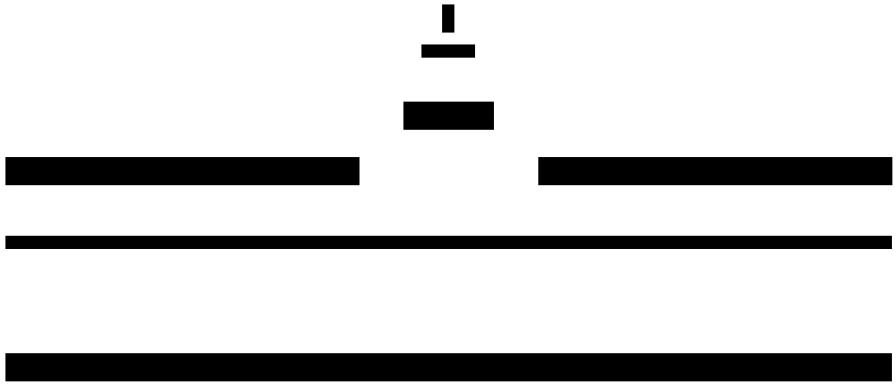

Westfälische Wilhelms-Universität
Münster

Masterarbeit zur Erlangung des akademischen
Grades „Master of Science“

Automatische Detektion von junction-associated intermittent lammelipodia (JAIL) in Mikroskopie-Videos von Endothelzellen

Autor:
Stefan Nückel

Betreuung:
Prof. Dr. Benedikt Wirth

Matrikelnummer:
371735

Institut für numerische und angewandte Mathematik
Münster, den 30. Oktober 2016

Zusammenfassung

In dieser Arbeit entwickeln wir einen Algorithmus, der junction-associated intermittend lammelipodia (JAIL) in Mikroskopie-Videos von Endothelzellen automatisch detektiert.

Dazu werden wir neben dem biologischen Hintergrund der JAIL zunächst diverse grundlegende Verfahren aus der Bildverarbeitung kennen lernen. Im weiteren Verlauf der Arbeit geht es um die explizite Verarbeitung eines JAIL-Videos. Dies schließt sowohl die Vorverarbeitung durch Entrauschung und Segmentierung der Daten als auch die Anwendung einiger selbst entwickelter Methoden zur Detektion von JAIL mit ein.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich sowohl während dieser Arbeit als auch während meiner gesamten Studienzeit unterstützt haben.

Ganz herzlich möchte ich mich bei Prof. Dr. Benedikt Wirth bedanken, der durch seine kompetente Art ein sehr guter Betreuer war und es mir erst ermöglicht hat mich mit diesem spannenden Thema zu beschäftigen.

In diesem Zusammenhang geht mein Dank auch an Prof. Dr. Med. Hans-Joachim Schnittler und Dr. rer. nat. Jochen Seebach aus dem Institut für Anatomie und Vaskuläre Biologie der WWU Münster, die für die Entdeckung der JAIL und die Weitergabe dieses Themas an uns verantwortlich waren und uns einen Einblick in den biologischen Hintergrund der JAIL gegeben haben.

Meinen besonderen Dank möchte ich Ulrich Hartleif aussprechen, der mich über die gesamte Zeit dieser Arbeit hin sehr gut betreut, mir hilfreiche Tipps und Ratschläge mit auf den Weg gegeben und einige Stunden für gemeinsame Besprechungen geopfert hat.

Weiterhin möchte ich mich bei Eva-Maria Brinkmann für eine Einführung in den Algorithmus von Chambolle und Pock sowie bei Dr. Hendrik Dirks für die Bereitstellung und Erklärung seiner Motion-Estimation-Toolbox bedanken.

Ein herzliches Dankeschön richte ich an meine Familie, die mich während der gesamten Zeit meines Studium nicht nur finanziell, sondern auch moralisch unterstützt hat und immer für mich da ist, wenn ich sie brauche.

Dies gilt auch für meine Freundin Christine Mucha, die selbst in den stressigsten Phasen des Studiums und dieser Arbeit immer für mich da war, mich ausgehalten und immer wieder erfolgreich motiviert hat. Vielen Dank dafür.

Zu guter Letzt bedanke ich mich ganz herzlich bei meinen Freunden und Kommilitonen, ganz besonders Magdalena Mayerhoffer und Paul Freitag, die mir die wunderbare Zeit meines Studiums durch gemeinsames Lernen und Zettelrechnen erheblich erleichtert haben und mit denen ich selbst langweilige Stunden ereignisreich und lustig gestalten konnte.

Inhaltsverzeichnis

1 Einleitung **1**

2 Biologischer Hintergrund **3**

3 Grundlagen der Bildverarbeitung **6**
3.1 Was ist ein Bild? 6
3.2 Wie verarbeitet man ein Bild? 7

4 Vorverarbeitung **13**
4.1 Extraktion des p20-Kanals 13
4.2 Entrauschung 13
4.3 Segmentierung 20

5 Verfahren zur JAIL-Detektion **23**
5.1 Detektion kreisförmiger Strukturen 23
5.1.1 Die Hough-Transformation 23
5.1.2 Faltung mit Gauss-Kreisen 27
5.1.3 Gauss-Halbkreise und Richtungsableitungen 31
5.2 Optischer Fluss 36
5.3 Aufsummierung der Intensitäten 37
5.4 Connected Components 39
5.5 Faltungswerte akkumulieren 42

6 Workflow **47**

7 Fazit und Ausblick **55**

Literaturverzeichnis **57**

1 Einleitung

In dieser Arbeit beschäftigen wir uns mit der Detektion von junction-associated intermittent lammelipodia, kurz JAIL. Diese sind noch nicht sehr lange erforscht, jedoch besteht die Vermutung, dass sie eine wichtige Rolle für medizinische und biologische Aspekte jeglicher Art spielen. Das Ziel dieser Arbeit ist es, einen Algorithmus zu erstellen, der JAIL in Mikroskopie-Videos von Endothelzellen automatisch detektiert. Dazu haben wir von Prof. Dr. Med. Hans-Joachim Schnittler und Dr. rer. nat. Jochen Seebach aus dem Institut für Anatomie und Vaskuläre Biologie der WWU Münster solche Videos gestellt bekommen. Das in dieser Arbeit betrachtete Video befindet sich als MAT-Datei mit dem Namen *p20_video.mat* auch auf der beiliegenden CD. Es zeigt Endothelzellen und die immer wieder dazwischen auftauchenden JAIL-Formationen.

Für die Programmierung haben wir Matlab von der Firma MathWorks in der Version R2014b benutzt. Die Ergebnisse der Programmierung befinden sich ebenfalls auf der beiliegenden CD.

Die Arbeit strukturiert sich wie folgt. In Kapitel 2 werden wir zunächst den biologischen Hintergrund sowie die Bedeutung der JAIL sehen. Das Hauptaugenmerk liegt dabei auf dem Zusammenhang zwischen JAIL und Endothelzellen sowie der Funktion der JAIL innerhalb eines Endothels.

Daraufhin geht es in Kapitel 3 mit den Grundlagen der Bildverarbeitung weiter. Hier führen wir zunächst die mathematische Definition eines Bildes ein, um dann einige grundlegende und häufig genutzte Verfahren der Bildverarbeitung kennenzulernen.

Kapitel 4 beschäftigt sich mit den ersten Schritten, die wir zur Vorverarbeitung der Rohdaten angewandt haben. Dies beinhaltet neben der Extraktion des für uns essentiellen Video-Kanals das Entfernen des Bildrauschens mithilfe des Kullback-Leibler-Datenterms und der TV-Regularisierung sowie die Segmentierung der Zellwände zur Eingrenzung der möglichen JAIL-Koordinaten.

In Kapitel 5, dem Hauptteil dieser Arbeit, geht es um die Verfahren, die für die Detektion der JAIL getestet wurden. Hier lernen wir zunächst verschiedene Verfahren zur Suche nach kreis- und halbkreisförmigen Strukturen kennen. Die darauf

folgenden Verfahren dienen zur Überprüfung der gefundenen und zur Entfernung falsch positiver JAIL.

Kapitel 6 beschäftigt sich anschließend mit dem Workflow, der im Endeffekt von uns genutzt wurde. Hier wird verdeutlicht, inwieweit die in den vorherigen Kapiteln erklärten Verfahren zu einem Algorithmus zusammengesetzt wurden. Des Weiteren sehen wir hier das Ergebnis des Algorithmus anhand eines JAIL-Beispiels.

Zum Schluss erfolgt in Kapitel 7 ein Fazit mit einem Ausblick auf weitere Forschungen. Im Fazit werden zunächst die wichtigsten Inhalte und Ergebnisse der Arbeit noch einmal zusammengefasst. Mit dem Ausblick geben wir daraufhin einige Ansatzpunkte, in welche Richtung mögliche weiterführende Forschungen zum Thema dieser Arbeit gehen könnten.

2 Biologischer Hintergrund

In diesem Kapitel werden wir uns mit dem biologischen Hintergrund der junction-associated intermittent lammelipodia (JAIL), beschäftigen. Wir werden uns hierbei im Wesentlichen auf [TS] beziehen.
In den uns zugrundeliegenden Videos sehen wir benachbarte Endothelzellen, die zusammen ein einschichtiges Endothel bilden, welches das Innere der Blutgefäße auskleidet. Sie bilden damit eine regulierbare Barriere zwischen dem Blutgefäß und dem Extravasalraum, also dem Flüssigkeitsraum außerhalb des Gefäßes. Durch diese Barriere lässt sich der Stoffaustausch zwischen Gewebe und Blut steuern. Die Dichte des Endothels hängt dabei von der Art des Gewebes und des dort notwendigen Stoffaustausches ab. So bilden die Endothelzellen im zentralen Nervensystem beispielsweise eine kontinuierliche, kaum durchlässige Barriere - die sogenannte „Blut-Hirn-Schranke“. Der unkontrollierte Übertritt von Blutbestandteilen oder im Blut gelösten Substanzen ist somit verhindert. In der Niere dagegen bilden die Endothelzellen ein fenestriertes Endothel, welches kleine Fenster zur Durchlässigkeit einiger Moleküle bildet. In der Milz wird sogar ein diskontinuierliches Endothel gebildet, welches Lücken in der Schicht aufweist und somit einen Durchtritt von Zellen ermöglicht (nach [BT, S. 20]).
In unseren Beispiel-Videos sehen wir Endothelzellen der Venen einer Nabelschnur („*Human umbilical vein endothelial cell*“ oder kurz „*HUVEC*“).

Die Bildung eines Endothels wird über sogenannte Adhärenz-Verbindungen realisiert. Diese sind Verbindungen zwischen Aktin-Filamenten zweier benachbarter Endothelzellen, die das Endothel mechanisch verstärken. Dafür werden sogenannte VE-Cadherine („*vascular endothelial cadherin*“) benötigt. Wird eine solche Verbindung unterbrochen, bildet sich nun ein JAIL. Er trägt zur Verschließung dieser Verbindung bei, indem er die Plasma-Membranen von benachbarten Zellen überlappt und dort die Formation neuer VE-Cadherin-Ablagerungen vereinfacht. Daraufhin können sich die Membranen an dieser Stelle wieder verschließen. Die lokale Konzentration der VE-Cadherine steht also im Zusammenhang mit der Häufigkeit einer JAIL-Formation. Daher bilden sich JAIL vorzugsweise in subkonfluenten Zellen, also an relativ lückenhaften Zell-Zell-Kontakten.
In Abbildung 2.1 sehen wir den unterschiedlichen Einfluss der Konzentration von VE-Cadherin in konfluenten bzw. subkonfluenten Zell-Zell-Kontakten.

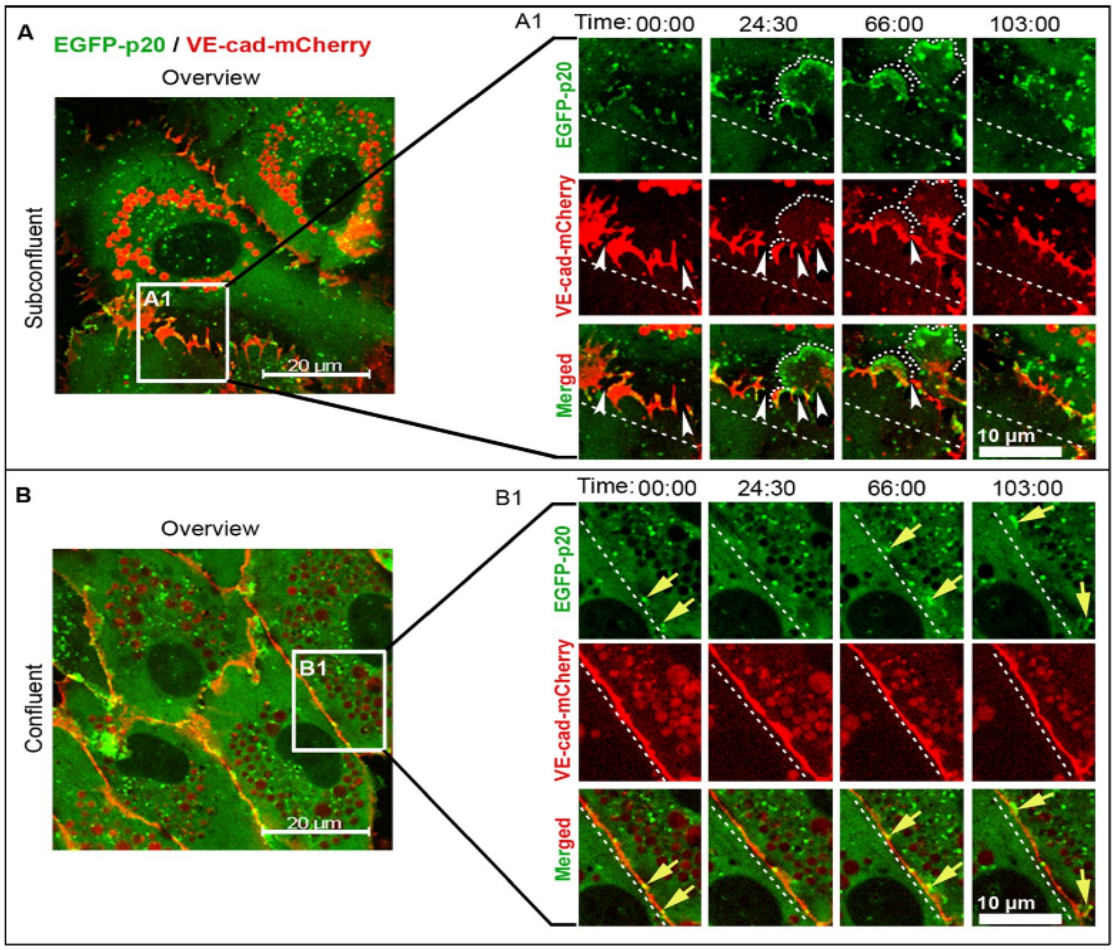

Abbildung 2.1: Unterschied der JAIL-Formation bei subkonfluenten (A) und konfluenten Endothelzellen (B) (aus [TS])

Um die VE-Cadherine von den JAIL unterscheiden zu können, wurde das für die Bildung von Aktinfilamenten benötigte Protein p20 mit einem grün fluoreszierenden Protein („*EGFP*“) und die VE-Cadherine mit einem rot fluoreszierenden Protein („*mCherry*“) markiert.
Zur Veranschaulichung der Bildung eines JAIL zeigt Abbildung 2.2 die Nahaufnahme eines wachsenden JAIL im Zeitraffer.

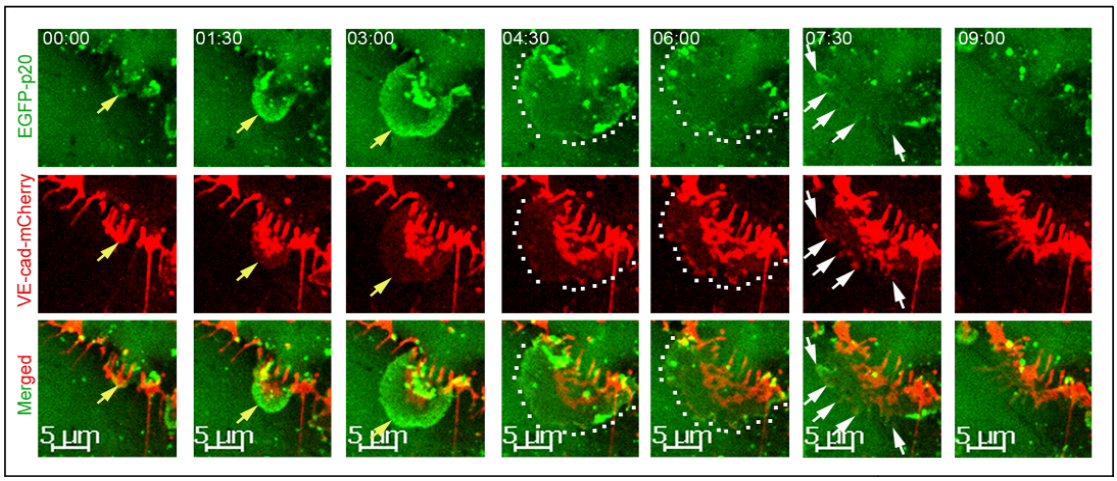

Abbildung 2.2: Aufnahme eines wachsenden JAIL (gelbe Pfeile) im Zeitraffer, der neue VE-Cadherin-Ablagerungen (gepunktete Linie) induziert (nach [TS])

Da JAIL also die Verbindungen zwischen Zell-Zell-Kontakten von Endothelzellen beeinflussen, spielen sie eine wichtige Rolle hinsichtlich der Regulierung von Gewebebildung, Zellpolarität oder Zellausbreitung. Außerdem sind sie essentielle Bestandteile der Themen Wundheilung und Entzündungen.

Die Videos der JAIL beinhalten zwei Kanäle. Einen, der die grün fluoreszierende p20-Konzentration zeigt und einen für die rot fluoreszierenden VE-Cadherine. Da die Bildung von JAIL in dem grünen Kanal deutlich wird, ist nur dieser Kanal für uns interessant. Den roten Kanal können wir dementsprechend unberücksichtigt lassen.

Weiteres zum biologischen Hintergrund der JAIL kann in [TS] nachgelesen werden.

3 Grundlagen der Bildverarbeitung

In diesem Kapitel beziehen wir uns auf [BD, Kap. 1] und gehen zunächst der grundlegenden Frage nach: „Was ist überhaupt ein Bild?“ Weiterhin stellen wir kurz die grundlegenden Verfahren in der Bildverarbeitung dar.

3.1 Was ist ein Bild?

Ein Bild kann auf verschiedene Arten entstehen. Es gibt zum Beispiel Bilder, die mit einer Digitalkamera aufgenommen wurden und alltägliche Szenen auf eine zweidimensionale Ebene übertragen. Weiterhin gibt es das Scan-Verfahren, durch das analoge Fotografien digitalisiert werden können. Die Computertomographie (CT) bietet eine weitere moderne Art, Bilder zu erstellen. Hierbei werden im Unterschied zu den vorherigen Verfahren durch Röntgenscans aus verschiedenen Richtungen dreidimensionale Bilder erzeugt. In dieser Arbeit verwenden wir Mikroskopie-Videos. Diese bestehen aus einer Folge von zweidimensionalen Bildern, die mit Mikroskopie-Technik aufgenommen wurden.
Wir sehen, dass es sich bei Bildern nicht zwangsläufig um zweidimensionale Objekte handeln muss. Daher wird im Allgemeinen von d-dimensionalen Bildern für $d \in \mathbb{N}$ gesprochen. Für diese Arbeit stellt der Fall $d = 2$ jedoch das Hauptaugenmerk dar.
Kommen wir zurück zu der Frage, was eigentlich ein Bild ist, beziehungsweise wie es im mathematischen Sinne zu verstehen ist.

Definition 1. ***(Bild)***
Ein Bild u ist eine Funktion

$$u : \Omega \to F,$$

die jedem Punkt aus dem Definitionsbereich Ω einen Farbwert aus dem Farbraum F zuordnet.

Hierbei unterscheiden wir zwischen diskreten und kontinuierlichen Bildern. Für diskrete Bilder gilt der Definitionsbereich $\Omega \subset \mathbb{N}^d$, wohingegen bei kontinuierlichen Bildern $\Omega \subset \mathbb{R}^d$ vorausgesetzt wird. In dieser Arbeit handelt es sich stets um

diskrete Bilder. Man kann sie sich als Matrix der Größe $M \times N$ mit $M, N \in \mathbb{N}$ vorstellen. Ein Element dieser Matrix wird als Bildpunkt oder Pixel bezeichnet. Der erste Pixel $u(0,0)$ liegt oben links im Bild und der letzte Pixel $u(M,N)$ unten rechts, wobei der erste Wert dem y-Wert entspricht und der zweite Wert dem x-Wert. Ein für uns typisches Bild ist also wie folgt aufgebaut:

$$\begin{bmatrix} u(0,0) & u(0,1) & \cdots & u(0,N-1) \\ u(1,0) & u(1,1) & \cdots & u(1,N-1) \\ \vdots & \vdots & & \vdots \\ u(M-1,0) & u(M-1,1) & \cdots & u(M-1,N-1) \end{bmatrix}$$

Für den Farbraum F haben wir ebenfalls verschiedene Möglichkeiten. So gilt für

- Schwarz-Weiß-Bilder bzw. Binärbilder $F = \{0,1\}$,
- Graubilder mit diskretem Farbraum und k bit Farbtiefe $F = \{0,\dots,2^k-1\}$
- Farbbilder mit k bit Farbtiefe und N Farbkanälen $F = \{0,\dots,2^k-1\}^N$.

Bei Graubildern entspricht 0 der Farbe schwarz und 1 bzw. 2^k-1 der Farbe weiß. In Farbbildern, wie zum Beispiel Bildern im RGB-Raum mit $F = \{0,\dots,255\}^3$, steigt die Intensität der drei Farben Rot, Grün und Blau mit höheren Werten. Auch hier entspricht $[0,0,0]$ schwarz und $[255,255,255]$ weiß.

3.2 Wie verarbeitet man ein Bild?

In der Bildverarbeitung gibt es verschiedene grundlegende Verfahren, die immer wieder auftauchen. Wir werden uns hier auf diejenigen Verfahren beschränken, die in dieser Arbeit von Bedeutung sind.

