# Entwicklung von Auswertemethoden zur Bestimmung von Proteinkinetiken nach Zellbestrahlungen am Rasterionenmikroskop SNAKE

*Studienarbeit*
*von*
*Tino Brüning*

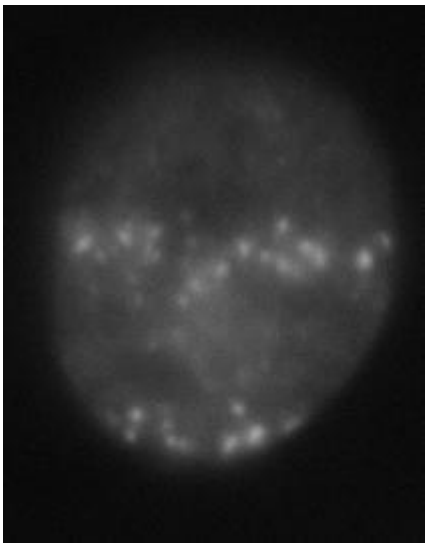
*Institut für angewandte Physik und Messtechnik*
*Universität der Bundeswehr München*

## Einleitung:

In der modernen Strahlenbiologie werden die Auswirkungen ionisierender Strahlung auf Organismen und einzelne Zellen untersucht. Besonderes Augenmerk liegt hierbei auf der Fähigkeit, schädigenden Einflüssen zu widerstehen und entstandene Defekte zu beheben. Um diese Reparaturmechanismen besser zu verstehen, werden einzelne Zellen mit Schwerionen bestrahlt und anschließend unter einem Fluorezsenzmikroskop beobachtet. Ziel ist es zu erkennen, wie Zellen auf gezielt herbeigeführte Beschädigungen reagieren. An der Universität der Bundeswehr in München führt man deshalb einzelnen Zellen einen Schaden zu, indem man sie mit Hilfe des Rasterionenmikroskops SNAKE (Supraleitendes Nanoskop für angewandte kernphysikalische Experimente) mit Schwerionen beschießt. Dabei wird eine Strahlauflösung von ca. 0,5µm (FWHM) erreicht. Durch eine Einzelionenpräparation und das am 14MV Münchner Tandembeschleuniger verfügbare breite Spektrum an Ionensorten und Energien kann die den Zellen verabreichte Dosis präzise eingestellt werden.

Durch den Ionenbeschuss wird die Doppelhelix, die als Träger der Erbinformationen dient, in der Zelle aufgebrochen. Um diesen Doppelstrangbruch zu beheben, übernehmen verschiedene Proteine in der Zelle eine Reparaturfunktion. Die zeitliche Verlagerung (Kinetik) GFP-markierter Proteine innerhalb einer Zelle kann durch eine neu entwickelte Lebendzellmikroskopie direkt am SNAKE-Bestrahlungsplatz in Zeitserien unmittelbar nach der Bestrahlung aufgenommen werden. Ziel der Untersuchungen ist es, den Ort und das Zeitverhalten der Umlagerung der Proteine, die für die Regeneration der Zelle verantwortlich sind, zu erkennen.

Ziel der vorliegenden Arbeit war es, geeignete Methoden zur Auswertung von Proteinkinetiken aus den Mikroskopdaten zu entwickeln und auf eine Zeitserie des Proteins Mdc1, welches im Zusammenwirken mit anderen Proteinen wesentlich für die Doppelstrangbruchreparatur verantwortlich ist, anzuwenden.
Zur Auswertung der Aufnahmen muss nun ein Verfahren entwickelt werden, mit dessen Hilfe man eine Aussage über die Umverteilung der Proteine machen kann. Dafür muss zunächst der räumliche Bereich, innerhalb dessen sich Reparaturproteine um einen Schaden anlagern, ermittelt werden, um anschließend eine Messung der sich mit der Zeit ändernden Proteinintensität über eine Bildserie durchzuführen. Einen Anhalt für die Proteinkonzentration gibt die Helligkeit der Konzentrationszentren (Foci) in einem Bild. Zwischen Helligkeit und Konzentration wird von einem proportionalen Zusammenhang ausgegangen.



Einzelne Zelle: Dieses Bild zeigt einen bestrahlten Zellkern. Die hellen Punkte zeigen eine erhöhte Proteinkonzentration (Foci) an.

