells active in the video.

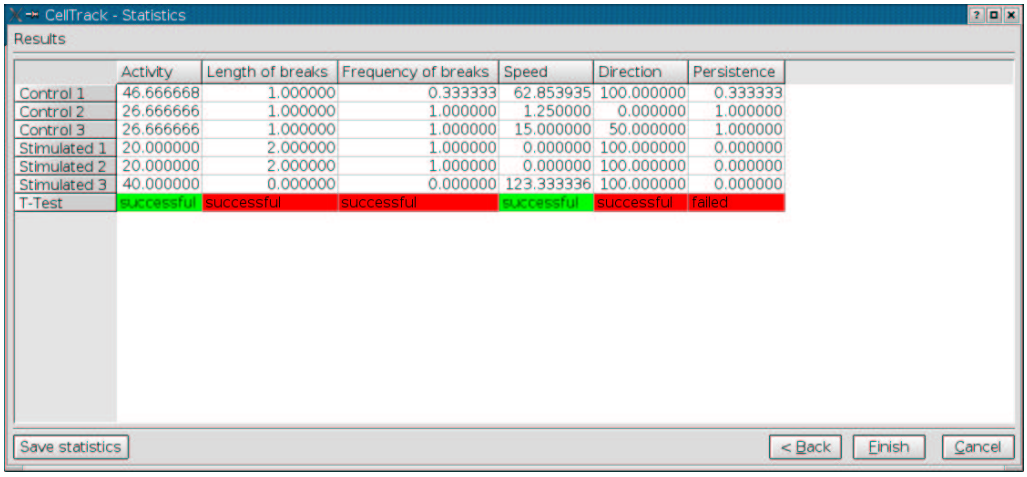


Figure B.14: Displaying the results

Length of breaks
This is the average length of breaks of all the active cells presented in frames measured in s.

Frequency of breaks
This is the average amount of breaks of all the active cells. A break is definied as consecutive n frames ($n \geq 1$) where the cell is not active.

Speed
The average speed of all active cells in the video measured in $\mu m/s$.

Direction
This is the percentage of cells moving right (towards the agent). The cell direction is defined as the angle between the vector defined by the start and end point and the horizontal axis calculated counterclockwise. Cells are considered moving right if this angle is less than 90 or greater than 270 degrees.

Persistence
This is the average of the quotients from the active cells defined by

$$\frac{\text{Euclidean distance between the start and the end point}}{\text{the distance travelled by the cell}}.$$

In the bottom line the results of the t-test are shown. If the χ^2-test for the normal distribution fails, the background of the table entry is red. In that case the user should not accept the t-test without further calculations by hand.

If the user wants to save the cell positions in the video and all the data presented in this page, he can press the `Save Statistics` button. Here he/she can decide on a filename and location for saving the data. These files can be opened by any Microsoft Excel® application.

The math behind the screen

For the interested reader an overview of the math used in the statistics tool will be given now. This will help to understand the data presented and saved by the tool.

Activity

The activity in this program is defined the following way:

A cell has to move a certain distance in a certain amount of time to be considered active. These thresholds are given by the user during the analysis. The tool defines for each frame of the movie, if the cell is active in that frame. Since the last frames do not have a corresponding frame in the future to compare the distance to, these frames are considered inactive for the purpose of the analysis. Once the activity of each cell in each frame has been determined, the percentage of active cells in each frame is calculated. This number is averaged over the group and shown in the graph. The number in the table shows the average activity in a video during the steady state interval.

Different ways of defining the activity of a cell can be found, but CellTrack uses this one due to the highest accuracy of the algorithms known during development.

t-test

For this analysis the unpaired and undirected t-test is used. This test works on two sets of values interpreted as probability distributions. It accepts or declines the null hypothesis which states, that the difference between the two distributions is only coincidental. The t-test works with an alpha error of 5%, i.e. the null hypothesis is falsely declined with a probability of 0.05.

χ^2-test

The t-test used above has as the mandatory prerequisite that the data has to obey to a Gaussian distribution. The χ^2-test allows to test for this distribution with an alpha error of 5%. If this test fails the user should manually apply a non-parametric test like the Mann-Whitney-U test.

Literaturverzeichnis

[AB94] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Trans. on PAMI*, 16(6):641 –647, 1994.