Das erste grundlegende Prinzip und eines der wichtigsten Elemente zur Weiterverarbeitung von Bildern ist das **Entrauschen**. Rauschen entspricht fehlerhaften Informationen in einem Bild. Dies können beispielsweise helle oder dunkle Punkte sein, die nicht zum eigentlichen Bild gehören. Solche Intensitätsfehler kann man als Realisierungen von unabhängigen Zufallsvariablen ansehen, die auf jedem Pixel getrennt arbeiten. Das einfachste Modell ist dabei das additive Rauschen. Hierbei gilt für die gemessenen Daten F bei einem optimalen Bild u und Rauschen δ

$$F = u + \delta.$$

Die Aufgabe des Entrauschens ist es, diese Störungen weitestgehend zu eliminieren, ohne dabei wichtige Strukturen im Bild zu verlieren. In Kapitel 4.2 kommen wir wieder auf dieses Thema zu sprechen.

Ein weiteres essenzielles Konzept der Bildverarbeitung ist die **Kantendetektion**. Kanten dienen dazu, Objekte im Bild voneinander zu trennen bzw. von dem Hintergrund zu unterscheiden. Zur Detektion der Kanten stehen verschiedene Operatoren zur Verfügung. Beispiele hierfür sind (nach [BB, Kap. 7.3])

- der Sobel-Operator (siehe Kapitel 5.1.1, Gleichung (5.2))
- der Vorwärts-Differenzen-Operator (siehe Kapitel 5.1.2, Gleichung (5.3)).
- der Prewitt-Operator $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ bzw. $\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

in Abbildung 3.1 sehen wir Beispiele der verschiedenen Kanten-Operatoren.

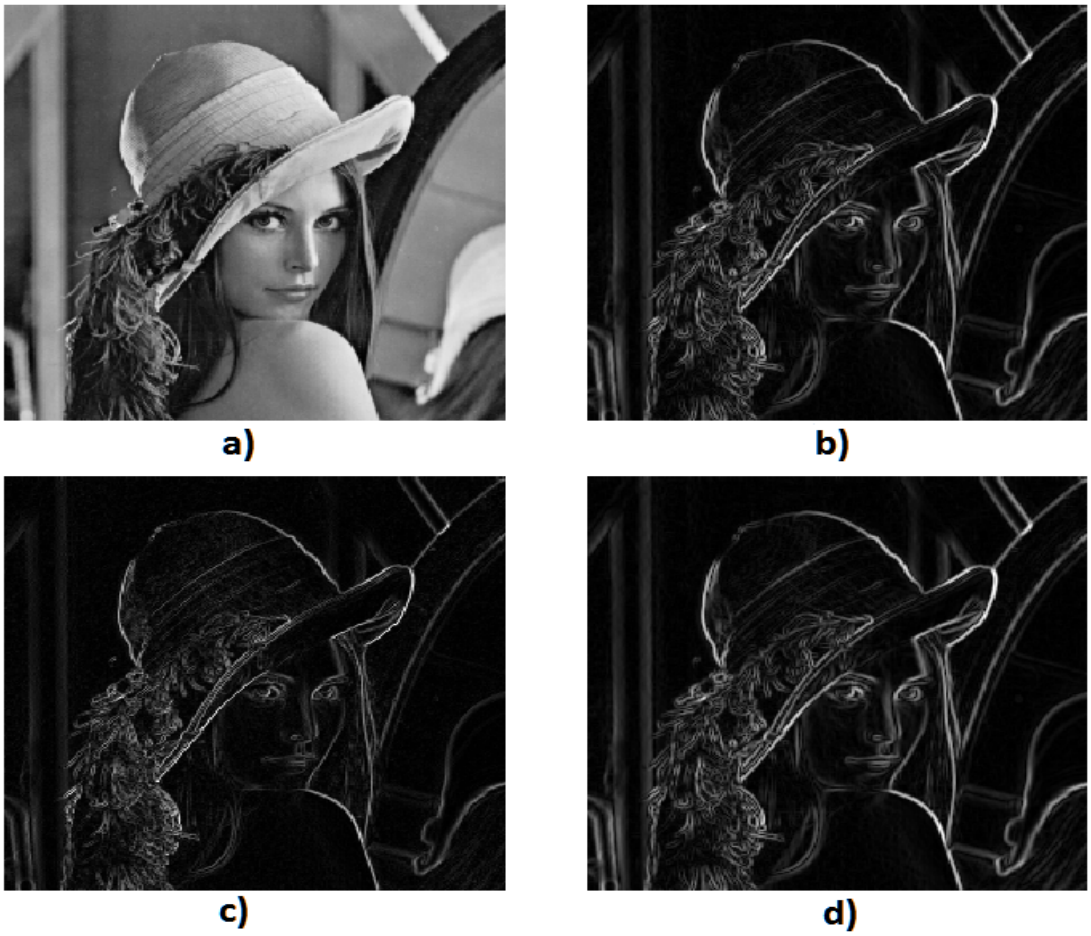

Abbildung 3.1: Kantenbilder am Beispielbild Lena. Originalbild (a) und Kantenbilder mit Sobel-Operator (b), Vorwärts-Differenzen-Operator (c) und Prewitt-Operator (d).

Wir werden vor allem in Kapitel 5.1 auf die Benutzung von Kantenbildern zurückgreifen.

Die Kanten-Operatoren nutzen das Prinzip der **Faltung**, was uns zur nächsten wichtigen Grundlage für diese Arbeit bringt.

Definition 2. *(**Faltung** - nach [We, S.265]).*
Die Faltung zweier $\mathcal{L}^1$-Funktionen f und g auf $\mathbb{R}^d$ ist definiert als

$$(f * g)(x) = \int_{\mathbb{R}^d} f(x-y)g(y)dy$$

bzw. im diskreten 2-dimensionalen Fall

$$(f * g)(x,y) = \sum_{\mu=-m}^{m} \sum_{\nu=-n}^{n} f(x-\nu, y-\mu)g(\nu,\mu).$$

für einen Faltungskern g der Größe $(2m+1) \times (2n+1)$.

Im diskreten Fall bedeutet dies, dass der Faltungskern, also beispielsweise einer unserer Kanten-Operatoren, einmal über jeden einzelnen Pixel des Bildes gelegt wird. Der Kern wird dabei mit seinem Mittelpunkt $(0,0)$ auf den jeweils zu betrachtenden Pixel gelegt, was voraussetzt, dass er eine Größe von $(2m+1)\times(2n+1)$ mit $m, n \in \mathbb{N}_0$ hat. Üblicherweise wählt man hier $m = n$. Der Faltungswert des aktuellen Pixels berechnet sich dann aus dem Skalarprodukt der Pixel des Kerns und derjenigen Pixel des Bildes, die von dem Kern abgedeckt werden. Hierbei ist zu beachten, dass der Wert oben links im Kern mit dem Wert unten rechts im Bildausschnitt multipliziert wird, der zweite Wert der oberen Zeile des Kerns mit dem zweitletzten Wert der unteren Zeile des Bildausschnitts usw. Für die Berechnung der Randwerte wird als Wert außerhalb des Bildrandes die 0 gewählt. So entsteht ein Faltungsbild, welches hohe Werte an den Stellen aufweist, wo eine möglichst große Übereinstimmung des Bildes mit dem Faltungskern auftritt. Eine Veranschaulichung der diskreten Faltung sehen wir an einem Beispiel in Abbildung (3.2).

Wir wollen nun ein Verfahren zur Detektion parametrisierbarer Strukturen einführen, die **Hough-Transformation**. Dieses wird auf einem Binärbild ausgeführt, welches zunächst mittels Kantendetektion erstellt wird. Daraufhin wird ein Parameterraum gebildet, in den für jeden Punkt auf einer Kante alle möglichen Parameter für diejenige Struktur eingetragen werden, nach der gesucht wird. Somit wird aus dem Problem der Merkmalsextraktion ein Problem der Maximumfindung im Parameterraum, welches im Allgemeinen leichter zu lösen ist. Außerdem ist bei diesem Verfahren keine Konturverfolgung notwendig, was eine hohe Robustheit gegenüber Lücken oder Störungen in einer dieser Strukturen bedeutet.

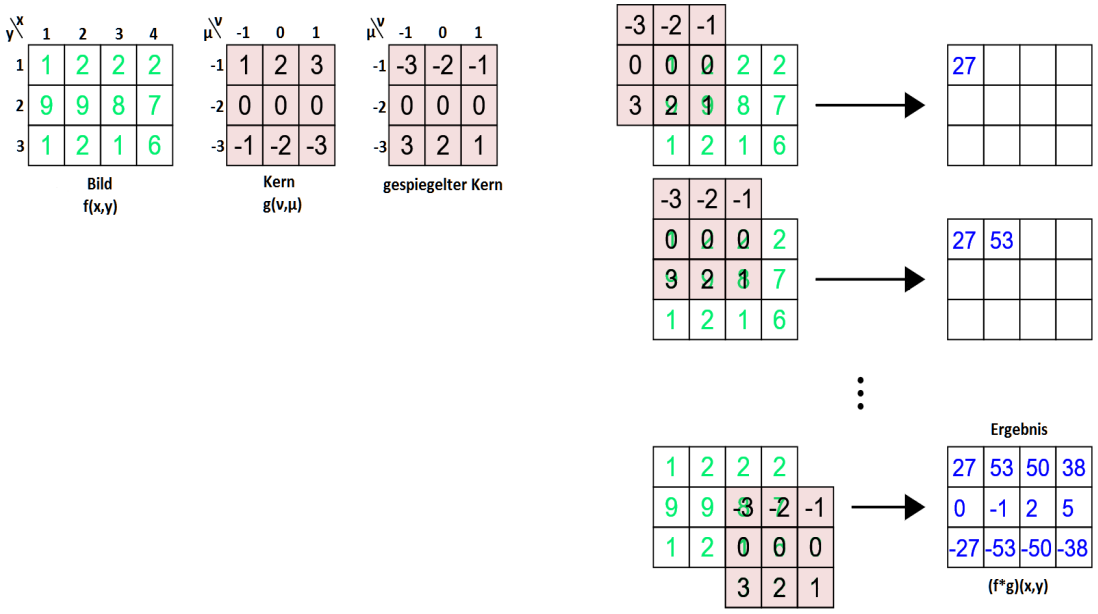

Abbildung 3.2: Visualisierung der diskreten Faltung für ein (3×4)-Bild mit einem (3×3)-Kern (nach [T])

Ein Beispiel für eine parametrisierbare Struktur ist eine Gerade. Diese kann im Allgemeinen in der Form

$$G_{mb} : y = mx + b$$

beschrieben werden, wobei (x, y) einen Punkt auf der Geraden, m ihre Steigung und b den Schnittpunkt mit der y-Achse darstellt. Betrachtet man einen Punkt (x_i, y_i) auf einer Kante, so lässt er sich durch unendlich viele Geraden mit der Eigenschaft $y_i = mx_i + b$ erzeugen. Alle diese Geraden können im mb-Parameterraum als eine Gerade

$$b = -x_i m + y_i$$

dargestellt werden. Betrachtet man nun eine Gerade $G_{m_0 b_0}$ im Bild, so führt jeder Punkt auf ihr zu einer Geraden im mb-Raum. Alle diese Geraden schneiden sich dann im Punkt (m_0, b_0), sodass der Parameterraum an dieser Stelle ein Maximum aufweist (nach [J]). Abbildung 3.3 verdeutlicht den Zusammenhang zwischen Bild- und Parameterraum. Näheres zu unserer Nutzung der Hough-Transformation wird in Kapitel 5.1.1 erläutert.

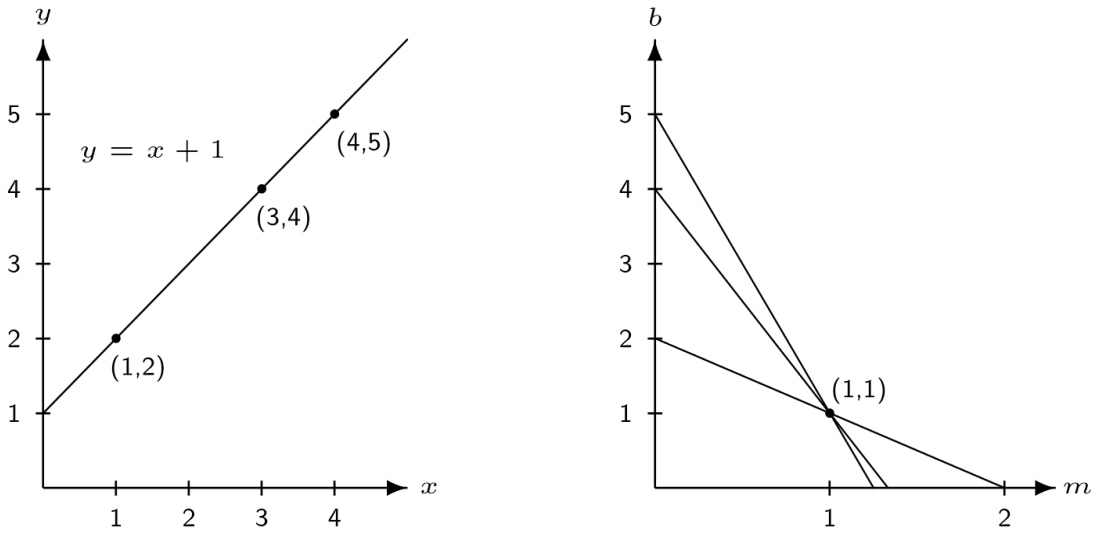

Abbildung 3.3: Die Punkte (1,2), (3,4) und (4,5) im Bildraum (links) führen jeweils zu einer Geraden im mb-Raum (rechts). Diese schneiden sich im Punkt (1,1), was auf eine Gerade der Form $y = x + 1$ im Bild hindeutet (nach [J]).

Hat man wie in unserem Fall eine Sequenz von aufeinanderfolgenden Bildern vorliegen, so stößt man in der Bildverarbeitung immer wieder auf den Begriff des **optischen Flusses**. Dieses Verfahren wird zur Analyse von Bewegungen eines Objekts verwendet. Es basiert auf den Veränderungen der Grauwerte der einzelnen Bilder. Somit gilt die Annahme, dass sich der Grauwert eines Pixels von einem Frame zum nächsten verschiebt, also

$$u(x,t) = u(x+\omega, t+1),$$

wobei $u(x,t)$ die Intensität des Pixels mit Koordinaten $x = (x_1, x_2)^T$ zur Zeit t und $\omega = (\omega_1, \omega_2)$ den Verschiebungsvektor darstellt. Es wird angenommen, dass die Intensität entlang einer Kurve $x(t)$ mit $\frac{dx}{dt} = v(x,t)$ konstant ist. Mit der multivariablen Kettenregel gilt dann (nach [DB])

$$0 = \frac{du}{dt} = \frac{\partial u}{\partial t} + \sum_{i=1}^{n} \frac{\partial u}{\partial x_i}\frac{dx_i}{dt} = u_t + \nabla u \cdot v. \tag{3.1}$$

Diese Gleichung wird auch als „optical flow constraint“ bezeichnet, was sich als ein lineares inverses Problem betrachten lässt. Unter einem inversen Problem versteht man im Allgemeinen ein Problem, bei dem von einer beobachteten Wirkung auf deren Ursache geschlossen werden muss (nach [Ri, S. vii]). Daher schreiben wir

(3.1) durch Tauschen der Positionen von v und ∇u um zu

$$(\nabla u)^T \cdot v = -u_t.$$

Mit den Definitionen von $g := -u_t$ und

$$\begin{aligned} A : L^p(\Omega)^2 &\to L^p(\Omega), \\ Av &= (\nabla u)^T \cdot v \end{aligned}$$

erfüllt die Gleichung das inverse Problem $Av = g$ für $p = 1$ oder $p = 2$. Näheres zu diesem Thema kann in [KA] und [BP] nachgelesen werden.

4 Vorverarbeitung

In diesem Kapitel werden wir die drei Schritte kennen lernen, die zur Vorverarbeitung der Rohdaten benutzt wurden. Diese sind neben dem Extrahieren des für uns wichtigen grünen Kanals die Entfernung von Rauschen sowie die Segmentierung der Zellwände zur Eingrenzung der möglichen JAIL-Koordinaten.

4.1 Extraktion des p20-Kanals

Wie schon in Kapitel 2 beschrieben, enthalten die Videos der JAIL zwei verschiedene Kanäle. Die Bildung von JAIL-Formationen wird in dem grünen Kanal deutlich. Dieser zeigt die Konzentration des p20-Proteins, welches mit dem EGFP markiert wurde.
Um für uns unbenötigte Informationen aus den Daten zu entfernen, müssen wir uns also den grünen Kanal aus dem Video extrahieren. Dazu haben wir das zur Analyse von biologischen Bild-Dateien frei verfügbare Programm Icy (`http://www.icy.bioimageanalysis.org/`) benutzt. Hiermit kann man sich die beiden Kanäle der Videos unabhängig voneinader anschauen oder sich einen bestimmten Kanal extrahieren. Weiterhin kann man Videos mit dem Matlab Im- und Exporter direkt in Form einer 3D-Matrix nach Matlab exportieren oder sich diese von Matlab nach Icy importieren. Auf diese Weise haben wir für den grünen Kanal des Videos die MAT-Datei mit dem Namen *p20_video.mat* erstellt.

4.2 Entrauschung

Im Folgenden halten wir uns - falls nicht anders erwähnt - an [SB].
Die Videos der JAIL wurden mit einem Konfokalmikroskop, also einem speziellen Lichtmikroskop erstellt. Die Rekonstruktion solcher Bilder ist in der Regel ein schlecht gestelltes inverses Problem. Schlecht gestellt bedeutet, dass bereits kleine Messfehler große Fehler in der Ursache nach sich ziehen (nach [Ri, S. vii]). Ziel der Rekonstruktion ist es, aus den gemessenen Daten eine Näherung des eigentlichen

Bildes zu berechnen. Folgende Gleichung können wir dazu aufstellen:

$$\bar{f} = \bar{K}\bar{u}.$$

Dabei sind $\bar{K}$ ein kompakter linearer Operator, $\bar{f}$ die exakten Daten und $\bar{u}$ das gewünschte exakte Bild. In der Praxis liegen jedoch nur verrauschte Versionen f und K von $\bar{f}$ und $\bar{K}$ vor. Dementsprechend suchen wir nach einer approximativen Lösung u von $\bar{u}$ mit

$$f = Ku.$$

K ist nun ein semi-diskreter Operator basierend auf $\bar{K}$, der u in den diskreten Raum $\mathbb{R}^N$ abbildet. Zur Schätzung von u wird in der Statistik üblicherweise die Maximum-a-posteriori-Methode (MAP) im Zusammenhang mit dem Satz von Bayes benutzt und geschieht durch eine Maximierung der a-posteriori-Wahrscheinlichkeitsdichte $p(u|f)$, der sogenannten Likelihood-Funktion. Der Satz von Bayes sagt Folgendes aus.

Satz 1. ***(Satz von Bayes** - nach [BK, S. 491]).*
Für zwei Ereignisse A und B gilt

$$p(A|B) = \frac{p(A) \cdot p(B|A)}{p(B)}$$

Beweis. Es gilt

$$p(A|B) = \frac{p(A \cap B)}{p(B)} \qquad \Leftrightarrow \qquad p(A \cap B) = p(B) \cdot p(A|B). \tag{4.1}$$

Ebenso gilt

$$p(A \cap B) = p(B \cap A) = p(A) \cdot p(B|A). \tag{4.2}$$

Aus (4.1) und (4.2) folgt

$$p(B) \cdot p(A|B) = p(A) \cdot p(B|A)$$

$$\Leftrightarrow \qquad p(A|B) = \frac{p(A) \cdot p(B|A)}{p(B)}$$

□

Bei Aufnahmen mit einem Fluoreszenzmikroskop entsteht aufgrund der statistisch verteilten Emission von Photonen Poissonrauschen, auch Schrotrauschen genannt. Es kann also jedes Pixel f_i unseres verrauschten Bildes f durch eine Poisson-verteilte Zufallsvariabe F_i mit Erwartungswert $(Ku)_i$ beschrieben werden. Daher betrachten wir die Poisson-Verteilung

$$p(f|u) = \prod_i \frac{(Ku)_i^{f_i}}{f_i!} e^{-(Ku)_i}. \tag{4.3}$$

Zur Anwendung des Satzes von Bayes benötigen wir weiterhin Informationen über die a-priori-Wahrscheinlichkeitsverteilung $p(u)$. Diese wird meistens als sogenannte Gibbs-Funktion gewählt (siehe [GG]), also

$$p(u) = e^{-\lambda R(u)}, \tag{4.4}$$

wobei λ ein positiver Parameter und R ein konvexes Funktional ist. Mit (4.3) und (4.4) gilt nach dem Satz von Bayes

$$p(u|f) = \prod_i \left(\frac{(Ku)_i^{f_i}}{f_i!} e^{-(Ku)_i} \right) e^{-\lambda R(u)} \frac{1}{p(f)}. \tag{4.5}$$

Wir wenden nun die MAP-Methode an und berechnen durch Maximierung von 4.5 das optimale $\hat{u}$. Aufgrund der Monotonie des Logarithmus können wir anstelle der Likelihood-Funktion auch die Log-Likelihood-Funktion betrachten.

$$\begin{aligned}
\hat{u} &= \operatorname*{arg\,max}_{u \in L^1_+(\Omega)} \{p(u|f)\} \\
&= \operatorname*{arg\,max}_{u \in L^1_+(\Omega)} \{\log p(u|f)\} \\
&= \operatorname*{arg\,max}_{u \in L^1_+(\Omega)} \left\{ \log \prod_i \left(\frac{(Ku)_i^{f_i}}{f_i!} e^{-(Ku)_i} \right) e^{-\lambda R(u)} \frac{1}{p(f)} \right\} \\
&\overset{(*)}{=} \operatorname*{arg\,max}_{u \in L^1_+(\Omega)} \left\{ \log \prod_i \left((Ku)_i^{f_i} e^{-(Ku)_i} \right) e^{-\lambda R(u)} \right\} \\
&= \operatorname*{arg\,min}_{u \in L^1_+(\Omega)} \left\{ -\log \prod_i \left((Ku)_i^{f_i} e^{-(Ku)_i} \right) e^{-\lambda R(u)} \right\} \\
&= \operatorname*{arg\,min}_{u \in L^1_+(\Omega)} \left\{ \sum_i \left((Ku)_i - f_i \log(Ku)_i \right) + \lambda R(u) \right\}
\end{aligned}$$

Hierbei ist $L^1_+(\Omega) = \{u \in L^1(\Omega) : u \geq 0 \text{ fast überall}\}$. Weiterhin haben wir in (*)

die beiden von u unabhängigen Ausdrücke $f_i!$ und $\frac{1}{p(f)}$ weggelassen. Als kontinuierliche Darstellung erhalten wir somit

$$\hat{u} = \underset{u \in L^1_+(\Omega)}{\arg\min} \left\{ \int_\Omega \Big(Ku - f \log(Ku) \Big) \, \mathrm{d}\mu + \lambda R(u) \right\}.$$

Der Datenterm, den wir erhalten, entspricht bis auf einige von u unabhängige Ausdrücke der berühmten Kullback-Leibler-Divergenz

$$\mathrm{D}_{\mathrm{KL}}(f, Ku) = \int_\Omega \left(Ku - f + f \log \left(\frac{f}{Ku} \right) \right) \mathrm{d}\mu.$$

Als Regularisierungsterm R nehmen wir die Totale Variation (TV) von u, auch als $\int_\Omega |\nabla u|$ bezeichnet, die durch

$$\int_\Omega |\nabla u| \, \mathrm{d}\mu = \sup \left\{ \int_\Omega u \operatorname{div} z \, \mathrm{d}x : z \in C_0^\infty(\Omega; \mathbb{R}^n), |z(x)| \leq 1 \, \forall x \in \Omega \right\}$$

definiert ist. Diese garantiert neben einer Unterdrückung des Rauschens einen hohen Erhalt an wichtigen Informationen im Bild, wie zum Beispiel scharfen Kanten oder homogenen Flächen (nach [CC]).
Wir erhalten somit das zu lösende Minimierungsproblem mit einem Kullback-Leibler-Datenterm und der totalen Variation als Regularisierer

$$\min_u \left\{ \int_\Omega \Big(Ku - f \log(Ku) \Big) \, \mathrm{d}\mu + \lambda \int_\Omega |\nabla u| \, \mathrm{d}\mu \right\}. \tag{4.6}$$

Im Folgenden benötigen wir einige Begriffe aus der konvexen Optimierung, die wir zunächst definieren wollen.