# Inhaltsverzeichnis

**Ziel der Studienarbeit:** ..... 5

Aufbau des Programms: ..... 7

Verfolgung der Zellbewegung: ..... 9

Die Translation: ..... 9

Die Rotation: ..... 11

Erstellung einer Schablone: ..... 12

Auswertung der Bilderserie: ..... 14

Benutzermenü: ..... 15

Durchgeführte Versuchsreihen und Ergebnisse: ..... 18

Auswertung der Ergebnisse: ..... 20

Anhang: ..... 21

Literatur: ..... 21

Quelltext: ..... 22

## Ziel der Studienarbeit:

Um den Verlauf der Reparaturvorgänge beschreiben zu können wird der Helligkeitsverlauf der Foci gemessen. Dazu muss der Wert der entsprechenden Pixel jedes einzelnen Bildes bestimmt werden. Da sich die Zellen im Laufe der Zeit innerhalb des Beobachtungsbereiches bewegen, wurde ein Programm entwickelt, das in der Lage ist, diese Bewegung zu erkennen. Auf diese Weise wird in jedem Bild der Zeitserie der Mittelpunkt der Zelle ermittelt. Er dient für alle weiteren Berechnungen als Bezugspunkt. Ist die Bewegung einer einzelnen Zelle einmal bekannt, so kann anhand dieser Werte zur Auswertung eine Schablone, die die Position der Foci innerhalb der Zelle wiederspiegelt, erstellt werden.

Diese Schablone ist das Abbild des Zellenausschnitts, welches jedoch nur zwei verschiedene Bildpunktwerte, schwarz und weiß, enthält. Ausgewertet werden später nur die Pixel eines Zellbildes, welche mit den weißen Pixeln der Schablone deckungsgleich sind. Die Schablone enthält also Informationen über die Lage und die Ausdehnung der Foci innerhalb der Zelle. Da sich die Schablone über den gesamten zeitlichen Verlauf einer einzelnen Zelle nicht verändert, wird die zu untersuchende Fläche konstant gehalten. Damit wird erreicht, dass die Anzahl der Pixel, die in die Auswertung einfließen, nicht verändert wird.

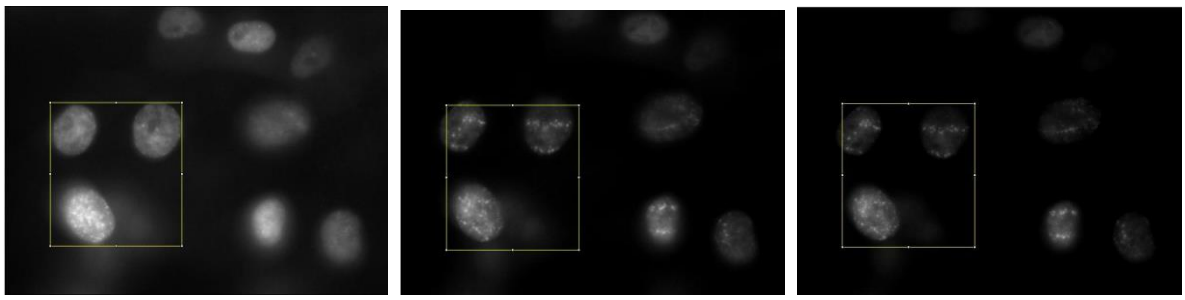
Als nächstes muss das Bleichen der Bilder kompensiert werden. Dieses ist dadurch bedingt, dass das Fluoreszenzprotein nicht unbegrenzt lange leuchten kann. Dies hat zur Folge, dass die Bilder nach und nach dunkler werden. Die messbare Helligkeit ist demzufolge geringer als die tatsächlich durch Umlagerung erzeugte. Der Einfluss dieses Effektes auf die Ergebnisse muss also ausgleichen werden.

Um die Verblassung der Bilder mit der Zeit zu kompensieren, wird neben den bestrahlten Bereichen auch ein unbestrahlter Zellkern beobachtet. Er gibt einen Anhaltspunkt dafür, wie schnell und mit welcher Intensität die Bilder an Helligkeit verlieren. Die Werte der bestrahlten und der unbestrahlten Zellkerne werden später in Relation gesetzt.