[Bar03] Kai Uwe Barthel. Verlustlose Bildkompression. *IT – Information Technology*, 45(5):247–255, 2003.

[Ber96] Manfred Berndtgen. Handlich bunt, Kompressionstechniken für Bild- und Videodateien im Vergleich. *Computer Technik*, (11):222ff, 1996.

[BKI03] C. Beierle and G. Kern-Isberner. *Methoden wissensbasierter Systeme*. Vieweg Verlag, 2. Auflage, 2003.

[BMG] Pressemitteilung des Bundesministeriums für Gesundheit und Soziale Sicherung vom 28.04.2004.
http://www.bmgs.bund.de/deu/gra/aktuelles/pm/bmgs04/5150_5267.cfm.

[Cam94] T. A. Camus. Real-time Optical Flow. Technical Report CS-94-36, Department of Computer Science, 1994.

[com] Data Compression Reference Center: The LZW Algorithm.
http://www.cs.usyd.edu.au/ loki/cs2csys/gif-info/lzw.html.

[dir] Dirac. http://dirac.sourceforge.net/.

[Dir99] B. Dirks. Video for Linux Two - Devices.
http://www.thedirks.org/v4l2/v4l2dev.htm, 1999.

[Dir01] B. Dirks. Video for Linux Two - Video Capture API Specification.
http://www.thedirks.org/v4l2/v4l2cap.htm, 2001.

[Ess04] Uniklinikum Essen. Schonende Strahlen gegen Krebs. *Klinikum Kurier*, Juni 2004.

[Fie00] M. Fiedler. Projektarbeit Datenkompression.
http://www-user.tu-chemnitz.de/˜mfie/compproj, 2000.

[GW02] R. Gonzales and R. Woods. *Digital Image Processing*. Prentice-Hall, Inc., New Jersey, 2. Auflage, 2002.

[Hei96] Bernd Heinrichs. *Multimedia im Netz, Bildcodierung und Videokommunikation*. Springer, 1996.

[IEC98] Norm: IEC60027-2, letter symbols to be used in electrical technology - part 2: Telecommunications and electronics, 1998.

[JZ04] F. Entschladen, T. L. Drell IV, K. Lang, J. Joseph and K. Zaenker. Tumor-cell Migration, Invasion, and Metastasis: Navigation by Neurotransmitters. *THE LANCET*, 5, April 2004.

[Ker03] D.M. Kerschner. Snakes für Aufgaben der digitalen Photogrammetrie und Topographie. Technical Report P13725-MAT, Institut für Photogrammetrie und Fernerkundung, 2003.

[KM01] P. Vachenauer K. Meyberg. *Höhere Mathematik 2*. Springer Verlag, 4. Auflage, 2001.

[Kre02] U. Krengel. *Einführung in die Wahrscheinlichkeitstheorie und Statistik*. Vieweg Verlag, Braunschweig/Wiesbaden, 6. Auflage, 2002.

[Käh95] W. M. Kähler. *Einführung in die Statistische Datenanalyse. Grundlegende Verfahren und deren EDV-gestützter Einsatz*. Vieweg Verlag, Braunschweig/Wiesbaden, 1. Auflage, 1995.

[LCPL01] Sven Loncaric, Ivan Ceškovic, Ratimir Petrovic, and Srecko Loncaric. 3D quantitative analysis of brain spect images. *Proc. SPIE Vol. 4322, Medical Imaging 2001*, 4322:1689–1695, 2001.

[LL93] F. Leymarie and M. Levine. Tracking Deformable Objects in the Plane Using an Active Contour Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):617–634, 1993.

[LW85] J. Lehn and H. Wegmann. *Einführung in die Statistik*. Wissenschaftliche Buchgesellschaft Darmstadt, Darmstadt, 1. Auflage, 1985.

[Mat00] Günter Matthiessen. *Relationale Datenbanken und SQL*. Addison-Wesley, 2000.

[MKT87] A. Witkin M. Kass and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1:321–331, 1987.

[MRA04] Dipti Prasad Mukherjee, Nilanjan Ray, and Scott T. Acton. Level Set Analysis for Leukocyte Detection and Tracking. *IEEE Transactions on Image Processing*, 13(4), 2004.