Definition 3. *(**konvexe Funktion** - nach [Wi, S. 32])*
Sei $C \subset \mathbb{R}^n$ eine konvexe Menge. Dann ist $f : C \to \mathbb{R} \cup \{\infty\}$ eine konvexe Funktion, falls gilt

$$f(\Theta x + (1 - \Theta) y) \leq \Theta f(x) + (1 - \Theta) f(y) \qquad \forall x, y \in C, \Theta \in [0, 1].$$

Definition 4. *(**zulässige Funktion** - nach [Ro, S. 24])*
Eine konvexe Funktion f heißt zulässig, falls

- *$f(x) < \infty$ für mindestens ein x und*
- *$-\infty < f(x)$ für alle x*

gilt.

Definition 5. *(**unterhalbstetig** - nach [Wi, S. 36])*
$f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ heißt unterhalbstetig, falls gilt

$$\liminf_{y \to x} f(y) \geq f(x) \qquad \forall x \in \mathbb{R}^n.$$

Definition 6. *(**konjugierte Funktion** - nach [Ro, S. 104])*
Die konjugierte Funktion zu einer Funktion f ist für $y \in \mathbb{R}^n$ definiert als

$$f^*(y) = \sup_{x \in \mathbb{R}^n} \{\langle y, x \rangle - f(x)\}.$$

Zur Lösung von (4.6) benutzen wir einen primal-dualen Algorithmus, der 2010 von Antonin Chambolle und Thomas Pock in [CP] vorgestellt wurde. Er dient der Lösung eines primal-dualen Minimierungsproblems in Sattelpunktform

$$\min_{x \in X} \max_{y \in Y} \langle \hat{K}x, y \rangle + G(x) - F^*(y). \tag{4.7}$$

Dabei sind X und Y zwei endlich-dimensionale reele Vektorräume mit Skalarprodukt $\langle \cdot, \cdot \rangle$ und Norm $\| \cdot \| = \langle \cdot, \cdot \rangle^{\frac{1}{2}}$. Die Abbildung $\hat{K}$ ist ein stetiger linearer Operator mit induzierter Norm

$$\|\hat{K}\| = \max\{\|\hat{K}x\| : x \in X \text{ mit } \|x\| \leq 1\}. \tag{4.8}$$

Weiterhin sind $G : X \to [0, +\infty)$ und $F^* : Y \to [0, +\infty)$ zulässig, konvex und unterhalbstetig, wobei F^* die konjugierte Funktion einer konvexen und unterhalbstetigen Funktion F ist.
Für den Algorithmus von Chambolle und Pock benötigen wir zunächst die Definition des Proximal-Operators.

Definition 7. *(**Proximal-Operator** - nach [CP, S. 3]).*
Der Proximal-Operator von F ist definiert durch

$$x = \text{prox}_{\tau F}(y) = (I + \tau \partial F)^{-1}(y) = \arg\min_x \left\{ \frac{\|x - y\|^2}{2\tau} + F(x) \right\}.$$

Außerdem beinhaltet der Algorithmus noch einen Relaxationsschritt.

Definition 8. *(**Relaxation**).*
Als Relaxation bezeichnet man den gewichteten Durchschnitt zwischen der neuen Variable x^{n+1} und der vorherigen Variable x^n:

$$\bar{x}^{n+1} = \omega x^{n+1} + (1 - \omega) x^n.$$

Man spricht von einer Unterrelaxation, falls $0 \leq \omega < 1$ bzw. einer Überrelaxation, falls $1 \leq \omega \leq 2$.

Folgende Schritte beinhaltet das Verfahren (nach [CP, S. 4]).

Algorithmus von Chambolle und Pock:
Initialisierung: Wähle $\sigma, \tau > 0, \theta \in [0,1], (x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^m$ und setze $\bar{x}^0 = x^0$.
Iterationen für $n \geq 0$: Aktualisiere x^n, y^n und $\bar{x}^n$ wie folgt:

$$\begin{cases} y^{n+1} = (I + \sigma \nabla F^*)^{-1}(y^n + \sigma \hat{K} \bar{x}^n) \\ x^{n+1} = (I + \tau \nabla G)^{-1}(x^n - \tau \hat{K}^T y^{n+1}) \\ \bar{x}^{n+1} = x^{n+1} + \theta(x^{n+1} - x^n) \end{cases}$$

Um den Algorithmus anwenden zu können, müssen wir das Minimierungsproblem (4.6) zunächst in ein Problem in Sattelpunktform (4.7) umwandeln. Dazu benötigen wir die in der Optimierung häufig verwendete erweiterte charakteristische Funktion einer Menge $S \subseteq \mathbb{R}^d$ (nach [CP, S. 24])

$$\chi_S(x) = \begin{cases} 0, & \text{falls } x \in S, \\ +\infty & \text{falls } x \notin S \end{cases} \tag{4.9}$$

Da es sich in unseren Videos um eine Abfolge von t diskreten Bildern der Größe $(M \times N)$ handelt, setzen wir voraus, dass $\Omega = (M \times N \times t) \subset \mathbb{R}^3$ gilt. Damit können wir nun wie folgt die Sattelpunktform von (4.6) bestimmen. Hierbei wählen wir λ als Gewichtung für den Kullback-Leibler-Datenterm und für den Operator K wie bei der Entrauschung üblich die Identität I. Dann gilt

$$\begin{aligned}
&\min_x \left\{ \lambda \int_\Omega (x - f \log x)\, \mathrm{d}x + \int_\Omega |\nabla x|\, \mathrm{d}x \right\} \\
=&\min_x \Bigg\{ \lambda \int_\Omega (x - f \log x)\, \mathrm{d}x \ldots \\
&\qquad \ldots + \max_y \left\{ \int_\Omega (x \operatorname{div} y)\, \mathrm{d}z : |y(z)| \leq 1 \, \forall z \in \Omega, y \in C_0^\infty(\Omega, \mathbb{R}^n) \right\} \Bigg\} \\
=&\min_x \max_y \left\{ \lambda \int_\Omega (x - f \log x)\, \mathrm{d}x + \int_\Omega (x \operatorname{div} y)\, \mathrm{d}z \right\}, \text{ s.d. } |y(z)| \leq 1 \, \forall z \in \Omega, x > 0 \\
\overset{(4.9)}{=}&\min_x \max_y \left\{ \lambda \int_\Omega (x - f \log x)\, \mathrm{d}x + \int_\Omega (x \operatorname{div} y)\, \mathrm{d}z + \chi_{(0,\infty)}(x) + \chi_Y(y) \right\} \\
&\qquad \text{mit } Y := \{z : |z| \leq 1\} \\
\overset{(*)}{=}&\min_x \max_y \Big\{ \underbrace{\lambda \int_\Omega (x - f \log x)\, \mathrm{d}x + \chi_{(0,\infty)}(x)}_{:= G(x)} \underbrace{- \int_\Omega (\nabla x \cdot y)\, \mathrm{d}z}_{:= \langle \hat{K} x, y \rangle} + \underbrace{\chi_Y(y)}_{:= -F^*(y)} \Big\}.
\end{aligned}$$

(*) folgt hierbei aus der mehrdimensionalen partiellen Integration mit verschwindendem Randterm, da $y \in C_0^\infty(\Omega, \mathbb{R}^n)$.
Als nächstes benötigen wir die Proximaloperatoren für F^* und G. Da F^* lediglich die charakteristische Funktion der Menge Y beschreibt, gilt

$$\text{prox}_{\sigma F^*}(y) = \frac{y}{\max(1, |y|)}.$$

Die Berechnung von $\text{prox}_{\tau G}(y)$ gestaltet sich dagegen ein wenig komplizierter. Es gilt nach Definition 7

$$\begin{aligned} \text{prox}_{\tau G}(y) &= \arg\min_v \left\{ \frac{\|v-y\|^2}{2\tau} + \lambda \int_\Omega (v - f \log v)\, \mathrm{d}v + \chi_{(0,\infty)}(v) \right\} \qquad (4.10) \\ &= \arg\min_v \left\{ \frac{\int_\Omega (v^2 - 2vy + y^2)\, \mathrm{d}v}{2\tau} + \lambda \int_\Omega (v - f \log v)\, \mathrm{d}v + \chi_{(0,\infty)}(v) \right\}. \end{aligned}$$

Die charakteristische Funktion χ wird zunächst nicht berücksichtigt. Stattdessen wird sie nach Berechnung des Proximaloperators über eine Maximumfunktion der Form $\max(\text{prox}_{\tau G}, 0)$ realisiert.
Weiterhin führen wir eine komponentenweise Berechnung durch. Aus (4.10) folgt somit

$$\begin{aligned} & \frac{v_i - y_i}{\tau} + \lambda - \frac{\lambda f_i}{v_i} = 0 \\ \Leftrightarrow \quad & \frac{v_i}{\tau} - \frac{\lambda f_i}{v_i} = \frac{y_i}{\tau} - \lambda \\ \Leftrightarrow \quad & \frac{v_i^2 - \lambda\tau f_i}{\tau v_i} = \frac{y_i}{\tau} - \lambda \\ \Leftrightarrow \quad & v_i^2 - \lambda\tau f_i = \left(\frac{y_i}{\tau} - \lambda\right) \tau v_i \\ \Leftrightarrow \quad & v_i^2 - \left(\frac{y_i}{\tau} - \lambda\right) \tau v_i = \lambda\tau f_i \end{aligned}$$

Mit Verwendung der p-$q-$Formel folgt

$$\begin{aligned} \text{prox}_{\tau G}(y) = v_i &= \frac{1}{2}\left(\frac{y_i}{\tau} - \lambda\right)\tau \pm \sqrt{\frac{1}{4}\left(\frac{y_i}{\tau} - \lambda\right)^2 \tau^2 + \lambda\tau f_i} \\ &= \frac{1}{2}(y_i - \lambda\tau) \pm \sqrt{(\frac{1}{2}y_i - \frac{1}{2}\lambda\tau)^2 + \lambda\tau f_i}. \end{aligned}$$

Als letzten Schritt benötigen wir nun noch eine Diskretisierung des Gradienten ∇u. Dazu sei $X = \mathbb{R}^{M \times N}$ ein endlich dimensionaler Vektorraum mit Standardskalarprodukt

$$\langle u, v \rangle_X = \sum_{i,j} u_{i,j} v_{i,j} \ , \qquad u, v \in X.$$

Der Gradient ∇u ist ein Vektor im Vektorraum $Y = X \times X$. Zur Diskretisierung von $\nabla : X \to Y$ benutzen wir finite Differenzen mit Schrittweite $h = 1$ und Neumann-Randbedingungen

$$(\nabla u)_{i,j} = \begin{pmatrix} (\nabla u)^1_{i,j} \\ (\nabla u)^2_{i,j} \end{pmatrix},$$

wobei

$$(\nabla u)^1_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j} & \text{falls } i < M \\ 0 & \text{falls } i = M \end{cases}, \quad (\nabla u)^2_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j} & \text{falls } j < N \\ 0 & \text{falls } j = N \end{cases}$$

(nach [CP, S. 21]).

4.3 Segmentierung

Es ist bekannt, dass JAIL nur in unmittelbarer Nähe der Zellwände auftreten. Daher ist es sinnvoll, eine Umgebung der Zellwände zu segmentieren, deren Radius der Größe eines JAIL entspricht. Dadurch lassen sich falsch positive JAIL innerhalb einer Zelle ausschließen. Der hierzu benötigte Verlauf der Zellwände wurde mithilfe des „CellBorderTrackers“ (siehe [ST]) erstellt und uns zur Verfügung gestellt. Abbildung 4.1 zeigt ein Beispielbild der Verwendung des CellBorderTrackers, der nach [ST] eine Genauigkeit von 90% aufweist.

Es hat sich herausgestellt, dass wir für eine bessere Erkennung der richtigen JAIL auf drei unterschiedlich große Segmentierungen der Zellwände zurückgreifen müssen. Die grundlegende Segmentierung umfasst einen Umgebungsradius von 30 Pixeln. Diese Umgebungsgröße reicht aus, um JAIL jeglicher Größe einzuschließen. Sucht man nun nicht im gesamten Bild nach möglichen JAIL, sondern nur in dem segmentierten Bild, so lassen sich Strukturen im Inneren einer Zelle direkt ausschließen.
Abbildung 4.2 zeigt diese grundlegende Segmentierung am Beispiel eines Frames mit einem JAIL unten links im Bild. Da wir hier einen sehr deutlichen JAIL vorliegen haben, werden wir von nun an sämtliche Abbildungen zur Veranschaulichung der Verfahren an diesem Beispiel zeigen.

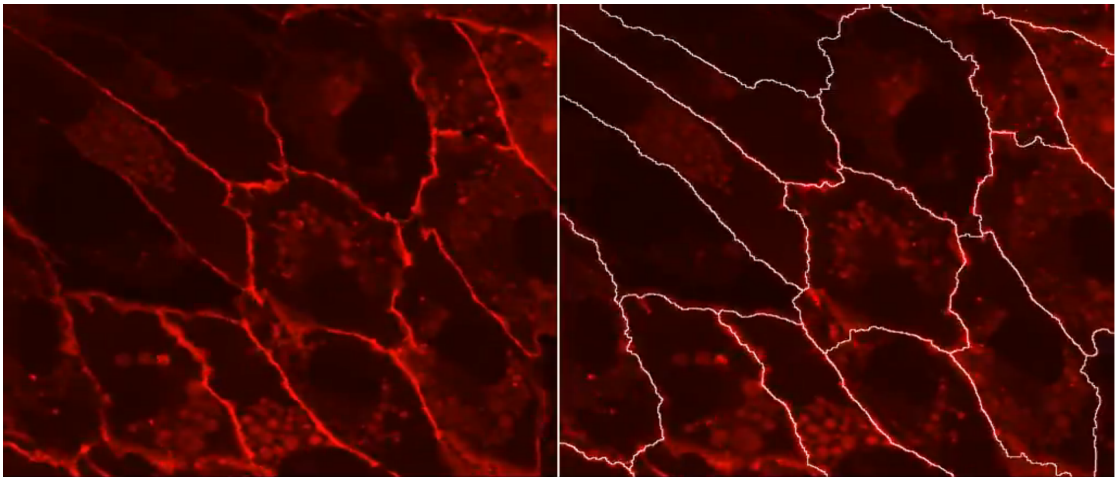

Abbildung 4.1: Endothelzellen (links) und gekennzeichnete Zellgrenzen mithilfe des CellBorderTrackers (rechts)

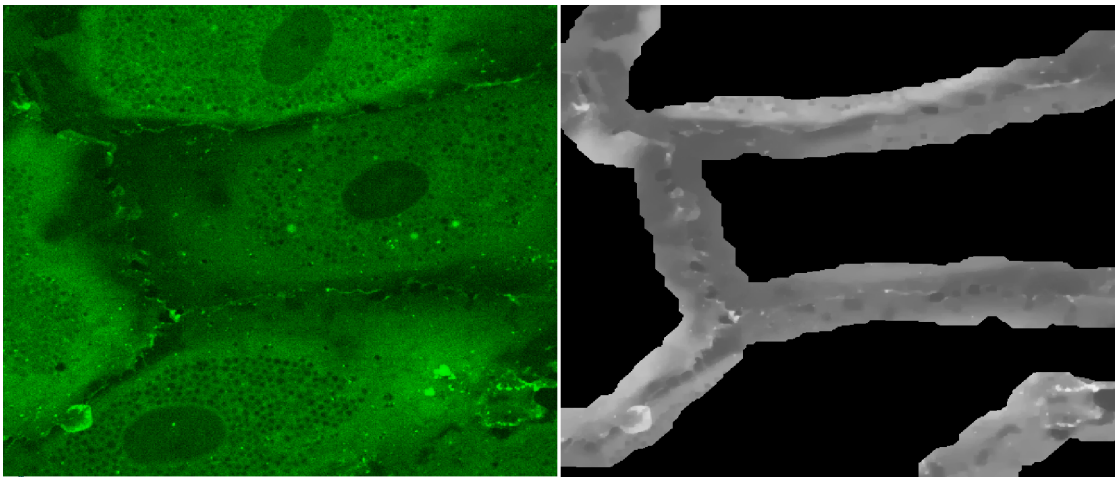

Abbildung 4.2: Originalbild (links) und segmentiertes, entrauschtes Bild (rechts)

In Kapitel 5.1.2 wird erklärt, dass zur Faltung eines Bildes mit einem Gauss-Kreis zunächst das Kantenbild erstellt werden muss. Würde man dieses nun von dem obigen segmentierten Bild erstellen, so würden die Kanten der Umgebung als Kante mit sehr hohem Wert gezählt. Um dies zu verhindern, benutzen wir daher eine weitere Segmentierung, deren Umgebungsradius um fünf Pixel höher als zuvor eingestellt ist. Hiervon wird dann zunächst das Kantenbild erstellt, um daraufhin mithilfe der ersten Segmentierung die falschen Kanten, also die äußeren fünf Pixel der Umgebung zu entfernen. Abbildung 4.3 verdeutlicht dieses Vorgehen.

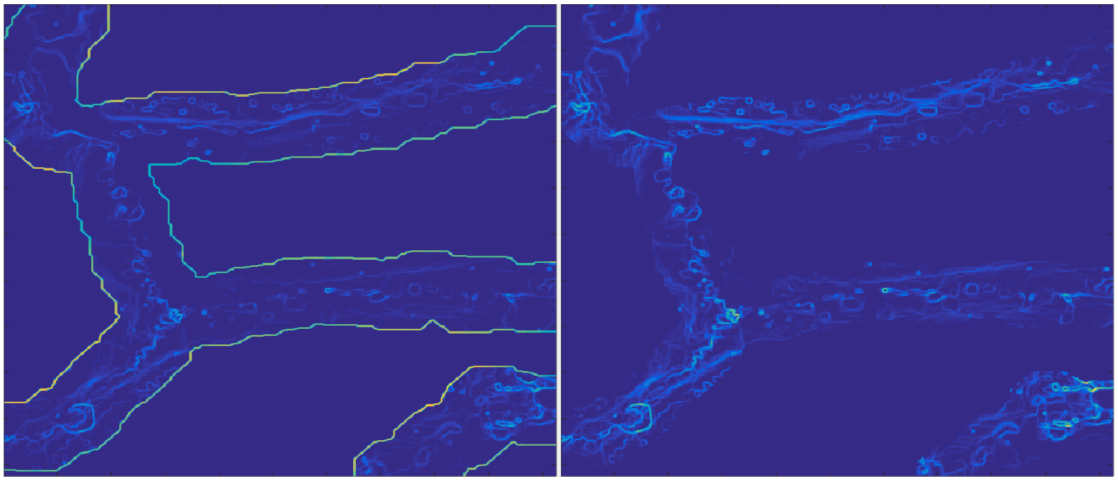

Abbildung 4.3: Große Segmentierung mit falschen Kanten (links) und Überreste ohne falschen Kanten nach Vergleich mit ursprünglicher Segmentierung (rechts)

Die dritte und kleinste Segmentierung wird benutzt, um sicherzustellen, dass nur diejenigen JAIL in Betracht gezogen werden, deren Mittelpunkte sich in unmittelbarer Nähe zur Zellwand befinden. Hierfür nehmen wir einen Umgebungsradius von fünf Pixeln. Mithilfe dieser Segmentierung löschen wir alle Pixel in dem gefalteten Bild (siehe Kap. 5.1.2), die nicht in der Umgebung liegen. Werden also JAIL mit einem Mittelpunkt außerhalb dieser Umgebung gefunden, so werden diese als falsch positive JAIL gewertet. In Abbildung 4.4 sehen wir den Verlauf der Zellwände und das in Kapitel 5.1.2 vorgestellte Faltungsbild mit $r = 17$, segmentiert mit der 5-Pixel-Umgebung für die Zentren der JAIL. Das zugrundeliegende Frame ist das gleiche wie in Abbildung 4.3. Man erkennt dabei im Faltungsbild anhand der gelben Pixel unten links im Bild das Zentrum des deutlich sichtbaren JAIL.