## Aufbau des Programms:

Für die Bearbeitung der Bildserien steht das Open-Source Programm ImageJ zur Verfügung. Bestandteil dieses Programms ist eine Makrosprache, mit deren Hilfe das nachfolgende Programm erstellt wurde.

Um einen Bereich einer Zelle untersuchen zu können, muss zunächst der Ort dieses Bereiches ermittelt werden. Da sich die Zellen im Laufe der Untersuchung bewegen, verändert sich auch die absolute Position der einzelnen Foci innerhalb des Bildausschnittes. Ihre Relativposition innerhalb einer Zelle bleibt hingegen näherungsweise konstant. Um nun global den Ort eines Focus zu bestimmen muss man den globalen Ort eines Bezugspunktes, hier der Mittelpunkt der Zelle, und die relative Lage des Focus zum Bezugspunkt, hier mit Hilfe der Schablone, bestimmen.



Bewegung der Zellen: Die drei Bilder einer Zeitserie zeigen die Bewegung der Zellen. Der gelbe Auswahlrahmen hat immer die gleiche Position.

Die Bewegung der Zelle kann man in zwei wesentliche Bewegungsarten unterscheiden. Die erste ist die Translation. Die Aufgabe des Programms soll es also sein, die Position der Zelle auf den einzelnen Bildern zu erkennen. Hierbei wird davon ausgegangen, dass die Zellen nur kleine Bewegungen vollziehen.

Eine Vermischung der Zellen untereinender und das Wiedererkennen einer bestimmten Zelle soll nicht Aufgabe des Programms werden.
Die zweite Bewegungsart ist die Rotation der Zellen. Das Programm muss also in der Lage sein, Strukturen und Unterschiede selber zu erkennen, sie mit den Folgebildern zu vergleichen und auszuwerten.

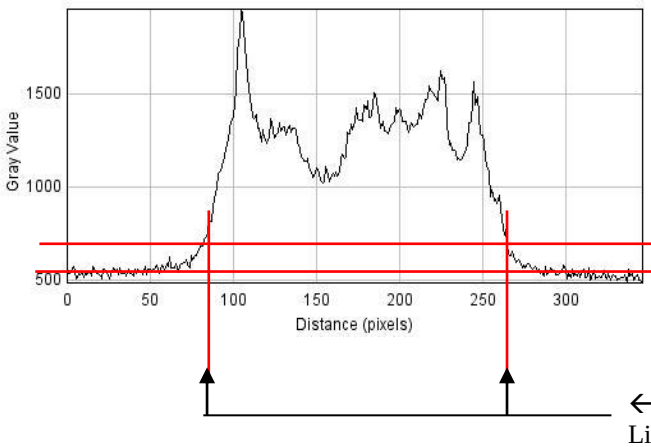
Nach der Ermittlung der Bewegung soll das Programm selbstständig einen Vorschlag für eine Schablone liefern. Ähnlich wie der Mensch orientiert es sich dabei an Veränderungen in der Zelle. Statische Strukturen, die irrtümlicher Weise als Foci erscheinen werden dadurch ausgeblendet. Veränderungen durch das Ausbleichen der Bilder oder das Bildrauschen müssen hierbei extra untersucht werden. Die vom Programm erstellte Schablone kann falls nötig im Nachhinein vom Benutzer noch editiert werden. Hierzu wird die in ImageJ bereits vorhandene Draw-Funktion verwendet. Der Benutzer kann damit ausgewählte Schablonenpunkte löschen oder hinzufügen.

Abschließend soll die Zelle anhand der gefertigten Schablone untersucht und das Ergebnis in einer Datei oder einer Exceltabelle zur weiteren Bearbeitung ausgegeben werden.
Die einzelnen oben genannten Schritte sollen im Folgenden genauer betrachtet werden.