[MV96] James D. Murray and William VanRyper. *Encyclopedia of graphics file formats*. O'Reilly, 1996.

[NRL02] S.T. Acton N. Ray and Klaus Ley. Tracking Leukocytes In Vivo With Shape and Size Constrained Active Contours. *IEEE Transactions on Medical Imaging*, 21(10):1222–1235, 2002.

[NZ00] F. Entschladen, M. Gunzer, C. M. Scheuffele, B. Niggemann and K. S. Zänker. T Lymphocytes and Neutrophil Granulocytes Differ in Regulatory Signaling and Migratory Dynamics with Regard to Spontaneous Locomotion and Chemotaxis. *Cellular Immunology*, 199(2):104–114, Academic Press, 2000.

[Pei85] J. Peil. *Grundlagen der Biomathematik*. Verlag Harri Deutsch, Frankfurt/Main, 1. Auflage, 1985.

[pos] PostgreSQL. http://www.postgresql.org.

[RG] B. Rudiak-Gould. Huffyuv project homepage. http://neuron2.net/www.math.berkeley.edu/benrg/huffyuv.html.

[RP] G. Randers-Pehrson. MNG (Multiple-image Network Graphics) Format Version 1.0. http://www.libpng.org/pub/mng/spec/.

[Set99] J.A. Sethian. *Level Set Methods and Fast Marching Methods - Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.

[Son99] M. Sonka. *Image Processing, Analysis and Machine Vision*. PWS Publishing, Bonn, 2. Auflage, 1999.

[sql] SQL. http://www.mysql.de.

[Ste93] R. Steinbrecher. *Bildverarbeitung in der Praxis*. Oldenbourg Verlag, Münschen, Wien, Oldenburg, 1993.

[Sut02] A. Sutcliffe. *Multimedia User Interface Design*. University of Manchester Institute of Science and Technology, 2002.

[T+94] L. Tripp et al. *IEEE recommended practice for software requirements specifications*. Inst. of Electrical and Electronics Engineers, 1994.

[VC02] Luminita A. Vese and Tony F. Chan. A Multiphase Level Set Framework for Image Segmentation - Using the Mumford and Shah Model. *International Journal of Computer Vision*, 50(3):271–293, 2002.

[Wag96] C. Wagener. *Molekulare Onkologie*. Thieme Verlag, 1996.

[Was04] Michael Wasilewski. Active Contours using Level Sets for Medical Image Segmentation. http://www.cgl.uwaterloo.ca/m̃mwasile/cs870/, 2004.

[WB01] G. Welsh and G. Bishop. An introduction to the Kalman Filter. *SIGGRAPH2001 Course 8*, 2001.

[Wel83] Terry A. Welch. United States Patent: 4,558,302, High speed data compression and decompression apparatus and method, 1983.

[XP98] C. Xu and J.L. Prince. Snakes, Shapes, and Gradient Vector Flow. *IEEE Transactions on image processing*, 7(3):359–369, 1998.

[Zau01] A. Zaugg. Snakes: Active Contour Models. Studienarbeit, www.medialab.ch/archiv/pdf_studien_diplomarbeiten/ 1sa02/snake(zaugg&maechler).pdf, 2001.

[ZD03] Lodish, Berk, Matsudaira, Kaiser, Krieger, Scott, Zipursky and Darnell. *Molecular Cell Biologie*. eBook, 5. Auflage, 2003. Kapitel 1 und 23.

[ZE03] T.L. Drell IV, J. Joseph, K. Lang, B. Niggemann, K.S. Zaenker and F. Entschladen. Effects of neurotransmitters on the chemokinesis and chemotaxis of MDA-MB-468 human breast carcinoma cells. *Breast Cancer Research and Treatment*, 80:63–70, 2003.

[ZE04] B. Niggemann, T. L. Drell IV, J. Joseph, C. Weidt, K. Lang, K. S. Zaenker and F. Entschladen. Tumor cell locomotion: differential dynamics of spontaneous and induced migration in a 3D collagen matrix. *Experimental Cell Research*, 298(1):178 – 187, Elsevier Inc., 2004.

[ZL77] Jacob Ziv and Abraham Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[ZL78] Jacob Ziv and Abraham Lempel. Compression of Individual Sequences via Variable-Rate Coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.