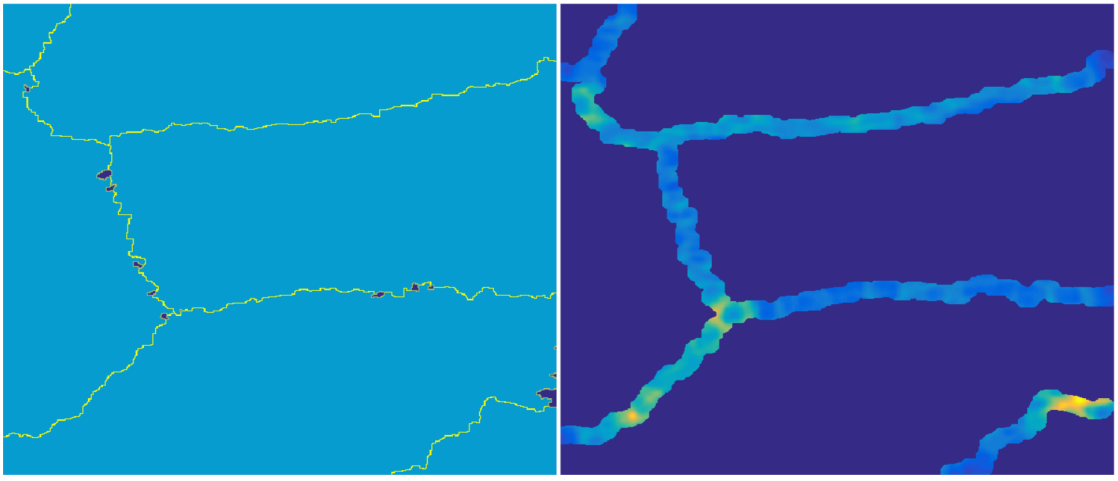

Abbildung 4.4: Verlauf der Zellwände (links) und Faltungsbild mit zulässiger Umgebung für JAIL-Zentren (rechts)

5 Verfahren zur JAIL-Detektion

In diesem Kapitel werden wir die Verfahren kennen lernen, die in dieser Arbeit zur Detektion der JAIL getestet wurden. Diese sind die Hough-Transformation sowie die Faltung mit Gauss-Kreisen und -Halbkreisen zur Erkennung von kreisförmigen Strukturen, das Optical-Flow-Verfahren zur Bewegungsanalyse, das Aufsummieren von Intensitäten, die Benutzung von Zusammenhangskomponenten sowie die Akkumulation von Faltungswerten.

5.1 Detektion kreisförmiger Strukturen

Da ein JAIL stets eine kreisförmige Struktur beschreibt, spielt die Detektion solcher Strukturen eine wesentliche Rolle in dieser Arbeit. Sie wird als erstes Verfahren zur JAIL-Detektion benutzt und bildet damit eine grundlegende Liste der potenziellen JAIL. Aus dieser Liste werden dann mit den später erklärten Verfahren möglichst viele falsch positive JAIL gelöscht. Da bei dem Vergleich der JAIL-Liste mit den anderen Verfahren also keine neuen potenziellen JAIL mehr hinzukommen, wählen wir die Parameter und Thresholds bei dieser Detektion so, dass zwar auch falsche JAIL erkannt werden, dafür aber möglichst alle richtigen JAIL mit dabei sind.

5.1.1 Die Hough-Transformation

In diesem Abschnitt beziehen wir uns auf [Rh].
In Abschnitt 3.2 haben wir bereits die Hough-Transformation zur Detektion parametrisierbarer Strukturen am Beispiel von Geraden kennen gelernt. Da zur Detektion der JAIL nicht nach Geraden, sondern nach kreisförmigen Strukturen gesucht werden muss, ist es notwendig die Hough-Transformation für Kreise anzuwenden. Ein Kreis mit Mittelpunkt (c_1, c_2) und Radius r kann als Koordinatengleichung mit

$$(x - c_1)^2 + (y - c_2)^2 = r^2$$

oder in Parameterdarstellung mit

$$\begin{aligned} x &= c_1 + r\cos(\varphi), \\ y &= c_2 + r\sin(\varphi) \end{aligned} \tag{5.1}$$

für $0 \leq \varphi < 2\pi$ dargestellt werden. Somit führt jeder Punkt (x, y) auf einem möglichen Kreis mit Mittelpunkt (c_1, c_2) und bekanntem Radius r im Bildraum zu einem Kreis mit Mittelpunkt (x, y) und Radius r im Parameterraum (siehe Abbildung 5.1).

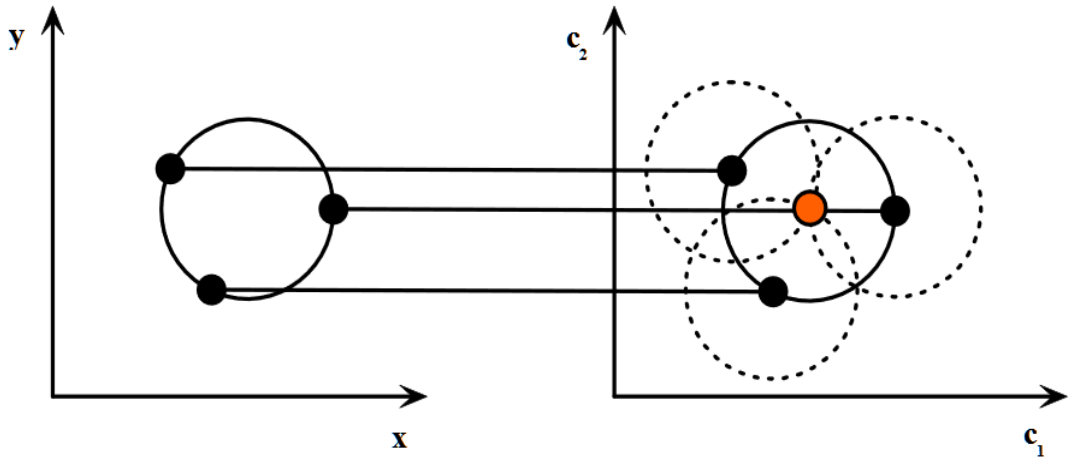

Abbildung 5.1: Darstellung der Hough-Transformation für Kreise mit bekanntem Radius. Jeder Punkt im Bildraum (links) generiert einen Kreis im Parameterraum. Alle diese Kreise schneiden sich in dem Mittelpunkt des Kreises im Bildraum (aus [Rh, S. 2])

In unserem Falle eines unbekannten Radius wird ein dreidimensionaler Raum mit dem Radius als dritter Dimension gebildet. Für die möglichen Radien zwischen 5 und 30 Pixeln bildet also jeder Kantenpunkt (x, y) im Bildraum einen Kegel im Parameterraum (x, y, r), der sich aus mehreren Kreisen mit verschiedenen Radien zusammensetzt. Liegt nun ein Kreis mit Mittelpunkt (c_1, c_2) und Radius r_0 im Bildraum vor, so schneiden sich alle diese Kegel im Punkt (c_1, c_2, r_0) des Parameterraums. An dieser Stelle liegt dementsprechend der höchste Wert vor. Um die passenden Objekte im Bildraum zu finden, sucht man also nach lokalen Maxima im Parameterraum.
Bei der von uns benutzten und in Matlab vordefinierten Funktion *imfindcircles.m* wird das Kantenbild zunächst mit dem Sobel-Operator erstellt, der aufgrund von doppelter Gewichtung der zentralen Filterzeile bzw. -spalte neben der Kantendetektion eine zusätzliche Glättung orthogonal zur Ableitungsrichtung bewirkt. Dies

geschieht mithilfe von Faltungen des Bildes und der jeweiligen Operatoren S_x und S_y in x- und y-Richtung. Sei also F unser Ursprungsbild, dann gilt (nach [BB, S. 122]) für die beiden Gradienten

$$\begin{aligned} G_x &= S_x * F = \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}}_{\substack{\text{Sobel-Operator} \\ \text{in } x\text{-Richtung}}} * F, \\ G_y &= S_y * F = \underbrace{\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}}_{\substack{\text{Sobel-Operator} \\ \text{in } y\text{-Richtung}}} * F. \end{aligned} \tag{5.2}$$

Durch Zusammenbringen der beiden Ergebnisse mittels

$$G = \sqrt{G_x^2 + G_y^2}$$

entsteht dann das richtungsunabhängige Kantenbild. Zusätzlich kann man der Funktion einen Threshold „EdgeThreshold“ mitgeben. Dieser Wert zwischen 0 und 1 gibt an, welche Art von Kanten gefunden werden sollen. Ein hoher Wert bedeutet, dass nur starke Kanten gefunden werden, während ein niedriger Wert auch schwache Kanten zulässt. Wird der Funktion kein Wert mitgegeben, so wird er automatisch mithilfe der Funktion *graythresh.m* ermittelt.
Für die in dem entstandenen Bild enthaltenen Kantenpunkte wird anschließend wie oben beschrieben der Parameterraum erstellt. Der Unterschied hierbei liegt darin, dass in der Matlab-Funktion für unbekannte Radien kein dreidimensionaler, sondern ein zweidimensionaler Parameterraum erstellt wird. Hier werden dann die Werte für alle Radien akkumuliert, sodass ein weiterer Schritt zur Schätzung der jeweiligen Radien nötig ist. Hierfür gibt es zwei Methoden, zum einen das „Two-Stage-Verfahren“ und zum anderen das „Phase-Coding-Verfahren“ (näheres zu dieser nachträglichen Schätzung der Radien kann in [AK, S.795-803], [YP, S.71-77] und [Da, Kap. 10] nachgelesen werden). Trotz dieses zusätzlich nötigen Schrittes ist das gesamte Verfahren auf diese Weise weniger speicher- und rechenintensiv als bei der Nutzung eines dreidimensionalen Parameterraumes (siehe Matlab-Dokumentation zu *imfindcircles.m*).

Neben den schon beschriebenen optionalen Parametern gibt es weitere Werte, die man der Funktion mitgeben kann. So wäre da beispielsweise die Sensitivität „Sensitivity“ zur Kreisdetektion. Dieser Wert liegt ebenfalls zwischen 0 und 1 und

wird, falls nicht explizit mitgegeben, standardmäßig auf 0.85 gesetzt. Ein höherer Wert liefert mehrere Kreise, wobei dadurch das Risiko einer höheren Fehlerrate besteht. Außerdem gibt es den Parameter „ObjectPolarity“, den man der Funktion als „bright“ oder „dark“ mitgeben kann. So werden nur die Kreise gefunden, die heller beziehungsweise dunkler als der Hintergrund sind. Aufgrund der hohen Intensitäten der JAIL haben wir uns hier für die Suche nach hellen Kreise entschieden.

Wir haben den Code aus Algorithmus 1 unter Nutzung der Matlab-Funktion *imfindcircles.m* verwendet:

Algorithm 1 Hough-Transformation für Kreise, *Find_Circles.m*

Input:

$Video$	▷ entrauschtes Video
RR	▷ Radius-Range, bei uns 5-30
KT	▷ Kanten-Threshold, bei uns 0.01
$Methode$	▷ Methode zur Radius-Schätzung, bei uns „TwoStage“
$Polarität$	▷ Verhältnis Objekt zu Hintergrund, bei uns „bright“

Output:

$Zentren$	▷ Mittelpunkte der gefundenen Kreise
$Radien$	▷ Radien der gefundenen Kreise

1: $Video$=$Video$/max($Video$) ▷ Normierung des Videos

2: **for all** Frames of $Video$ **do**

3: $[Zentren, Radien]$ = *imfindcircles*(Frame, RR, ..., $Methode$, KT, $Polarität$)

4: **end for**

5: **return** $[Zentren, Radien]$

Die hohe Variabilität an Parametern sowie die zum Teil fehlende Kenntnis über den genauen Ablauf der Hough-Transformation in Matlab haben uns dazu motiviert, ein eigenes Verfahren zur Detektion von kreisförmigen Strukturen zu programmieren. Dieses werden wir in den folgenden Abschnitten 5.1.2 und 5.1.3 erläutern.

5.1.2 Faltung mit Gauss-Kreisen

Anstelle der Hough-Transformation kann man auch mit einer geeigneten Faltung des Kantenbildes nach kreisförmigen Strukturen suchen. Als Faltungskerne dienen uns dabei Kreise verschiedener Größe, die mit der Gauss-Funktion geglättet werden. Die Größe der Kreise, nach denen wir suchen, liegt auch hier bei einem Radius zwischen 5 und 30 Pixeln. Damit sind die Größen aller möglichen JAIL abgedeckt. Wir benötigen zunächst einen Faltungskern f_G. Damit der geglättete Kreis komplett in dem Kern enthalten ist, wird für f_G bei jeweiligem Radius r eine Größe von $(4r + 1) \times (4r + 1)$ festgelegt. In diesen Kern wird ein Kreis mit Radius r „gezeichnet“. Es werden also Einsen in diejenigen Pixel des Kerns geschrieben, die dann zusammen einen möglichst runden Kreis ergeben (aufgrund der Diskretheit unserer Bilder und des Kerns kann kein vollständig perfekter Kreis entstehen). Um dem Kreis eine gewisse Grundbreite zu geben, wird er anschließend auf eine Breite von 5 Pixeln $(r - 2, \ldots, r + 2)$ erweitert.
Die Glättung der Kreise geschieht daraufhin mit einem Gaussfilter der Größe (7×7). Da ein kleiner JAIL der Form eines Kreises näher kommt als ein großer JAIL (vgl. Abbildung 5.2), müssen wir die Standardabweichung für den Gaussfilter an den Radius anpassen. Als geeignetste Anpassung hat sich hierfür $\sigma = \sqrt{r}$ bewährt.

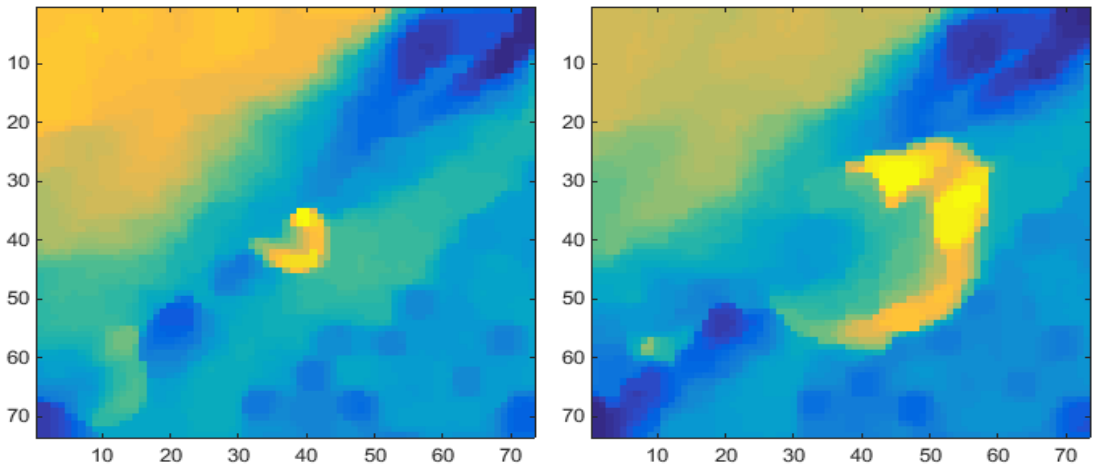

Abbildung 5.2: JAIL zur Zeit seines Auftretens (links) und kurz vor dem Verschwinden (rechts). Je mehr ein JAIL wächst, desto mehr verliert er seine kreisförmige Struktur.

Damit Kreise mit großem Radius nicht automatisch einen höheren Faltungswert erzielen als Kreise mit kleinerem Radius, wird jeder Gauss-Kreis noch mittels $\frac{s}{\text{sum}(s(:))}$ normiert. Abbildung 5.3 zeigt zwei Gauss-Kreise mit verschiedenen Radien.

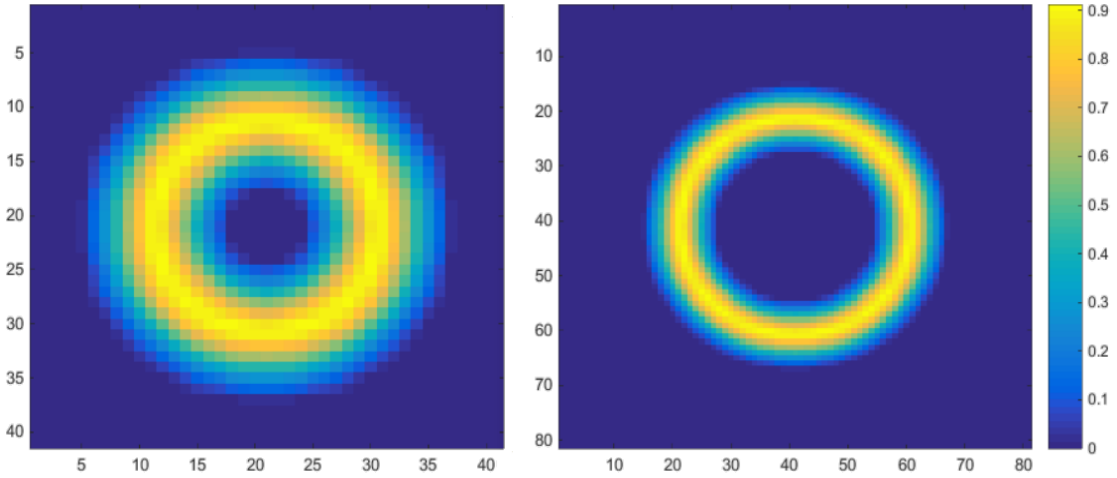

Abbildung 5.3: Kreise mit Radius 10 (links) bzw. 20 (rechts), geglättet mit Gaussfilter der Größe (7×7), Standardabweichungen $\sigma = \sqrt{10}$ bzw. $\sigma = \sqrt{20}$.

Als nächstes benötigen wir das Kantenbild f_K. Hierzu benutzen wir den „Vorwärts-Differenzen-Operator“, der jeden Pixel mit dem jeweils nächsten benachbarten Pixel in x- und y-Richtung vergleicht. Für die Ableitungen in x- und y-Richtung gilt demnach

$$\frac{du}{dx} = u(x+1) - u(x) \qquad \text{bzw.} \qquad \frac{du}{dy} = u(y+1) - u(y). \tag{5.3}$$

Das Kantenbild f_K wird dann durch die Berechnung der Norm der beiden Ableitungen festgelegt, also

$$f_K = \sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{du}{dy}\right)^2}.$$

Nun erfolgt die Faltung des Kantenbildes mit dem Gauss-Kreis. Zur Visualisierung dieser Faltung mit einem Gauss-Kreis passender Größe ist in Abbildung 5.4 der Ausschnitt aus Kanten- und Faltungsbild eines deutlich zu erkennenden JAIL dargestellt.

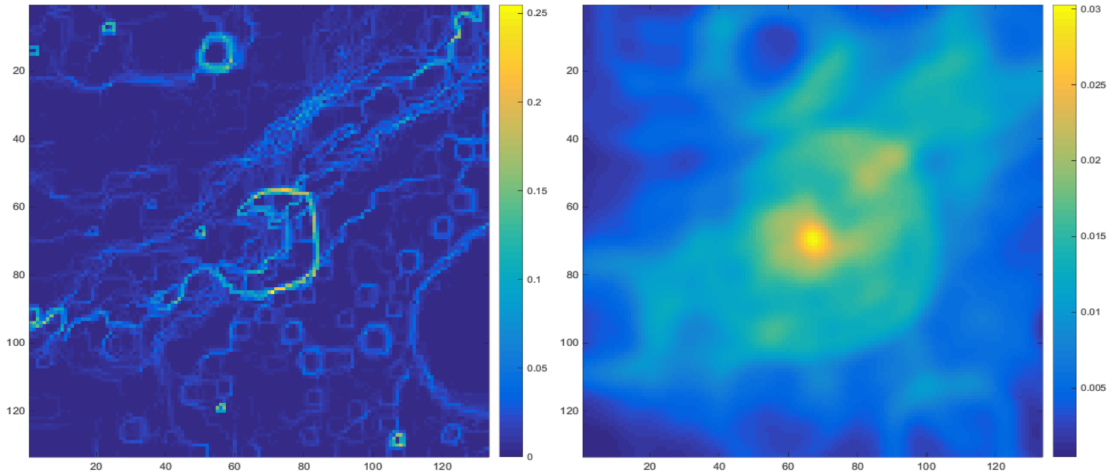

Abbildung 5.4: Ausschnitt des Kantenbildes mit dem Vorwärts-Differenzen-Operator (vgl. (5.3)) eines deutlichen JAIL (links) und Faltungsbild durch Gauss-Kreis mit passendem Radius 17 (rechts).
Die hohen Werte im Faltungsbild deuten auf den Mittelpunkt eines Kreises im Kantenbild hin.

Damit nicht alle Koordinaten mit einem hohen Faltungswert einem möglichen Kreismittelpunkt zugeordnet werden, teilen wir das Faltungsbild in Raster der Größe (40×40) ein. In diesen Rastern wird dann nur der jeweils maximale Wert behalten, alle anderen Pixel bekommen den Wert 0 zugewiesen. Liegt der Wert eines Pixels der Faltung nun über einem den Faltungswerten angepassten Threshold, so wird er zusammen mit seinen Koordinaten und dem Radius, nach dem aktuell gesucht wird, zu den jeweiligen Listen der bereits gefundenen potenziellen JAIL hinzugefügt.
Im letzten Schritt werden die Werte aller gefunden Kreise in unmittelbarer Nähe zueinander miteinander verglichen. Dies ist notwendig, da für einen JAIL mit Radius r nicht nur die Faltung mit dem genau passenden Gauss-Kreis einen hohen Wert liefert, sondern auch die mit Gauss-Kreisen, die einen etwas kleineren oder größeren Radius haben. Deshalb vergleichen wir alle Kreise miteinander, deren x- und y-Koordinaten nicht weiter als 20 Pixel auseinander liegen. Auch hier behalten wir nur den Kreis mit dem größten Faltungswert. Alle restlichen Kreise werden zusammen mit ihren jeweiligen Werten und Radien aus der Liste gelöscht.
Für die Umsetzung der Suche nach Kreisen in Matlab haben wir den Code aus Algorithmus 2 verwendet.