## Verfolgung der Zellbewegung:

### Die Translation:

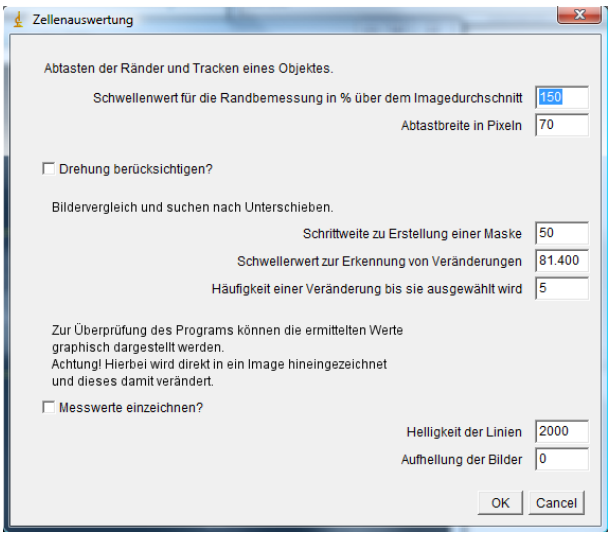
Um den Mittelpunkt eines Zellkerns zu bestimmen, kann man sich am Zellkernrand orientieren. Dieser ist durch einen Sprung in der Helligkeit vom Umgebungswert auf den Helligkeitswert der Zelle gekennzeichnet. Hat der Benutzer einmal im ImageJ einen rechteckigen Bereich um eine Zelle markiert, kann nun vom Rand dieser Markierung nach innen hin die Helligkeit ermittelt werden. Überschreitet die Helligkeit einen gewissen Schwellenwert, den der Benutzer am Anfang der Auswertung in relativer Abweichung von der Durchschnittshelligkeit eines Bildes selber festlegen kann, wird dieser Abstand zum Rand der Markierung als Zellenrandabstand gespeichert. Um eventuelles Verformen des Zellrandes und das natürliche Rauschen des Bildes zu unterdrücken, wird die Helligkeit nicht nur an einer Stelle ermittelt, sondern über einen gewissen Bereich gemittelt.




Durchschnitt durch die Zelle:
Dieses Bild zeigt den Helligkeitsverlauf entlang eines Schnitts durch eine Zelle. Die Werte unterhalb der Umgebungshelligkeit sind nicht mit aufgenommen.

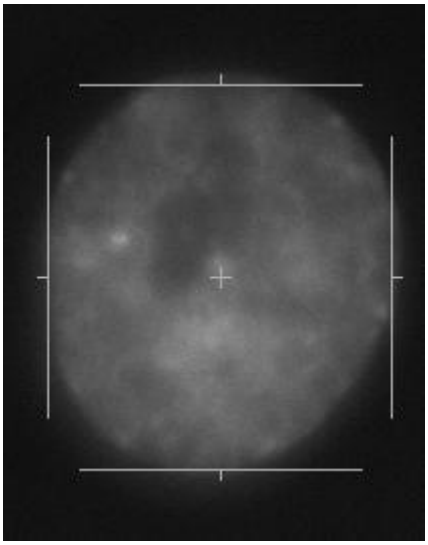
← Beispielwert: 50% heller als die Umgebung
← Durchschnittswert der Umgebung

← Schnitt des Helligkeitsverlaufes mit der 150% Linie, gleichzeitig festgelegter Zellenrand




Startmenü:

← Eingabe des Randwertes (Relativ zur Umgebung)
← Eingabe der Abtastbreite in Pixeln



Randberechnung: Hier ist der ermittelte Rand einer Zelle durch Linien dargestellt. Die Länge der Linien entspricht dem Bereich des Abtastens.

Um den Mittelpunkt der Zelle zu bestimmen, werden jeweils paarweise der obere und der untere, sowie der linke und der rechte Rand vermessen. Der Mittelpunkt wird dann als Punkt aus jeweils halber Strecke zwischen den Rändern festgelegt.
Die Koordinaten der Zelle werden dann in einer Matrix gespeichert und der Mittelpunkt der vom Benutzer vorgenommenen Markierung wird für das nachfolgende Bild in den berechneten Mittelpunkt gelegt. Auf diese Weise wird die Bewegung der Zelle getrackt.

### Die Rotation:

Um eine Drehung zu erkennen, gibt es in einer Zelle keine so signifikante Struktur, wie den Zellkernrand, an der man sich orientieren könnte. Aus diesem Grund wird der Helligkeitsverlauf eines Ringes um den berechneten Mittelpunkt gespeichert. Durch Drehung dieses Ringes kann der Helligkeitsverlauf mit der Zelle des nachfolgenden Bildes verglichen werden. Das Programm wählt dabei aus, bei welcher Gradzahl sich die Verläufe am ähnlichsten sehen. Die Veränderung der Drehlage wird für die spätere Bearbeitung in einer Matrix gespeichert.

## Erstellung einer Schablone:

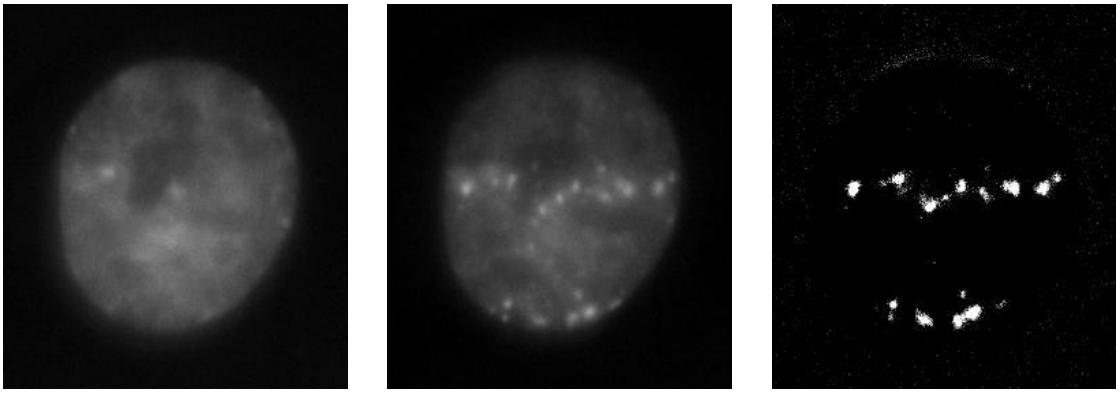
Die Schablone ist ein Bild, welches exakt die Ausmaße vom Auswahlrahmen besitzt, die der Benutzer beim Start des Programms um eine Zelle gezogen hat. Sie dient dazu, auszuwählen welche Pixel eines Bildes von einer Zelle ausgewertet werden sollen und welche nicht. Dabei gelten für alle Bilder einer Zeitserie die gleichen Voraussetzungen. In der Schablone kommen nur zwei Pixelwerte vor. 0 steht dafür, dass der Bildpunkt, der sich an den gleichen Koordinaten auf der Bildserie befindet nicht in die Betrachtungen mit einbezogen werden. Hat der Schablonenbildpunkt einen Wert von 255 (Weiß), so wird dieser Punkt zur weiteren Betrachtung herangezogen.

Auf den oben gewonnenen Daten basierend kann das Programm nun diese Schablone erstellen.

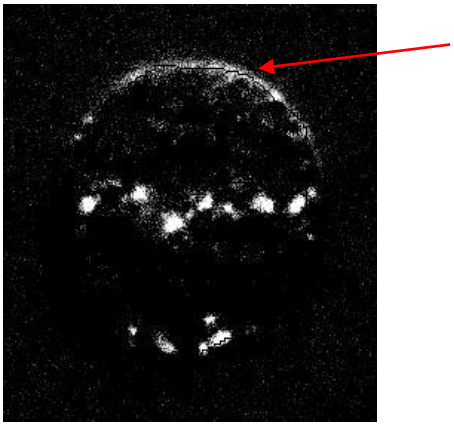
Um nun zu entscheiden, welche Pixel in die Auswahl hineinkommen und welche nicht, vergleicht das Programm zwei Bilder miteinander und registriert die Abweichungen voneinander. Bereiche, die sich verändern, kommen in die Auswahl, Bereiche, die gleich sind, werden nicht in die Auswahl mit hinein genommen. Da die Veränderung einer Zelle zwischen zwei aufeinanderfolgenden Aufnahmen einer Zeitserie oft geringer ist als das Bildrauschen, werden nicht nachfolgende Bilder miteinander verglichen, sondern Bilder mit größerem zeitlichem Abstand. Dies gewährleistet eine deutlich feststellbare Veränderung der Zellen, die sich gegenüber dem Rauschen abheben kann. Die Schwelle des Rauschens und die Schrittweite der Vergleichsmessung können beim Start des Programms durch den Benutzer individuell festgelegt werden. Um eventuelle Störungen, die höher sind als die vom Benutzer festgelegte Schwelle und nur singulär auftreten, auszublenden, kann man hier

zusätzlich noch angeben, wie oft eine Veränderung der Bilder auftreten muss, damit sie in die Schablone eingetragen wird.