Algorithm 2 Faltung mit Gauss-Kreisen, *gauss_kreise.m*

Input:

$Video$	▷ entrauschtes Video
r	▷ Radius, nach dem gesucht werden soll
$Seg1$	▷ Segmentierung der Zellwände im Kantenbild (Abb. 4.3)
$Seg2$	▷ Segmentierung der Zellwände für die Kreis-Zentren (Abb. 4.4)
RG	▷ Raster-Größe, bei uns 40
FT	▷ Threshold für den Faltungswert

Output:

$Zentren$	▷ Mittelpunkte der gefundenen Kreise
$Radien$	▷ Radien der gefundenen Kreise
$Werte$	▷ Faltungswerte der gefundenen Kreise

1: $KG = 4 * r + 1$ ▷ Größe des Faltungskerns
2: $Kern =$ Kreis in $(KG \times KG)$-Matrix ▷ wie im Text beschrieben
3: Faltung von $Kern$ mit Gauss-Filter ▷ Filtergröße $(7 \times 7), \sigma = \sqrt{r}$
4: $Kern = Kern/\text{Summe}(Kern)$ ▷ Normierung

5: **for all** Frames of $Video$ **do**
6: $Kante =$ Kantenbild des Frames ▷ „Vorwärts-Differenzen-Operator“
7: Segmentierung von $Kante$ mittels $Seg1$
8: $Falt =$ Faltung von $Kante$ und $Kern$
9: Segmentierung von $Falt$ mittels $Seg2$
10: Behalte in jedem $(RG \times RG)$-Raster nur den maximalen Wert
11: **for all** $Pixel$ in $Falt$ **do**
12: **if** $Pixel > FT$ **then**
13: Schreibe Koordinaten des Pixels in Liste $Zentren$
14: Schreibe r in Liste $Radien$
15: Schreibe Wert des Pixels in Liste $Werte$
16: **end if**
17: **end for**
18: **end for**

19: **return** $[Zentren, Radien, Werte]$

5.1.3 Gauss-Halbkreise und Richtungsableitungen

Eine Möglichkeit zur Optimierung der Suche nach JAIL bietet an Stelle der Faltung mit Kreisen die Faltung mit Halbkreisen. Da ein JAIL nie einen vollständigen Kreis beschreibt, liefert die Suche nach Halbkreisen eine in Relation höhere Übereinstimmung des Kantenbildes mit dem Gauss-Kern. Hier suchen wir nach Halbkreisen in 8 unterschiedliche Richtungen. Von Halbkreisen, die nach oben zeigen und bei uns die Richtungsnummer 1 erhalten, gehen wir also im Uhrzeigersinn und 45°-Schritten zu den Halbkreisen, die nach oben links zeigen und die Richtungsnummer 8 bekommen.
Zum Erstellen der Faltungskerne verfahren wir zunächst genauso wie schon in Kapitel 5.1.2 beschrieben. Von einem fertigen Gauss-Kreis wird dann für den neuen Faltungskern lediglich noch die Hälfte der Pixel gelöscht, sodass nur noch ein Halbkreis übrig bleibt, der in die jeweils passende Richtung zeigt. Abbildung 5.5 zeigt alle 8 möglichen Halbkreise.

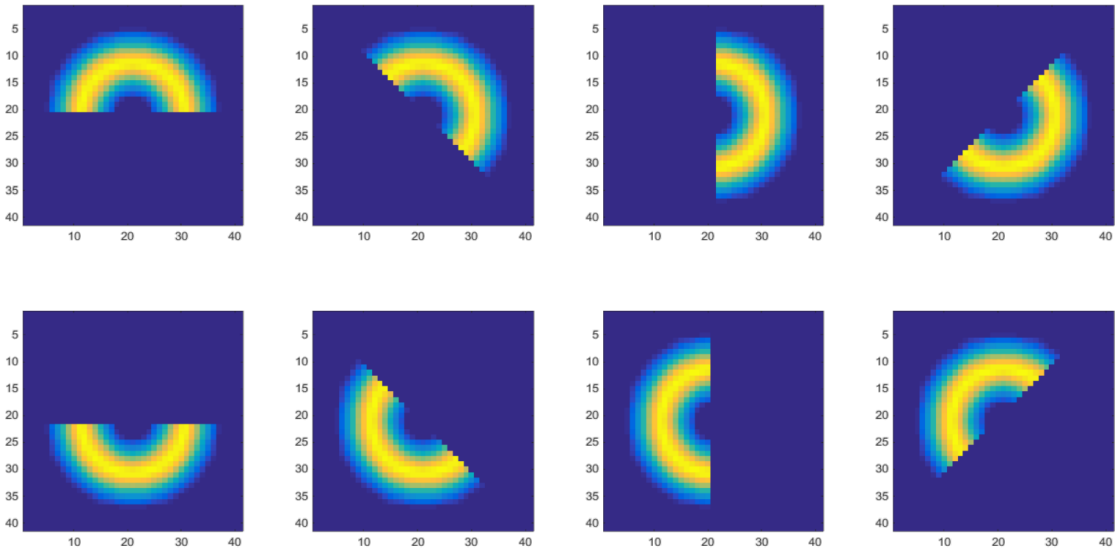

Abbildung 5.5: Gauss-Halbkreise in alle 8 Richtungen, $r = 10$.

Wir suchen nun nacheinander nach Halbkreisen in jede der 8 Richtungen und pro Richtung nach Radien zwischen 5 und 30 Pixeln. Bei dieser Suche machen wir Gebrauch von den Richtungsableitungen $\frac{du}{dx}$ und $\frac{du}{dy}$, deren Berechnung mit dem „Vorwärts-Differenzen-Operator“ wir schon in Gleichung (5.3) gesehen haben. Das Ergebnis der Berechnung der Richtungsableitungen sehen wir in Abbildung 5.6 anhand des gleichen Ausschnitts eines JAIL wie schon zuvor in Abbildung 5.4.

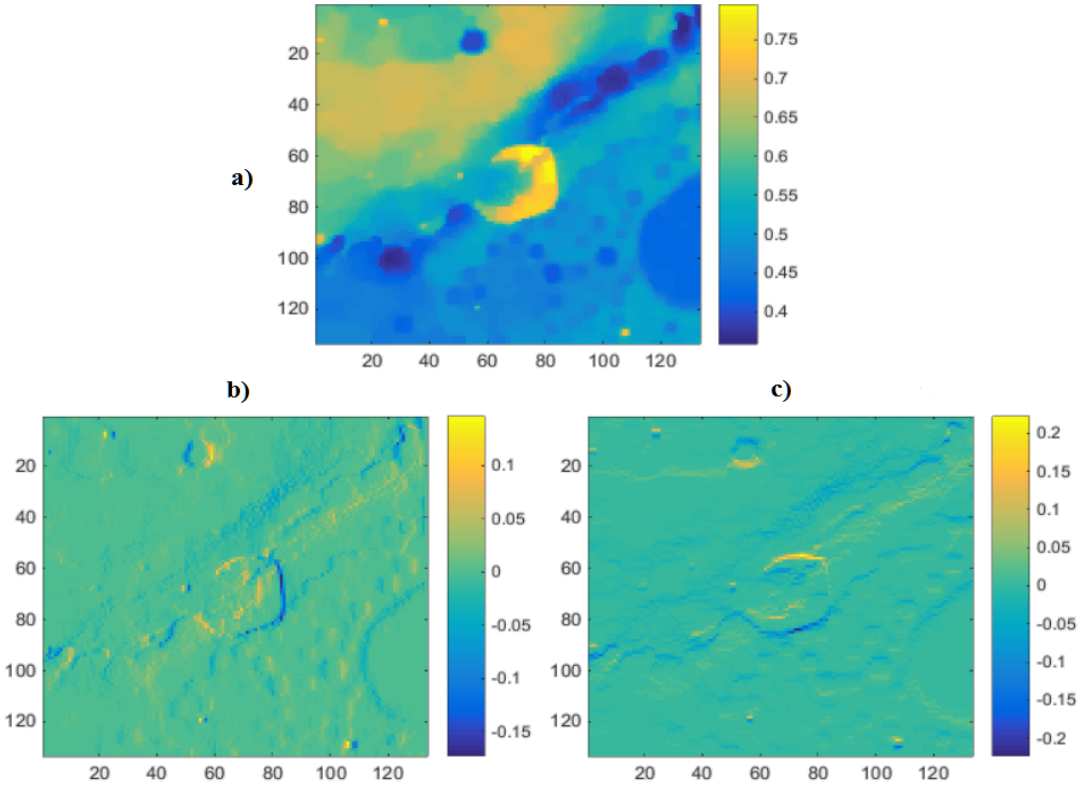

Abbildung 5.6: Ausschnitt eines JAIL (a) und den dazugehörigen Richtungsableitungen in x- (b) und y-Richtung (c)

Für jede der beiden Richtungsableitungen berechnen wir nun eine Faltung. Wir benötigen also einen Kern für die Faltung des Gradienten in x-Richtung und einen für die Faltung in y-Richtung. Im Beispiel dieses nach rechts zeigenden JAIL müssen wir aufgrund der Definition der Faltung für eine hohe Übereinstimmung des Kerns und der Richtungsableitungen einen Halbkreis benutzen, der nach links ausgerichtet ist. Der nach obiger Anleitung gebildete Halbkreis besitzt ausschließlich positive Werte. Wie in Abbildung 5.6 zu erkennen ist, ergibt die Ableitung in x-Richtung negative Werte an der rechten Kante des JAIL, während die Ableitung in y-Richtung am oberen Teil positive und am unteren Teil negative Werte ergibt. Dies liegt daran, dass der Gradient von den kleinen Werten außerhalb eines JAIL in Richtung der großen Werte im Inneren des JAIL zeigt. Hierbei ist zu beachten, dass Matlab die x-Richtung eines Bildes von links nach rechts und die y-Richtung von oben nach unten ansieht.
Um also auf möglichst hohe positive Werte im Faltungsbild zu kommen, sollte der Kern für die Faltung in x-Richtung auf der linken Seite negative und auf der rechten Seite positive Werte aufweisen. Der Faltungskern in y-Richtung sollte analog

dazu negative Werte im oberen Bereich sowie positive Werte im unteren Bereich des Halbkreises enthalten. Somit ist im Falle eines JAIL garantiert, dass sich bei der Faltung positive Werte der Richtungsableitung mit positiven Werten des Faltungskerns und negative Werte der Richtungsableitung mit negativen Werten des Faltungskerns multiplizieren. Ein hoher Gesamtwert der Faltung ist dementsprechend das Resultat.
Dieses Ergebnis erreichen wir durch eine Multiplikation eines Pixels aus dem ursprünglichen Faltungskern mit seinem jeweiligen Abstand vom Mittelpunkt in x- und y-Richtung. Damit der Abstand der Halbkreispixel zum Mittelpunkt bei großen Halbkreisen genauso gewertet wird wie bei kleinen Halbkreisen, wird der Abstand normiert. Für unser Beispiel sei also s ein nach links zeigender Gauss-Halbkreis mit Radius r. Weiterhin sei s_x der neue Faltungskern für die x-Richtung sowie s_y der für die y-Richtung. Dann gilt

$$\begin{aligned} \text{norm} &= \sqrt{(j-\text{mid})^2 + (i-\text{mid})^2}, \\ s_x(i,j) &= s(i,j) * (j-\text{mid})/\,\text{norm}, \\ s_y(i,j) &= s(i,j) * (i-\text{mid})/\,\text{norm} \end{aligned} \tag{5.4}$$

für $\text{mid} = \text{round}(\frac{4r+1}{2})$ und $i, j \in \{1, \ldots, 4r+1\}$.
Für den zum Beispiel-JAIL passenden Radius $r = 17$ sind die neuen Faltungskerne in Abbildung 5.7 dargestellt.

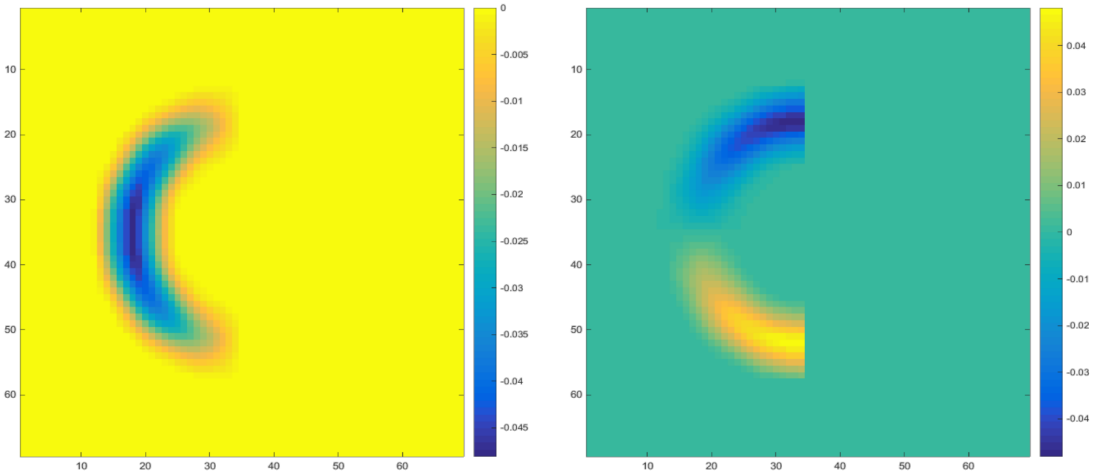

Abbildung 5.7: Faltungskerne mit Richtungsinformation eines nach links zeigenden Halbkreises in x- (links) und y-Richtung (rechts), $r = 17$

Mit diesen Kernen falten wir nun die beiden Richtungsableitungen und addieren die beiden Ergebnisse miteinander. Sei also $G_x = \frac{du}{dx}$ und $G_y = \frac{du}{dy}$. Außerdem sei

f_F das endgültige Faltungsbild. Dann gilt

$$f_F(i,j) = (G_x * s_x)(i,j) + (G_y * s_y)(i,j)$$

für $(i,j) \in \Omega$.
Mit diesem Ergebnis geht das Verfahren dann genauso weiter, wie es am Ende von Kapitel 5.1.2 bei der Faltung mit ganzen Kreisen erklärt wurde.

Um dieses Verfahren der Detektion von Halbkreisen nicht nur an den Videos von JAIL zu testen, haben wir uns als Testbeispiel zwei verschieden große Ellipsen erzeugt. Eine Ellipse der Form

$$E = \left\{ (x,y) : \frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} \leq 1 \right\}$$

mit dem Ursprung als Mittelpunkt und Radien r_x in x-Richtung sowie r_y in y-Richtung und $r_x > r_y$ sollten sowohl kleine, nach außen zeigende Halbkreise am linken und rechten Rand aufweisen als auch große, nach oben und unten zeigende Halbkreise am oberen und unteren Rand. Wir sehen in Abbildung 5.8 genau dieses Ergebnis an den zwei Ellipsen. Dies lässt auf eine hohe Genauigkeit der Halbkreiserkennung mit diesem Verfahren schließen.

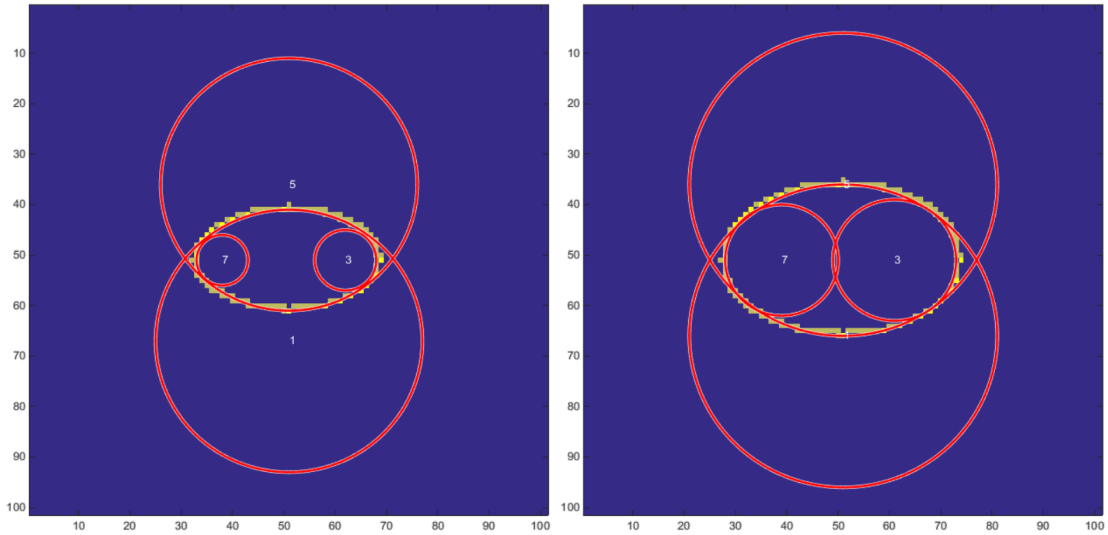

Abbildung 5.8: Ellipsen mit $(r_x, r_y) = (18, 10)$ (links) und $(r_x, r_y) = (23, 15)$ (rechts). Die roten Kreise stellen die gefundenen Halbkreise mit jeweiliger Richtungsnummer dar.

Zur Darstellung von Halbkreisen in Matlab haben wir der Einfachheit halber auf die Kreisdarstellung mittels *viscircles.m* zurückgegriffen und die Richtungsnummer

des Halbkreises an die Position seines Mittelpunktes geschrieben. Die Nummern gehen dabei von 1 für einen Halbkreis, der nach oben zeigt, im Uhrzeigersinn bis zur 8 für einen Halbkreis, der nach oben links zeigt (vgl. Abbildung 5.5).

Wir haben die Faltung mit Gauss-Halbkreisen mit dem Code aus Algorithmus 3 ausgeführt.

Algorithm 3 Faltung mit Gauss-Halbkreisen, *gauss_halbkreise_bestimmen.m*

Input:

$Video$	▷ entrauschtes Video
r	▷ Radius, nach dem gesucht werden soll
$Seg1$	▷ Segmentierung der Zellwände im Kantenbild (Abb. 4.3)
$Seg2$	▷ Segmentierung der Zellwände für die Kreis-Zentren (Abb. 4.4)
RG	▷ Raster-Größe, bei uns 40
FT	▷ Threshold für den Faltungswert
RN	▷ Richtungsnummer der Halbkreise

Output:

$Zentren$	▷ Mittelpunkte der gefundenen Kreise
$Radien$	▷ Radien der gefundenen Kreise
$Werte$	▷ Faltungswerte der gefundenen Kreise

$KG = 4 * r + 1$ ▷ Größe des Faltungskerns
$Kern =$ Kreis in $(KG \times KG)$-Matrix ▷ wie im Text beschrieben
Faltung von $Kern$ mit Gauss-Filter ▷ Filtergröße $(7 \times 7), \sigma = \sqrt{r}$
Lösche nicht benötigte Hälfte des Kreises ▷ abhängig von RN
$Kern = Kern/\text{Summe}(Kern)$ ▷ Normierung
Erstelle sx und sy ▷ siehe Beschreibung im Text
for all Frames of $Video$ **do**
    $[Gx, Gy] =$ Ableitungen in x- und y-Richtung ▷ „Vorwärts-Differenzen“
    Segmentierung von Gx und Gy mittels $Seg1$
    $Falt = (Gx * sx) + (Gy * sy)$
    Segmentierung von $Falt$ mittels $Seg2$
    Behalte in jedem $(RG \times RG)$-Raster nur den maximalen Wert
    for all $Pixel$ in $Falt$ **do**
        if $Pixel > FT$ **then**
            Schreibe Koordinaten des Pixels in Liste $Zentren$
            Schreibe r in Liste $Radien$
            Schreibe Wert des Pixels in Liste $Werte$
        end if
    end for
end for

return $[Zentren, Radien, Werte]$

5.2 Optischer Fluss

Ein JAIL ist kein statisches Objekt, sondern dehnt sich im Laufe seiner Existenz halbkreisförmig aus. Daher ist es sinnvoll, das Verfahren des optischen Flusses anzuwenden, welches Informationen über die Bewegung einer Struktur liefert. Zur Berechnung des optischen Flusses in x- und y-Richtung haben wir eine Toolbox bereitgestellt bekommen. Details zur Implementierung dieser Toolbox können in [Di] nachgelesen werden.
Da wir sowohl eine hohe Robustheit gegenüber Rauschen anstreben als auch Unstetigkeiten im Bewegungsfeld erlauben wollen, benutzen wir das $L^1 - TV$-Modell

$$\arg\min_v \|u_t + \nabla u \cdot v\|_1 + \alpha \sum_{i=1}^{n} |\nabla v_i|.$$

Der Testalgorithmus der Toolbox ermittelt die Bewegungsfelder von jeweils zwei aufeinanderfolgenden Frames. Für ein Video mit n Frames bekommt man also $n-1$ Bewegungsfelder in x- und genauso viele in y-Richtung. Als Beispiel dazu sehen wir in Abbildung 5.2 zwei aufeinanderfolgende Frames eines JAIL und die berechnete Bewegung dazwischen als Vektorfeld, dass sich aus den Bewegungsfeldern in beide Richtungen zusammensetzt. Um ein möglichst inhomogenes Bewegungsfeld zwischen zwei Frames der entrauschten Daten zu bekommen, hat sich dabei der Parameter $\alpha = 0.006$ bewährt.

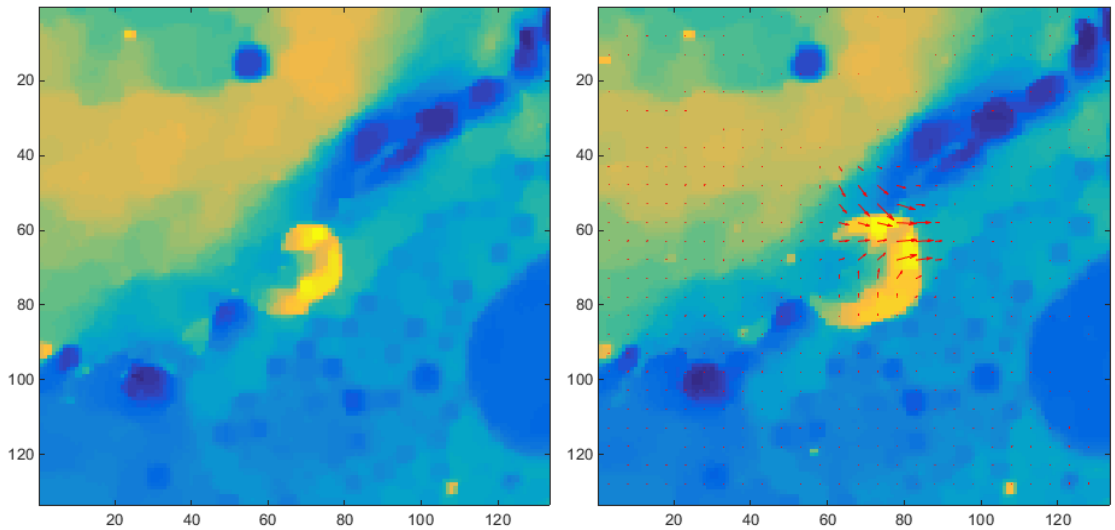

Abbildung 5.9: Zwei aufeinanderfolgende Frames eines JAIL mit dem optischen Fluss zwischen ihnen (rechts), $\alpha = 0.006$.

Zum Löschen von falsch positiven JAIL benutzen wir nun die euklidische Norm

$$\|\omega\|_2 = \sqrt{\omega_x^2 + \omega_y^2}$$

der beiden Bewegungsfelder ω_x und ω_y. Davon erstellen wir zunächst ein Binärbild mithilfe eines Thresholds. Um auch hier genug Spielraum zu haben und nicht zu viele Pixel zu löschen, wählen wir dessen Wert als 1. Mithilfe dieses Binärbildes überprüfen wir nun jeden gefundenen JAIL mit Radius r. Wir testen die Übereinstimmung der Flow-Norm mit einer Kreisscheibe um den Mittelpunkt des JAIL. Diese bekommt einen inneren Radius von $r - 4$ und einen äußeren Radius von $r + 4$. Überdeckt die Kreisscheibe nun weniger als 20 übrig gebliebene Pixel der Flow-Norm, so wird angenommen, dass es sich um einen falsch detektierten JAIL handelt und er wird gelöscht.

5.3 Aufsummierung der Intensitäten

Da die Pixel eines JAIL aufgrund des grün fluoreszierenden Proteins EGFP, mit dem das p20-Protein für die Bildung von Aktinfilamenten markiert wurde (vgl. Kapitel 2), eine höhere Intensität als seine Umgebung aufweisen, kann man mit einem Vergleich der Intensitäten falsch positive JAIL ausschließen. Hierzu gehen wir die Liste der potenziellen JAIL durch und überprüfen jeweils, ob zur Zeit des Auftritts ein lokales Maximum in der Intensitätenkurve über die Zeit t vorliegt oder nicht. Wir vergleichen dabei nur die Intensitäten in einer kreisförmigen Umgebung U um den JAIL mit Mittelpunkt $mid = (m_x, m_y) \in \mathbb{R}^{m \times n}$. Der Radius der Umgebung entspricht dabei dem ermittelten Radius r des JAIL. Es gilt also

$$U = \left\{ (i,j) : \sqrt{(i - m_y)^2 + (j - m_x)^2} \leq r \right\}.$$

Für jedes Frame werden nun alle Intensitäten in dieser Umgebung aufsummiert und als Kurve über die Zeit dargestellt. Für die Intensität der Umgebung zur Zeit t gilt somit

$$I(t) = \sum_{(i,j) \in U} f(i,j,t).$$

Befindet sich in einer Zeitspanne von einem Frame vor bis einem Frame nach der Detektion des JAIL ein lokales Maximum in der Intensitäten-Kurve, so wird er weiterhin als potenzieller JAIL gewertet. Andernfalls wird er als falsch positiver JAIL aus der Liste gelöscht.
Ein Beispiel für diese Kurve bei einem positiven JAIL sehen wir in Abbildung

5.10. Hierbei geht es um den JAIL in Frame 8 unseres Ausschnittes (oben rechts in der Abbildung). Wir vergleichen also die Intensitäten eines jeden Frames in der Umgebung U mit Radius $r = 17$ um den Mittelpunkt des JAIL aus Frame 8. Da in diesem Frame ein deutlich zu erkennendes Maximum in der Intensitätenkurve vorliegt, wird der JAIL nicht aus der Liste gelöscht.

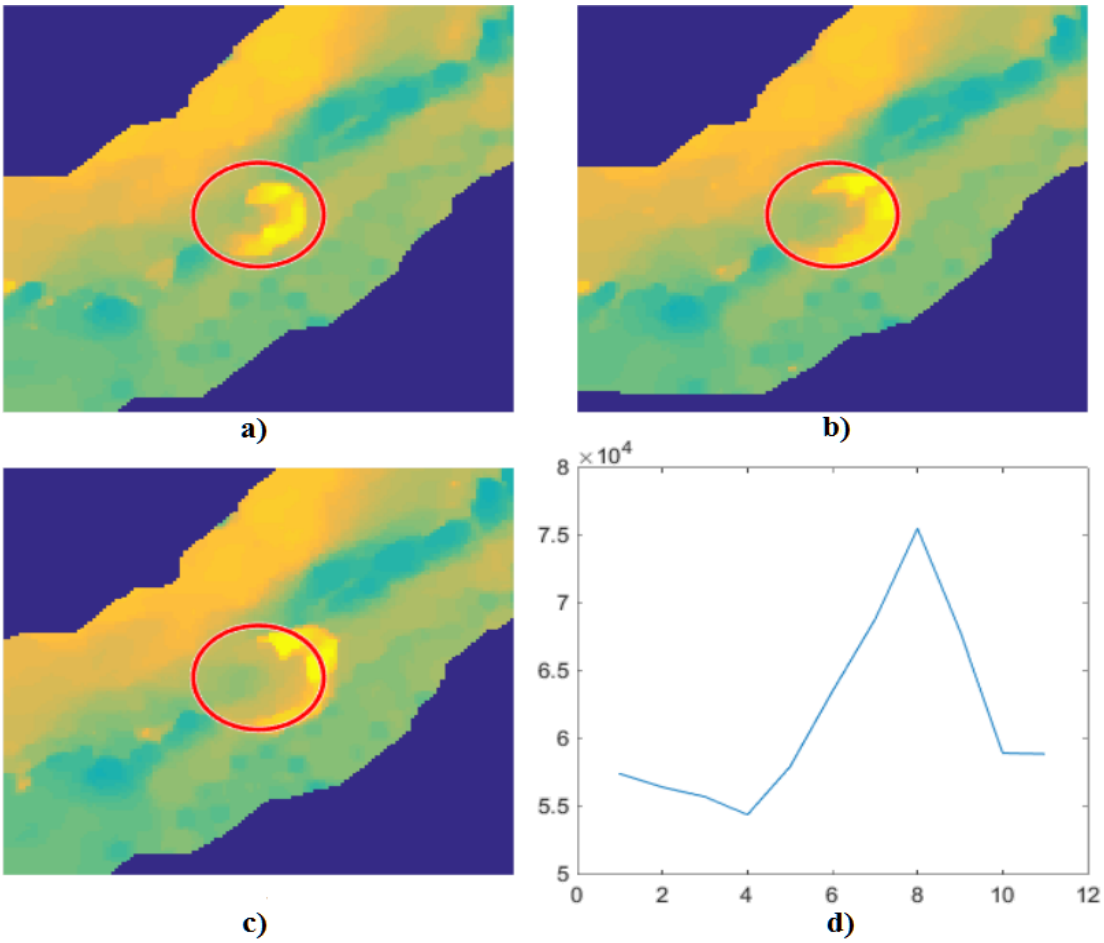

Abbildung 5.10: Frame 7 (a), 8 (b) und 9 (c) eines JAIL-Ausschnittes mit insgesamt 11 Frames und die passende Intensitäten-Kurve (d). Der rote Kreis entspricht der Umgebung U um den Mittelpunkt des JAIL aus Frame 8 mit dazugehörigem Radius $r = 17$.

Für die Aufsummierung der Intensitäten in Matlab haben wir den Code aus Algorithmus 4 benutzt.

Algorithm 4 Aufsummierung der Intensitäten, *check_intensities.m*

Input:
$Video$ ▷ verrauschtes Original-Video
U ▷ Umgebung um aktuellen JAIL
FN ▷ FrameNumber, Zeitpunkt des JAIL-Auftritts
Output:
$IsJail$ ▷ 0 (kein JAIL) oder 1 (JAIL)

$YWerte :=$ leerer Vektor ▷ Länge $\widehat{=}$ Anzahl der Frames
for all Frames of $Video$ **do**
    $t :=$ Nummer des aktuellen Frames
    $Summe = \sum I(x, y, t)$ ▷ $\forall (x, y) \in U$
    $YWerte(t) = Summe$
end for
$IsJail := 0$
suche lokale Maxima in $YWerte$
for all Maxima **do**
    if Maximum liegt bei Zeit $t \in \{FN - 1, \ldots, FN + 1\}$ **then**
        $IsJail = 1$
    end if
end for

return $IsJail$

5.4 Connected Components

Eine weitere Idee, die Anzahl falsch positiver JAIL zu verringern, ist die Verwendung der sogenannten „Connected Components“. Da ein JAIL typischerweise ein zusammenhängendes Konstrukt mit hohen Intensitätswerten darstellt, suchen wir in der Nähe eines potenziellen JAIL nach Zusammenhangskomponenten und löschen ihn, falls sich keine solche Komponente in der Nähe befindet. Um diese Komponenten sinnvoll zu bestimmen, benötigt man zunächst einmal ein Binärbild. Dabei spielt die Größe des Intensitäten-Thresholds eine wichtige Rolle. Wählt man diesen zu klein und erstellt dann das Binärbild, erhält man zu wenige und zu große Komponenten. Andersherum werden zu viele kleine Komponenten gefunden, wenn der Threshold zu groß gewählt ist. Um möglichst alle korrekten JAIL beizubehalten, hat sich der Intensitätswert 90 hierbei als am geeignetsten herausgestellt. In Abbildung 5.11 sind Ergebnisse mit verschieden großen Thresholds dargestellt.

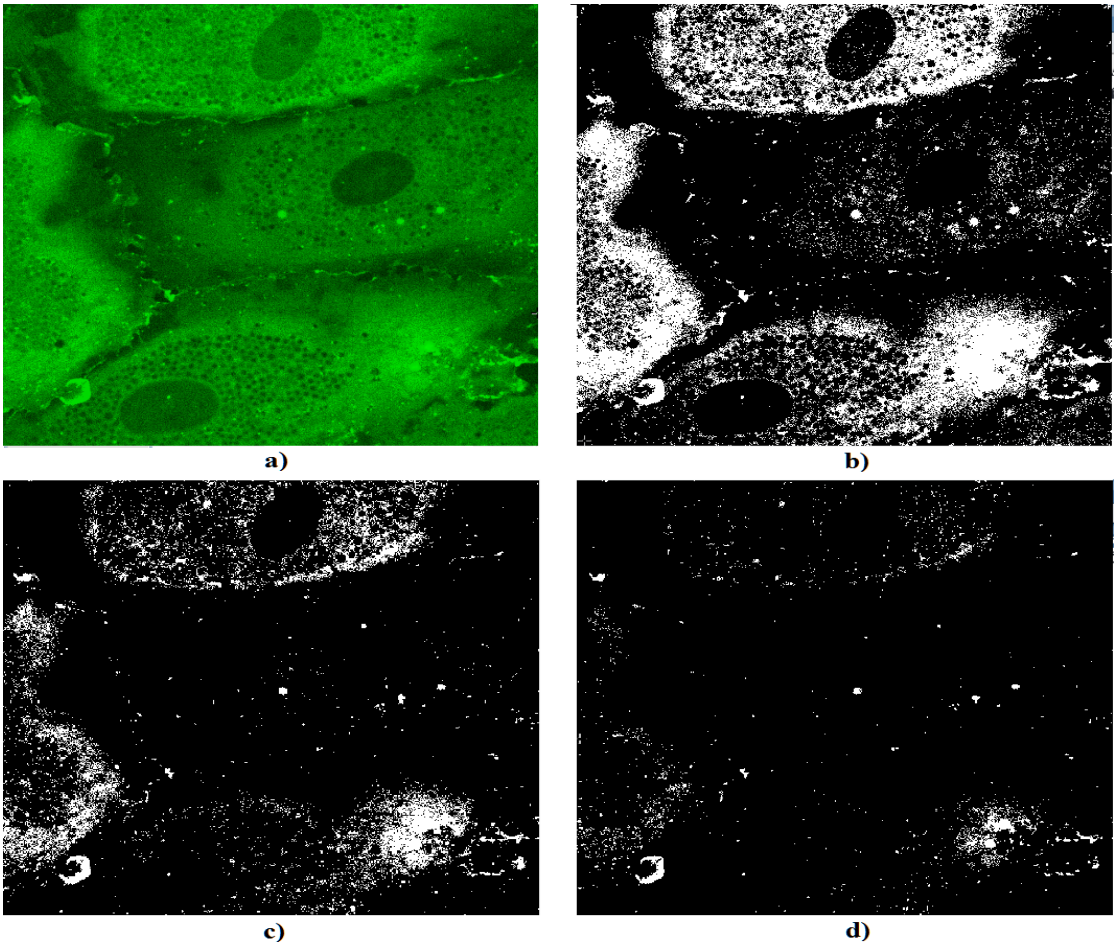

Abbildung 5.11: Originalbild (a) und Binärbilder mit Intensitäts-Thresholds 70 (b), 80 (c) und 90 (d)

In diesem Binärbild suchen wir nun Zusammenhangskomponenten. Hierbei zählen Pixel, die sich in einer Achter-Nachbarschaft, der sogenannten „Chessboard-Metrik“ (vgl. [SR, S. 9])

$$R_8(x, y) = \{(u, w) : \max(|u - x|, |w - y|) = 1\}$$

zueinander befinden, zur gleichen Komponente. Der Unterschied zu einer Vierer-Nachbarschaft oder „City-Block-Metrik“

$$R_4(x, y) = \{(u, w) : |u - x| + |w - y| = 1\}$$

ist in Abbildung 5.12 dargestellt.

Abbildung 5.12: Vierer-Nachbarschaft (a) und Achter-Nachbarschaft (b) (aus [KŽ, S. 397])

Da bei diesem Verfahren auch allein stehende Pixel als eigene Komponente gezählt werden, benötigen wir nun einen weiteren Threshold, der die Mindestgröße einer Komponente bestimmt. Um auch die kleinsten JAIL mit einzuschließen, hat sich hier der Wert von 40 Pixeln als minimale Größe bewährt. Abbildung 5.13 illustriert die Zusammenhangskomponenten des Binärbildes aus Abbildung 5.11.

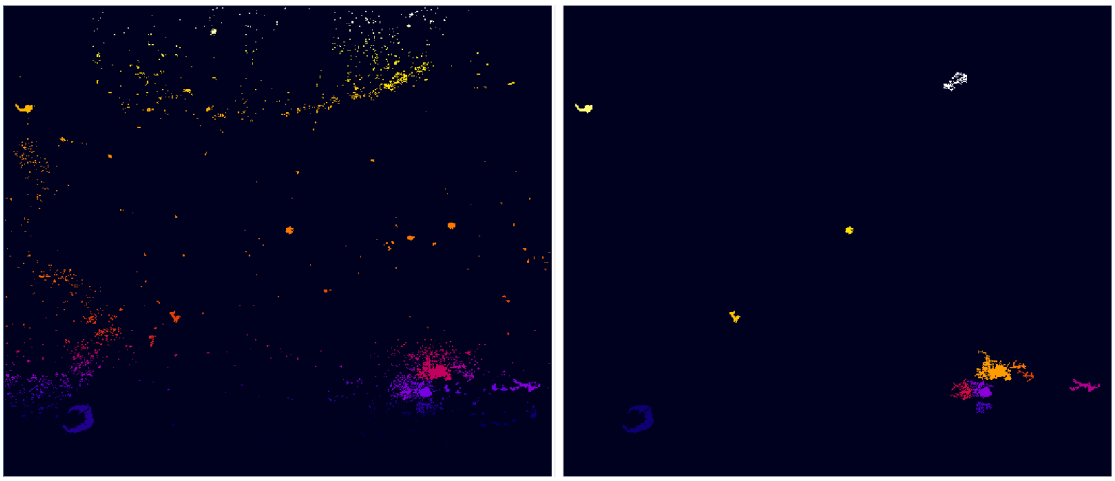

Abbildung 5.13: Zusammenhangskomponenten ohne (links) und mit Größen-Threshold 40 (rechts). Die Farbabstufungen dienen zur Unterscheidung der verschiedenen Komponenten.

Der Vergleich der Zentren von gefundenem JAIL und Zusammenhangskomponenten in seiner Umgebung liefert nun eine Aussage darüber, ob der JAIL als falsch positiv aus der Liste gelöscht wird oder nicht. Hier werden die x- und y-Koordinaten der beiden Zentren verglichen. Sind diese jeweils mehr als 12 Pixel voneinander entfernt, so wird der JAIL gelöscht.

Für die Berechnung der Connected Components in Matlab haben wir den Code in Algorithmus 5 benutzt:

Algorithm 5 Connected Components, *conn_comp.m*

Input:
$Video$ ▷ verrauschtes Original-Video
GT ▷ Größen-Threshold, bei uns 40
IT ▷ Intensitäts-Threshold, bei uns 90
Output:
$Zentren$ ▷ Mittelpunkte der Connected Components

1: $Video(Video < IT) = 0$
2: $Video(Video \geq IT) = 1$

3: **for all** Frames of $Video$ **do**
4: berechne Connected Components ▷ 8er-Nachbarschaft
5: **if** Anzahl Pixel einer Komponente $< GT$ **then**
6: lösche Komponente
7: **end if**
8: berechne Mittelpunkte für jede Komponente
9: Schreibe Mittelpunkte in Liste $Zentren$
10: **end for**

11: **return** $Zentren$

5.5 Faltungswerte akkumulieren

Ein Ansatz für eine mögliche Zeitstetigkeit bei der Erkennung von JAIL ist die Akkumulation der Faltungsbilder über die Zeit. Dabei könnte man in einem dreidimensionalen Ort-Radius-Diagramm mit den Koordinatenachsen (x, y, r) nach einer Abfolge von hellen Werten in r-Richtung suchen, welche auf einen wachsenden JAIL hindeuten. Der Wert eines Punktes in diesem Diagramm wird mit der Abbildung

$$(x, y, r) \longmapsto \int_0^T f_F(x, y, r, t)\,\mathrm{d}t \tag{5.5}$$

berechnet, wobei f_F ein Faltungsbild, x und y die Bildkoordinaten, r den Radius, t die Zeit und T die Anzahl der Frames darstellt.

Zur Verdeutlichung dieser Idee haben wir uns ein kleines Testbeispiel erstellt. Es besteht aus einem Video der Größe $(71 \times 71 \times 6)$, in denen ein nach unten zeigender, wachsender Halbkreis zu sehen ist. Dieser behält seinen Mittelpunkt im Punkt $(36, 36)$ und hat in dem ersten Frame einen Radius von 5 Pixeln und wird mit jedem Frame um 5 Pixel größer, bis er im sechsten Frame einen Radius von 30 Pixeln aufweist (siehe Abbildung 5.14).

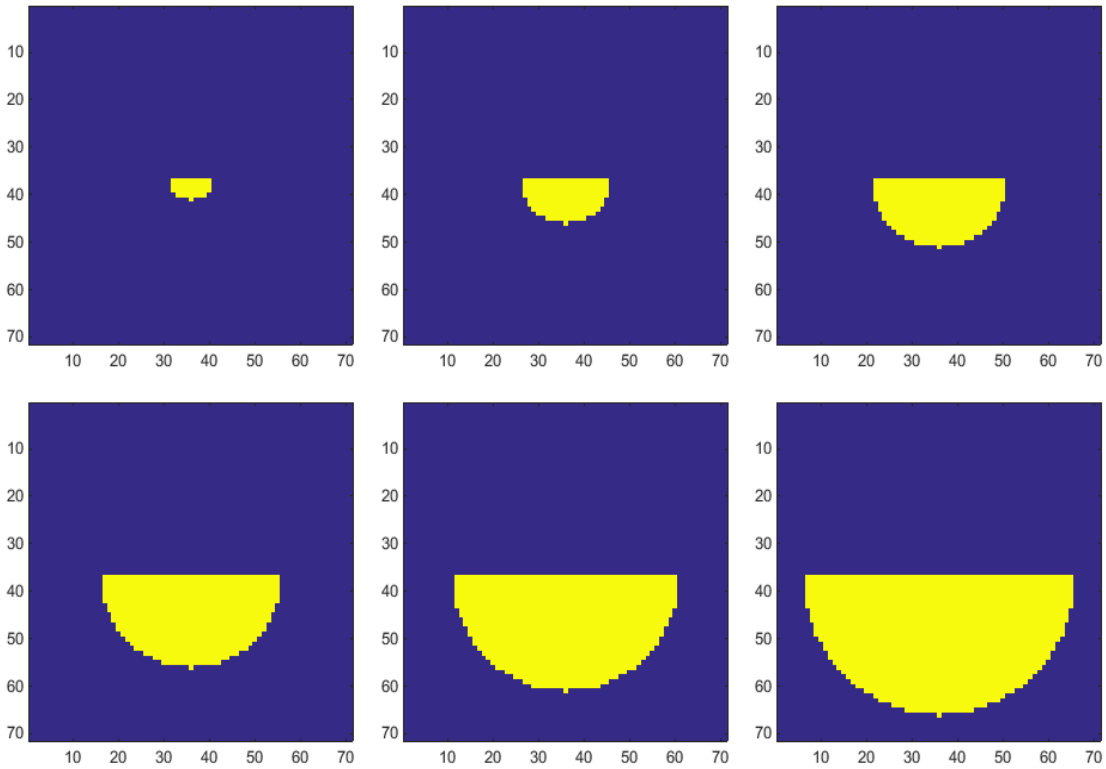

Abbildung 5.14: Selbst erstelltes Testbeispiel. Ein statischer Halbkreis, der von einem Radius von 5 Pixeln im ersten Frame zu 30 Pixeln im letzten Frame wächst.

Wir suchen nun mit der in Kapitel 5.1.3 erklärten Methode nach Halbkreisen in dem Testbeispiel. Hierbei legen wir das Augenmerk der Suche der Einfachheit halber nur auf diejenigen Halbkreise, die ebenfalls nach unten zeigen. Wenn wir Gleichung (5.5) auf unser Testbeispiel beziehen, erhalten wir mithilfe der diskreten Form

$$(x, y, r) \longmapsto \sum_{t=1}^{6} f_F(x, y, r, t)$$

das gewünschte Ort-Radius-Diagramm. Um eine bessere Visualisierung der Faltungswerte mit wachsendem Radius zu bekommen, ohne dabei zwischen mehreren Frames hin und her zu wechseln, permutieren wir das Diagramm und vertauschen die y- und die r-Achse. In Abbildung 5.15 sehen wir Frame 36 aus diesem Diagramm, also ein (30×71)-Bild, welches die Faltungswerte der verschiedenen Radien und x-Koordinaten über die Zeit aufsummiert und für festes $y = 36$ zeigt.

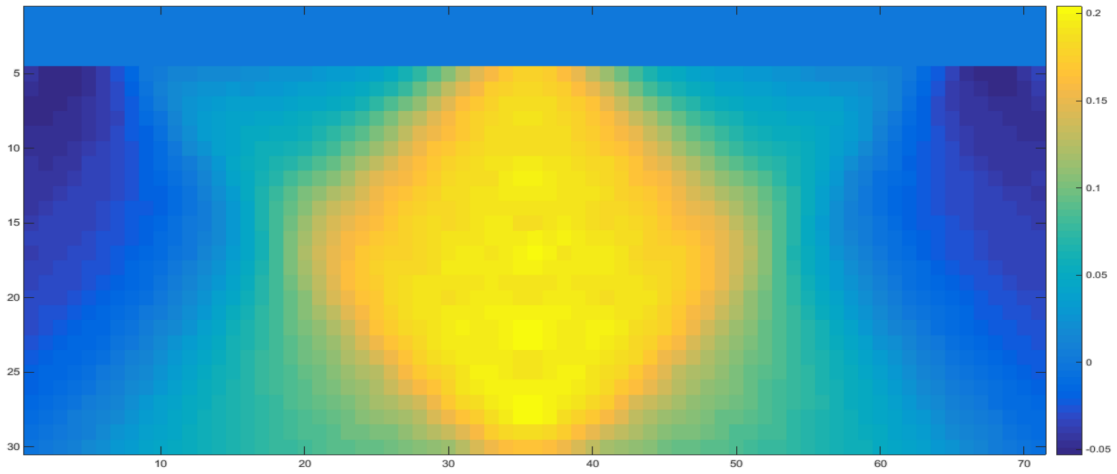

Abbildung 5.15: Akkumulierte Werte der Faltungen des Testbeispiels mit nach unten zeigenden Gauss-Halbkreisen (weitere Erläuterungen im Text).

Hier kann man bei $x = 36$ die höchsten Werte der Faltungen erkennen. Diese treten in regelmäßigen Abständen von $r = 5$ bis $r = 30$ in dem Bild auf.

Wir haben nun ein statisches Beispiel mit einem Halbkreis gesehen, der sich nicht bewegt, sondern nur auf der Stelle wächst. Für die Praxis und unsere Videos von JAIL ist so ein einzelnes Bild jedoch wenig aussagekräftig, da es sich dabei um Objekte mit einer gewissen Bewegung von Frame zu Frame handelt. Um dieses Verfahren also nutzen zu können, wäre es zunächst einmal sinnvoll, einen JAIL über die Zeit seiner Existenz zu verfolgen. Daraufhin könnte man eventuell anhand dieser Bewegung von einem Frame zum nächsten einen ebenso statischen Wachstumsvorgang erzeugen wie in unserem Testbeispiel.
Zur Veranschaulichung dieser Problematik sehen wir in Abbildung 5.16 ein weiteres Testbeispiel. Im Unterschied zum vorherigen Halbkreis verändert dieser nun jedoch seine Position und wandert von oben links zur Mitte des Bildes. Der Radius verändert sich genau wie bei dem obigen Beispiel und wächst von $r = 5$ im ersten Frame zu $r = 30$ im sechsten Frame.

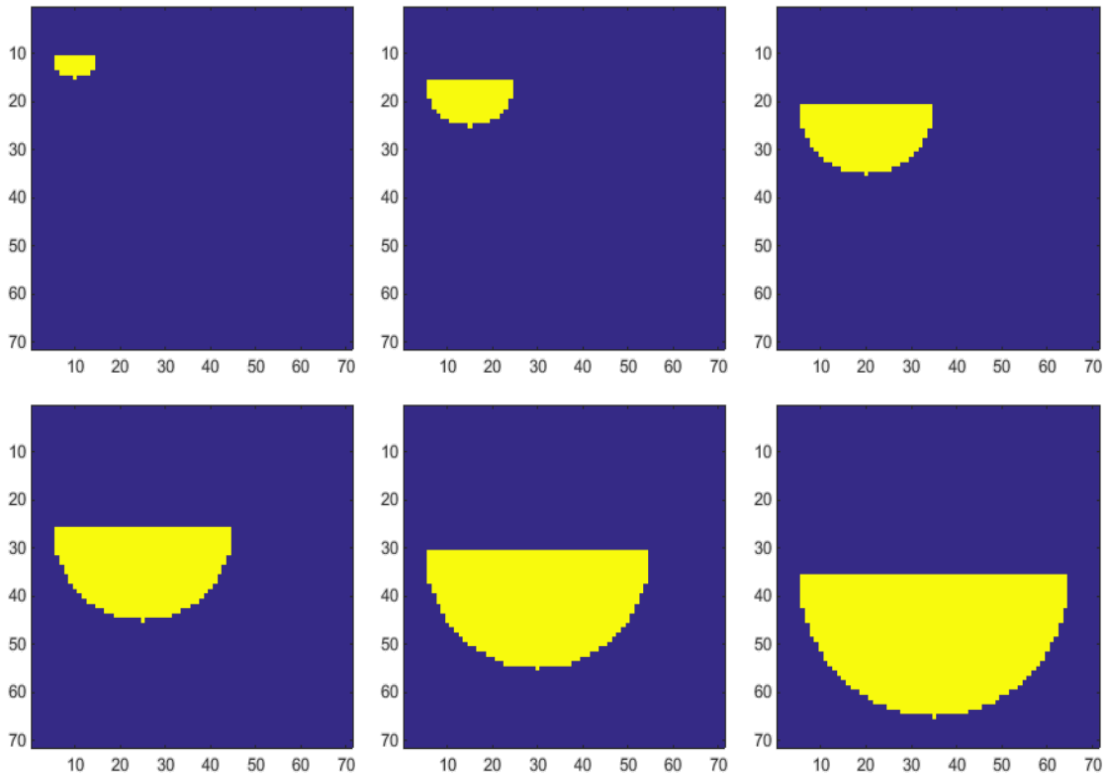

Abbildung 5.16: Weiteres selbst erstelltes Testbeispiel. Der Kreis aus Abbildung 5.14 ist hier nicht mehr statisch, sondern bewegt sich von oben links in die Mitte des Bildes.

Eine Abfolge von hellen Werten in einem einzigen Bild können wir nun nicht mehr erkennen. Wir müssten also anhand der Bewegung des Halbkreises in denjenigen Bildern nach hohen Werten suchen, die den jeweils passenden y-Wert des Halbkreismittelpunktes repräsentieren. Da sich dieser in dem Beispiel von $(10, 10)$ in regelmäßigen Abständen von 5 Pixeln zu $(35, 35)$ verschiebt, wären das die Frames $10, 15, 20, 25, 30$ und 35 in dem Akkumulator mit Koordinatenachsen (x, r, y). Abbildung 5.17 zeigt diese 6 Frames.

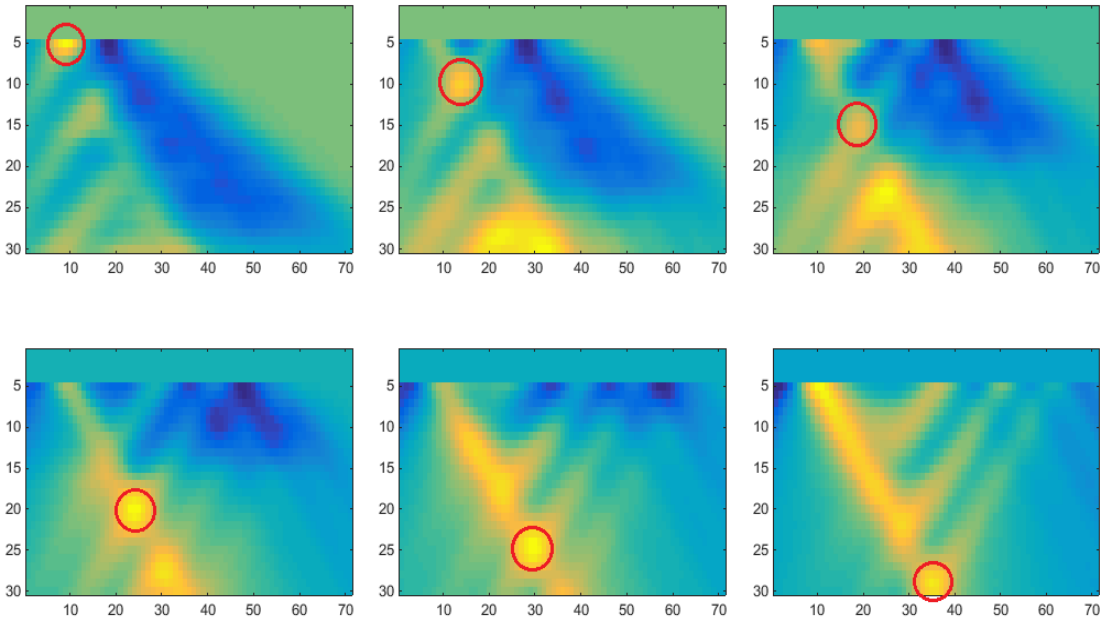

Abbildung 5.17: Akkumulierte Werte der Faltungen des zweiten Testbeispiels. Die dargestellten Frames sind diejenigen, die den jeweiligen y-Wert des Mittelpunktes aus Abbildung 5.16 repräsentieren. Der rote Kreis zeigt die Stellen, an denen sich die Mittelpunkte mit passendem Radius befinden.

Wir sehen hier, dass an den richtigen Stellen, die durch die roten Kreise in der Abbildung markiert sind, hohe Werte der akkumulierten Faltungen auftreten. Diese sind aber nicht die einzigen und auch nicht immer die größten Werte. Eine mögliche Erklärung dafür könnte die Akkumulierung der Faltungswerte über die Zeit sein, die bei einem sich bewegenden und wachsenden Kreis an vielen Stellen im Bild hohe Werte für verschiedene Radien liefert.

6 Workflow

In diesem Kapitel werden wir auf die in Kapitel 4 und 5 erklärten Verfahren zurückkommen und den daraus zusammengesetzten Workflow erklären. Wir werden also sehen, welche Verfahren letztendlich in welcher Reihenfolge angewandt wurden und was dabei für Ergebnisse herausgekommen sind.

Der erste Schritt war die Entrauschung mithilfe der Kullback-Leibler-Divergenz und der TV-Regularisierung. Wir haben festgestellt, dass das Kantenbild eines JAIL zum Teil eine ungleiche Verteilung an hohen Werten innerhalb der Kante aufweist. Dadurch wäre die Erkennung von Kreisen und Halbkreisen erheblich geschwächt worden. Deshalb haben wir vor dem Entrauschen des Videos alle Werte, die höher als 120 waren, auf 120 heruntergesetzt. Das Kantenbild war somit sehr viel gleichmäßiger, wie Abbildung 6.1 zeigt.

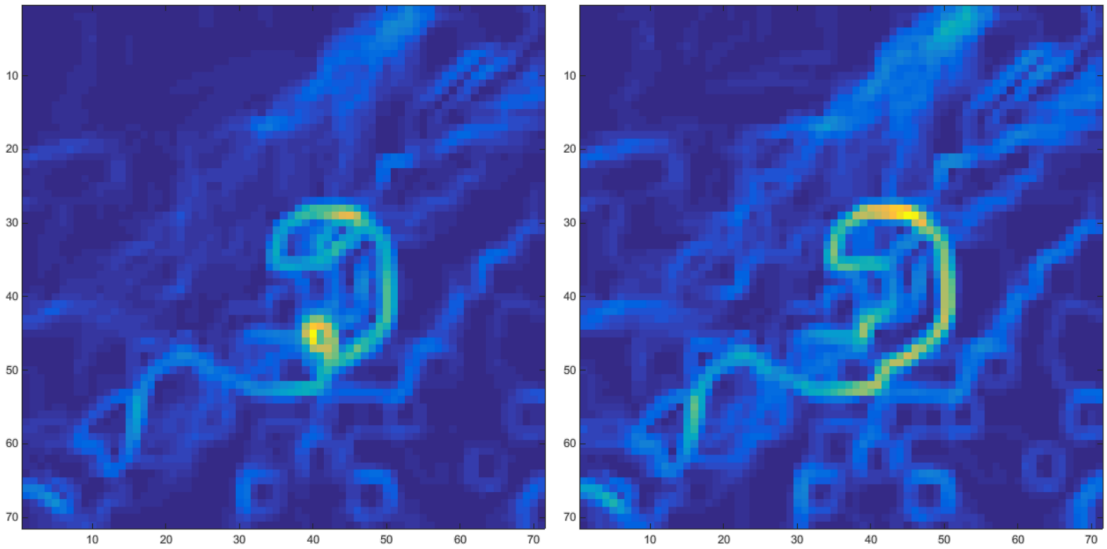

Abbildung 6.1: Kantenbild eines JAIL nach Entrauschung ohne (links) und mit runtergesetzten Werten (rechts)

Nach diesem Heruntersetzen der hohen Werte konnte das Video entrauscht werden. Wir haben dafür den Wert $\lambda = 8.5$ als Gewichtung für den Kullback-Leibler-Datenterm gewählt. Als die für den Algorithmus von Chambolle und Pock notwendigen Parameter haben wir $\sigma = \tau = \frac{1}{3}$ sowie $\theta = 1$ gewählt, womit der Algorithmus laut [CP] konvergiert. Abbildung 6.2 zeigt das Ergebnis der Entrauschung an dem Ausschnitt eines JAIL. Aufgrund der langen Laufzeit des Programms wurde das Video einmalig entrauscht und abgespeichert. In dem Workflow wird es also nur noch eingeladen und nicht jedes Mal erneut entrauscht.

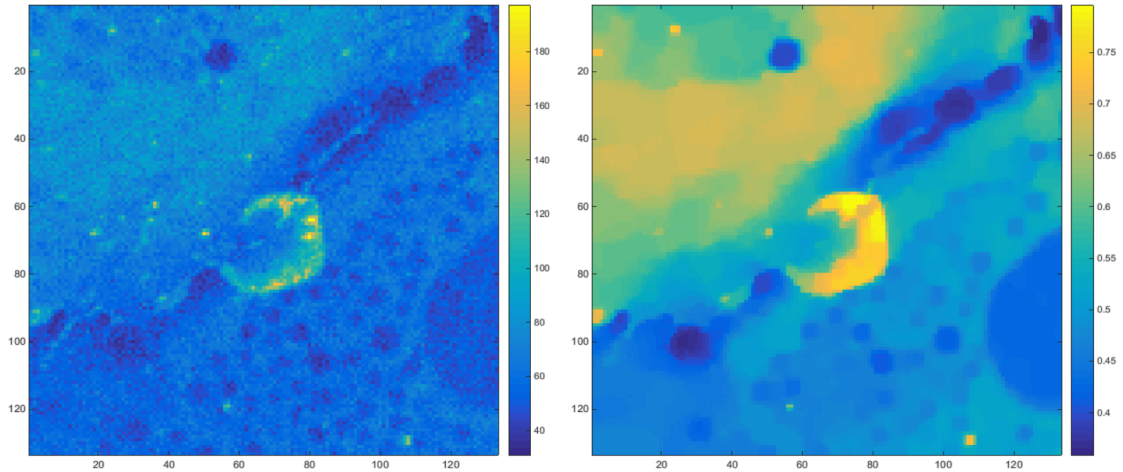

Abbildung 6.2: Ausschnitt eines JAIL vor (links) und nach der Entrauschung (rechts)

Vor diesem Einlesen des entrauschten Videos steht die Eingabe aller benötigten Parameter. Neben den bereits in den einzelnen Kapiteln erklärten Parametern kann man hier den gewünschten Ausschnitt des Videos bestimmen. Falls also nicht das gesamte Video bearbeitet werden soll, gibt man hier das Start- und das Endframe sowie die x- und y-Koordinaten des Ausschnittes an. Nach Erstellung der Segmentierungen werden dann das entrauschte Video, das Original-Video sowie die drei verschiedenen Segmentierungen auf eine einheitliche Größe geschnitten.
In Tabelle 6.1 haben wir die zu wählenden Parameter des Workflows noch einmal zusammengefasst.

Wie bereits erwähnt werden nun die drei in Kapitel 4.3 erklärten Segmentierungen erstellt. Das entrauschte Video wird daraufhin zunächst mithilfe der Segmentierung mit dem größten Umgebungsradius von 35 Pixeln segmentiert (siehe Abbildung 4.2). Dieses Video wird dann zusammen mit den anderen beiden Segmentierungen der Funktion *gauss_halbkreise_bestimmen.m* mitgegeben.

Parameter	Bedeutung	Wertebereich und Wahl
$x1$	linker Pixel des Ausschnitts	[1, 512], 1
$x2$	rechter Pixel des Ausschnitts	[1, 512], 512
$y1$	oberer Pixel des Ausschnitts	[1, 512], 1
$y2$	unterer Pixel des Ausschnitts	[1, 512], 512
$start\_frame$	erstes Frame des gewünschten Ausschnitts	[1, 503], 1
end_frame	letztes Frame des gewünschten Ausschnitts	[1, 503], 503
$rast\_size$	Größe des Rasters, in dem nur der maximale Faltungswert behalten wird	40
$score\_thresh$	unterer Threshold für den Faltungswert	[0, 1], 0.15
$umgeb\_radius$	Umgebungsgröße der Zellwände zur Segmentierung	30
$normierung$	1 = mit Normierung in (5.4), 0 = ohne Normierung	{0,1}, 1
$size\_cc$	minimale Größe der Connected Components	40
$intens\_cc$	Intensitäts-Threshold für Connected Components	90
$diff\_centers$	erlaubte Distanz der Zentren von JAIL und Connected Components	12
$flow\_thresh$	minimaler Wert der Flow-Norm	1
anz_pixel	Übereinstimmung von Werten der Flow-Norm und des JAIL an dessen Position	20

Tabelle 6.1: Wählbare Parameter für den Workflow

Mit dieser Funktion folgt dann die Detektion der Halbkreise. Da sich die anfängliche Suche nach ganzen Kreisen als wenig praktikabel erwiesen hat, benutzen wir nur noch Halbkreise für die Detektion von JAIL. Vor allem in Bezug auf die Ausrichtung der JAIL bekommen wir somit Informationen, die wir bei der Kreisdetektion nicht bekommen würden. Wir haben am Ende von Kapitel 5.1.3 am Beispiel von Ellipsen gesehen, dass das vorgestellte Verfahren im Grunde seinen Zweck erfüllt und genau die passenden vier Halbkreise detektiert. Leider ist dies bei den vorliegenden Daten nicht der Fall. Hier werden fast ausschließlich die kleinsten Kreise, also die mit Radius $r = 5$ mit einem hohen Wert versehen. Abbildung

6.3 zeigt dies anhand von vier aufeinanderfolgenden Frames unseres üblichen Beispiels. Auch hier geht die Halbkreisnummer von 1 (nach oben) bis 8 (nach oben links). Wir werden in den folgenden Abbildungen die Kantenbilder der entrauschten Daten zeigen, da man die Kontur der JAIL so deutlicher erkennen kann.

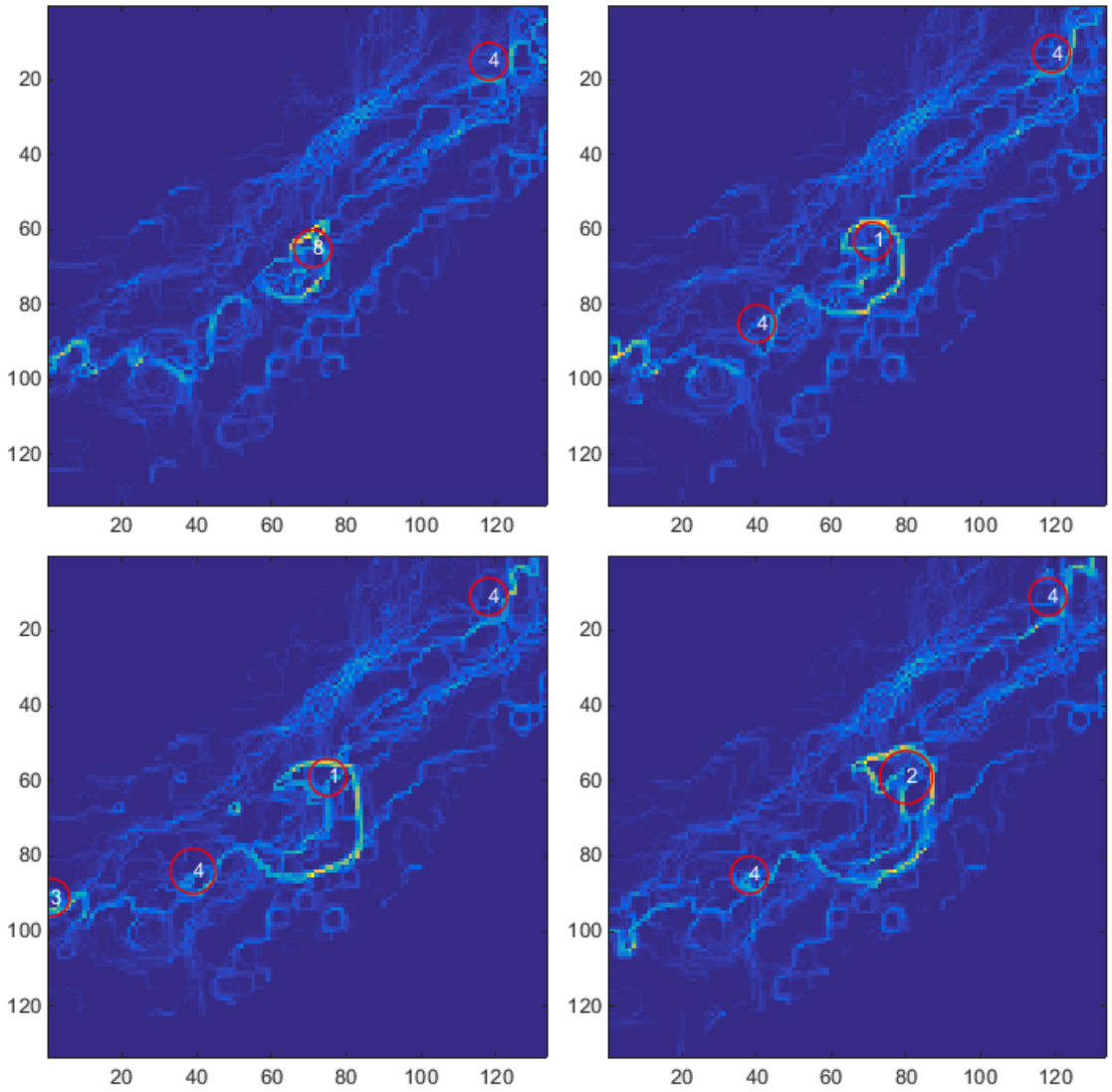

Abbildung 6.3: Kantenbilder von vier aufeinander folgenden Frames eines JAIL und die detektierten Halbkreise.

Der Grund für diesen Fehler ist uns nicht genau bekannt. Das Ergebnis der gefundenen Halbkreise lässt sich jedoch verbessern, indem wir bei der Erstellung der Faltungskerne in x- und y-Richtung in Gleichung (5.4) auf die Normierung

der Abstände verzichten. Obwohl dies mathematisch gesehen nicht sinnvoll ist, werden daraufhin neben einigen falsch positiven auch die richtigen, zu dem JAIL passenden Halbkreise gefunden. Dies zeigt Abbildung 6.4. Ob wir die Normierung im Workflow letztendlich benutzen möchten oder nicht, können wir mithilfe des Parameters „*Normierung*“ , den wir als 0 oder 1 wählen können, entscheiden.

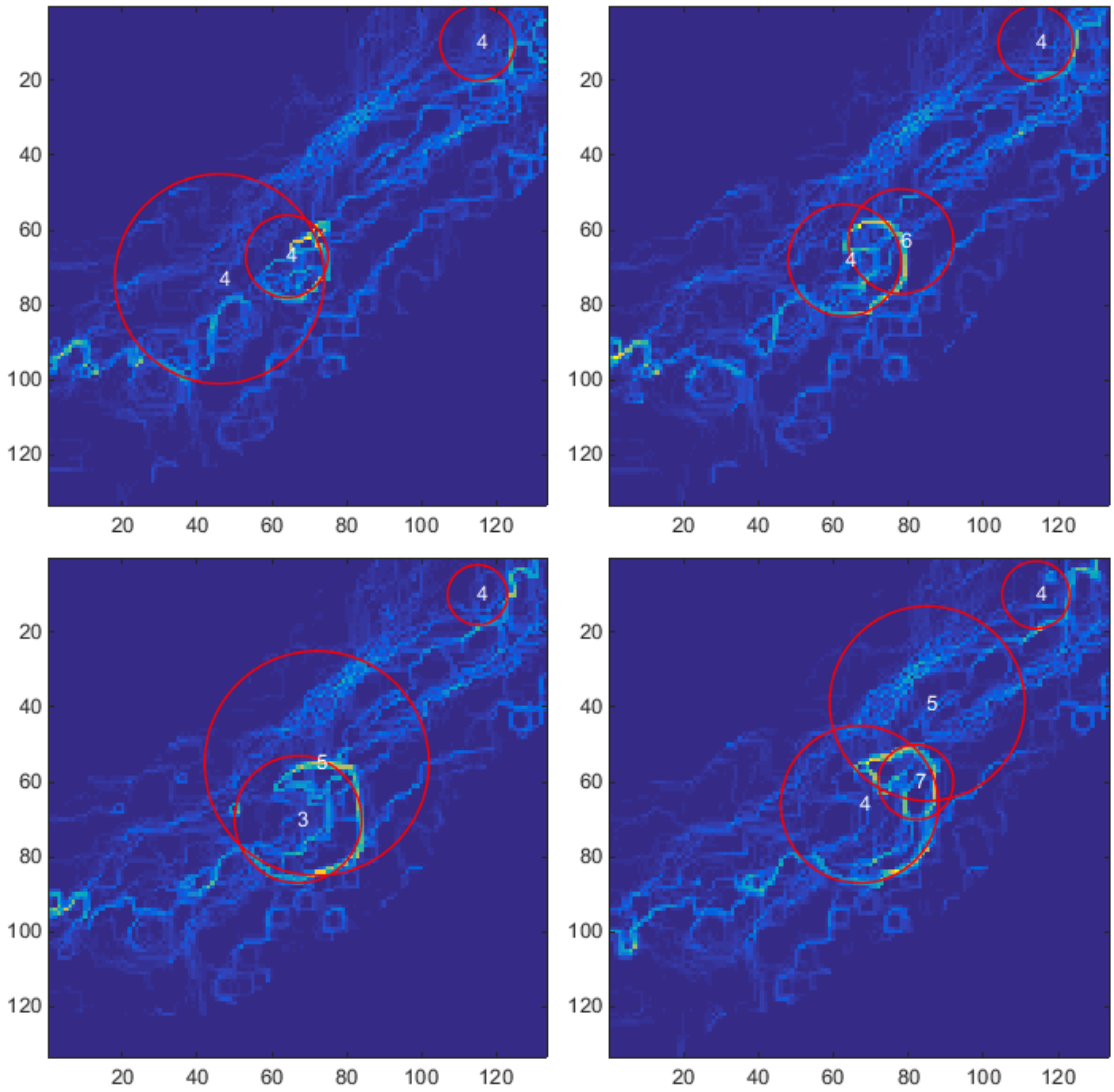

Abbildung 6.4: Die gleichen Frames wie in Abb. 6.3. Durch Auslassen der Normierung in Gleichung (5.4) werden passendere Halbkreise detektiert.

Für den Rest dieses Kapitels werden wir mit diesen Ergebnissen weiter arbeiten, um die Auswirkungen der anderen Verfahren zu zeigen.

Das erste Verfahren, dass nach der Detektion der Halbkreise in unserem Workflow ausgeführt wird, ist die Aufsummierung der Intensitäten (Kapitel 5.3). Es wird also für jeden gefundenen Halbkreis überprüft, ob in diesem Frame ein Maximum an Intensitäten in der Umgebung des Mittelpunktes vorliegt. Ist dies nicht der Fall, wird der potenzielle JAIL aus der Liste gelöscht. Abbildung 6.5 zeigt das Ergebnis nach Ausführung dieses Verfahrens. Wir sehen hier, dass dadurch in jedem Frame ein falsch positiver JAIL gelöscht wird.

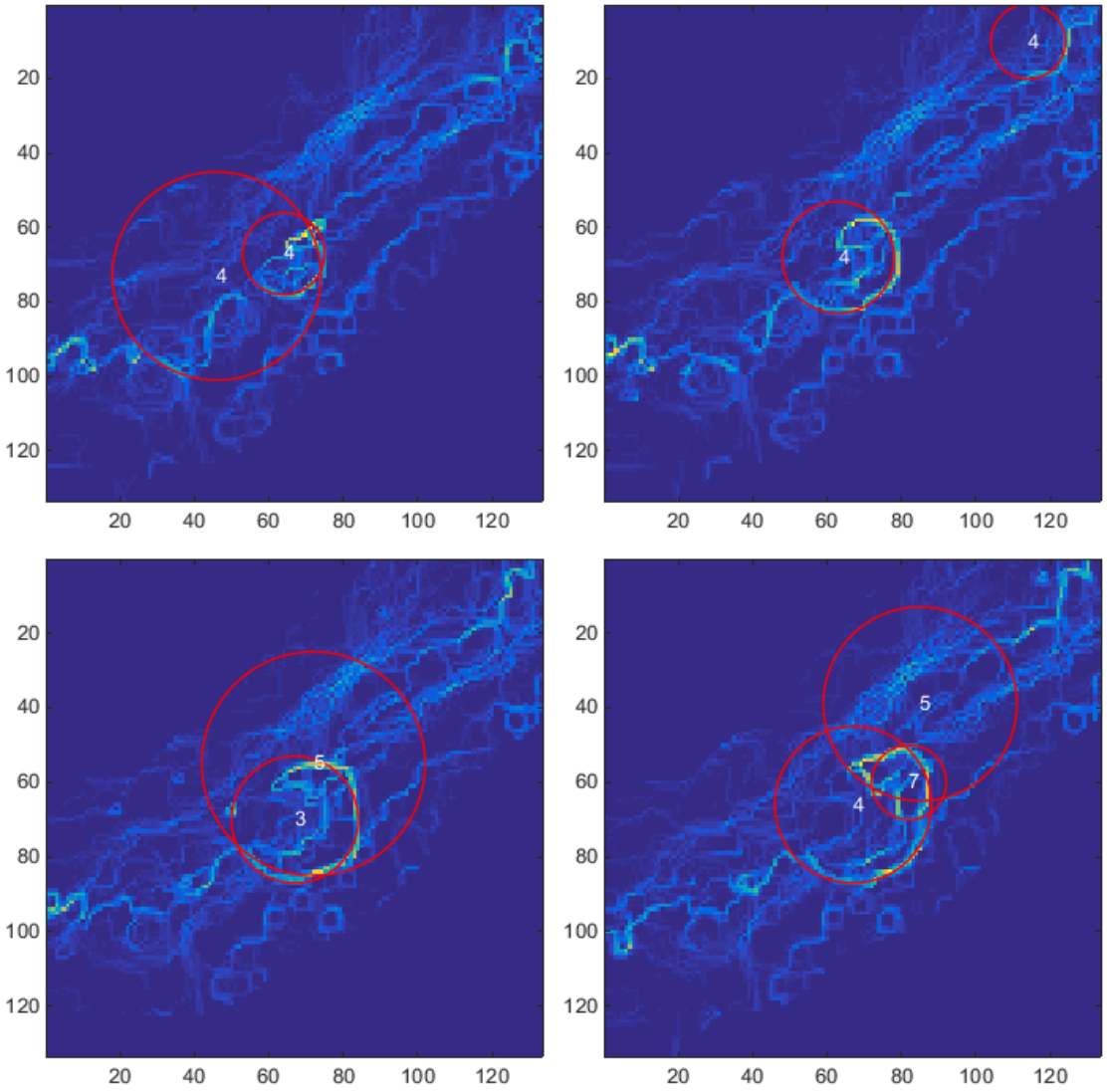

Abbildung 6.5: Ergebnis nach Detektion der Halbkreise und Überprüfung der Intensitäten

Das nächste Verfahren zur Aussortierung falsch erkannter JAIL ist der Abgleich mit den Connected Components. Wie in Kapitel 5.4 erklärt, wird hier überprüft, ob in der unmittelbaren Nähe eines JAIL eine Zusammenhangskomponente mit hohen Intensitätswerten und einer minimalen Anzahl an Pixeln exisitert. Wir sehen in Abbildung 6.6, dass dadurch weitere falsch detektierte JAIL entfernt werden.

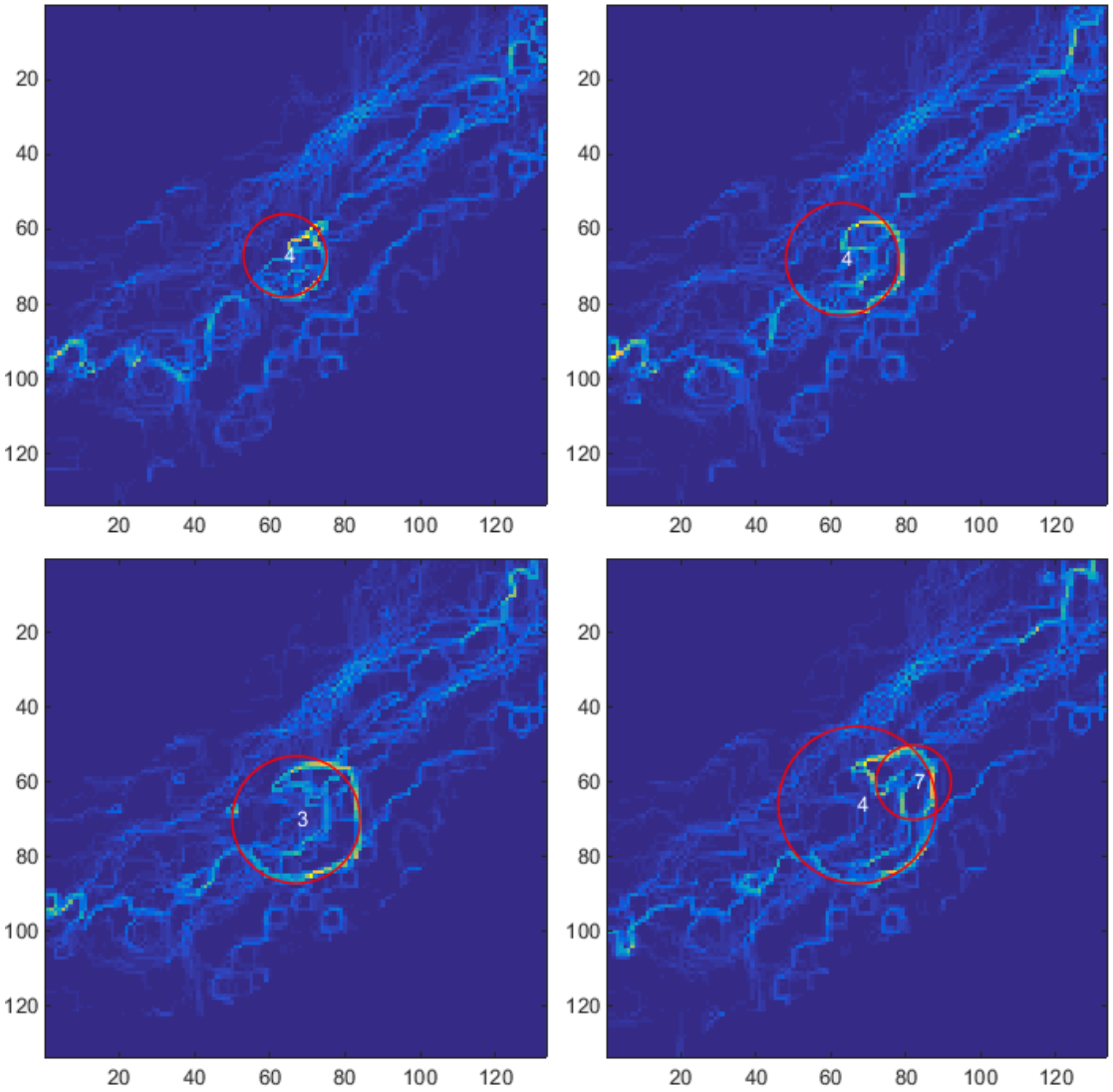

Abbildung 6.6: Ergebnis nach Detektion der Halbkreise, Überprüfung der Intensitäten und Abgleich mit den Connected Components.

Das dritte und damit letzte angewendete Verfahren in unserem Workflow ist der Vergleich mit der Optical-Flow-Norm, welches wir in Kapitel 5.2 kennen gelernt haben. In dem oben gezeigten Beispiel ändert sich hierbei jedoch nichts mehr und der Halbkreis mit Richtungsnummer 7 im vierten Frame bleibt als einziger falsch positiver JAIL in diesem Beispiel bestehen. Dies ist auf die Form des JAIL zu diesem Zeitpunkt zurückzuführen, die in dessen rechten oberen Ecke eine zusätzliche Ausbeulung in Form eines kleinen Halbkreises bildet.

7 Fazit und Ausblick

Das Ziel dieser Arbeit war die automatische Detektion von JAIL in Mikroskopie-Videos zu realisieren. Die Umsetzung folgte in Matlab; ein bewährtes Hilfsprogramm dazu war Icy.
Bevor wir im Verlauf der Arbeit einige getestete und benutzte Verfahren zur Detektion der JAIL gesehen haben, haben wir zunächst den biologischen Hintergrund der JAIL kennengelernt. Dies beinhaltete neben einem Überblick über den Zusammenhang zwischen JAIL und Endothelzellen auch die Darstellung der Bedeutsamkeit von JAIL für die Biologie.
Weiterhin haben wir einen Einblick in die mathematischen Grundlagen der Bildverarbeitung bekommen. Nach der mathematischen Definition eines Bildes wurden hier die Grundlagen der wichtigsten Verfahren für die Bearbeitung eines Bildes vorgestellt.
Das nächste Kapitel „Vorverarbeitung“ beinhaltete die drei ersten Schritte, die wir zur Verarbeitung der JAIL-Videos benutzt haben. Diese waren die Extraktion des p20-Kanals, die Entrauschung der Daten mittels Kullback-Leibler-Datenterm und TV-Regularisierung sowie die Segmentierung zur Eingrenzung der möglichen JAIL-Koordinaten in einer Umgebung der Zellwände.
Weiterhin haben wir im nächsten Kapitel die verschiedenen Verfahren zur JAIL-Detektion aufgeführt, die den Hauptteil der Programmierarbeit ausgemacht haben. Neben der Detektion von kreis- und halbkreisförmigen Strukturen waren dies der optische Fluss zwischen jeweils zwei Frames, die Aufsummierung der Intensitäten in einer Umgebung eines JAIL, die Verwendung der Connected Components sowie die Akkumulation der Faltungswerte. Diese wurden benutzt, um diejenigen JAIL zu löschen, die durch die Detektions-Verfahren möglicherweise als falsch positiv erkannt wurden.
Im letzten Kapitel haben wir dann noch den schließlich verwendeten Workflow kennengelernt, der aus den Verfahren der vorherigen Kapitel zusammengesetzt wurde.

Wie schon in Kapitel 6 beschrieben, ist das endgültige Ergebnis aus verschiedenen Gründen noch nicht ganz wie erwartet. Neben dem grundlegenden Problem der unterschiedlichen Formen und Größen der JAIL bereitet vor allem die angesprochene Normierung in Gleichung (5.4) Probleme im Hinblick auf die Detektion. Der

Versuch, trotz dieser Normierung die richtigen Halbkreise als JAIL zu erkennen, ohne kleine Kreise den größeren und passenderen vorzuziehen, wäre ein möglicher Ansatzpunkt für zukünftige Forschungen. Hier ist auch die Standardabweichung für die Gauss-Glättung der Faltungskerne zu nennen, die mit $\sigma = \sqrt{r}$ eventuell noch nicht ganz optimal gewählt worden ist.
Eine Weiterführung und Optimierung der in Kapitel 5.5 angeführten Idee von einer Akkumulation der Faltungswerte zur Zeitstetigkeit detektierter JAIL stellt einen weiteren Ansatzpunkt zur zusätzlichen Erforschung dar. Hier könnte man eventuell mit kleineren Ausschnitten des Videos arbeiten, um durch Aufsummierung der Werte über weniger Frames einen zu homogenen Akkumulator zu vermeiden.
Auf Dauer problematisch ist sicherlich auch die hohe Laufzeit des gesamten Algorithmus. Der Versuch, diese auch für große Ausschnitte eines Videos effektiver zu gestalten, würde die Testphasen enorm erleichtern.
Nicht zuletzt wäre es nach diesen Schritten weiterhin sinnvoll, noch weitere Videos als das von uns betrachtete hinzuzunehmen und mit dem Algorithmus zu testen.

Literaturverzeichnis

[AK] Atherton, T.J. & Kerbyson, D.J.: *Size invariant circle detection*, 1999.

[BB] Burger, Wilhelm & Burge, Mark James: *Digitale Bildverarbeitung: Eine Einführung mit Java und ImageJ*, Springer, 2005/2006.

[BD] Bredies, Kristian & Lorenz, Dirk: *Mathematische Bildverarbeitung: Einführung in Grundlagen und moderne Theorie*, Springer Vieweg, 2010.

[BK] Beierle, C. & Kern-Isberner, G.: *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen, 5. Auflage*, Springer Vieweg, 2014.

[BP] Becker, Florian & Petra, Stefania & Schnörr, Christoph: *Optical Flow*, Handbook of Mathematical Methods in Imaging, Seiten 1945-2004, 2015.

[BT] Bommas, Ulrike & Teubner, Philipp & Voss, Rainer: *Kurzlehrbuch Anatomie und Embryologie*, Thieme, 2005.

[CC] Caselles, V. & Chambolle, A. & Novaga, M.: *The Discontinuity Set of Solutions of the TV Denoising Problem and Some Extensions*, Society for Industrial and Applied Mathematics, 2007.

[CP] Chambolle, Antonin & Pock, Thomas: *A first-order primal-dual algorithm for convex problems with applications to imaging*, Journal of Mathematical Imaging and Vision, Seiten 120-145, 2011.

[Da] Davies, E.R.: *Machine Vision: Theory, Algorithms, Practicalities*, Elsevier, 2005.

[DB] Dirks, Hendrik & Burger, Martin & Frerking, Lena: *On Optical Flow Models for Variational Motion Estimation*, 2015.

[Di] Dirks, Hendrik: *Variational Methods for Joint Motion Estimation and Image Reconstruction*, Doktorarbeit, 2015.

[GG] Geman, S. & Geman, D.: *Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Seiten 721-741, 1984.

[J] Jiang, Xiaoyi: *Folien zur Vorlesung Computer Vision und Mustererkennung an der WWU Münster*, URL: http://cvpr.uni-muenster.de/teaching/ws12/ComputerVisionMustererkennungWS12/script/CVME-07-Segmentation-Hough.pdf, 2012 (zugegriffen im Oktober 2016).

[KA] Kornprobst, Pierre & Aubert, Gilles: *Mathematical problems in image processing: partial differential equations and the calculus of variations*, Springer, 2006.

[KŽ] Klette, Reinhard & Žunić, Jovisa: *Combinatorial Image Analysis*, Springer, 2004.

[Rh] Rhody, Harvey: *Hough Circle Transform*, Chester F. Carlson Center for Imaging Science, 2005.

[Ri] Rieder, Andreas: *Keine Probleme mit Inversen Problemen: Eine Einführung in ihre stabile Lösung*, Vieweg, 2003.

[Ro] Rockafellar, R. T.: *Convex Analysis*, Princeton University Press, 2. Auflage, 1972.

[SB] Sawatzky, A. & Brune, C. & Koesters, T. & Wübbeling, F. & Burger, M.: *EM-TV Methods for Inverse Problems with Poisson Noise.* Level Set and PDE Based Reconstruction Methods in Imaging, Seiten 71-142, 2013.

[SR] Süsse, Herbert & Rodner, Erik: *Bildverarbeitung und Objekterkennung: Computer Vision in Industrie und Medizin*, Springer Vieweg, 2014.

[ST] Seebach, Jochen & Taha, Abdallah Abu & Lenk, Janine et al.: *The CellBorderTracker, a novel tool to quantitatively analyze spatiotemporal endothelial junction dynamics at the subcellular level*, Histochemistry and Cell Biology, Seiten 517-532, 2015.

[T] Thormählen, Thorsten: *Folien zur Vorlesung Multimediakommunikation an der Philipps-Universität Marburg*,
URL: http://www.mathematik.uni-marburg.de/~thormae/lectures/mmk/mmk_6_1_ger_web.html#7, 2015 (zugegriffen im Oktober 2016).

[TS] Taha, Abdallah Abu & Taha, Muna & Seebach, Jochen & Schnittler, Hans-Joachim: *ARP2/3-mediated junction-associated lamellipodia control VE-cadherin-based cell junction dynamics and maintain monolayer integrity*, Institute of Anatomy and Vascular Biology, 2013.

[We] Werner, Dirk: *Einführung in die höhere Analysis*, Springer, 2006.

[Wi] Wirth, Benedikt: *Folien zur Vorlesung Optimierung an der WWU Münster*, URL: http://wwwmath.uni-muenster.de/num/Vorlesungen/Optimierung_WS14/material/notes.pdf, 2014 (zugegriffen im Oktober 2016).

[YP] Yuen, H.K. & Princen, J & Illingworth, J & Kittler, J.: *Comparative study of Hough transform methods for circle finding*, Elsevier, 1990.

Eidesstattliche Erklärung

Hiermit versichere ich, Stefan Nückel, dass die vorliegende Arbeit

Automatische Detektion von junction-associated intermittent lammelipodia (JAIL) in Mikroskopie - Videos von Endothelzellen

selbständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken - auch elektronische Medien - dem Wortlaut oder Sinn nach entnommen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind.

(Datum, Unterschrift)

Ich erkläre mich mit einem Abgleich der Arbeit mit anderen Texten zwecks Auffindung von Übereinstimmungen sowie mit einer zu diesem Zweck vorzunehmenden Speicherung der Arbeit in eine Datenbank einverstanden.

(Datum, Unterschrift)