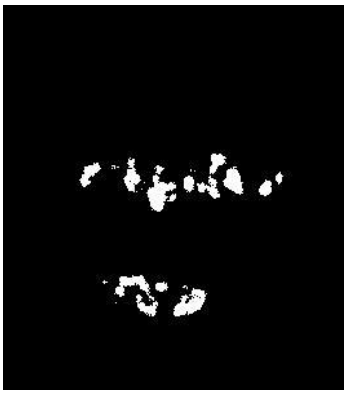
Nach dem erstellen der Schablone hat der Benutzer die Option diese von Hand nachträglich zu bearbeiten. Dies bietet ihm die Möglichkeit unerwünschte Artefakte, die das Programm nicht herausgefiltert hat, zu korrigieren oder nicht beachtete Bereiche nachträglich mit in die Betrachtungen einfließen zu lassen.



Beispiel: Links das erste Bild, Mitte das zwanzigste Bild einer Zeitserie. Beide Zellen sind getrackt und in der Rotation korrigiert. Rechts ist die Subtraktion des ersten Bildes vom zwanzigsten. Nur die Veränderungen sind noch zu erkennen. Sind diese hell genug und kommen mehr als einmal vor, werden sie in die Schablone (gleicher Bildausschnitt) aufgenommen.



Zellvergleich mit Fehlern: Aus der Gleichen Zeitserie wie oben (hier Subtraktion Bild 3 mit 22). Am oberen Rand der Zelle tritt ein Störbereich auf. Da dieser nur einmal in der Zeitserie vorhanden war wird dieser Bereich nicht berücksichtigt.



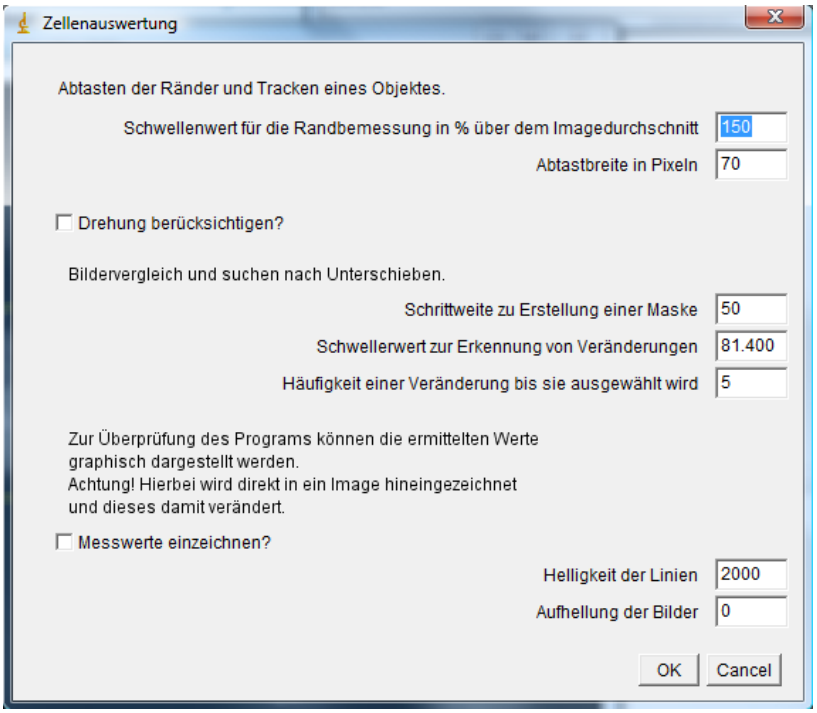
Aus er Zeitserie entstandene Schablone: Dieses Bild enthält nur schwarze und weiße Pixel. Die weißen Pixel befinden sich an den Orten, an denen sich die Zelle verändert hat.

## Auswertung der Bilderserie:

Um nun zu der eigentlichen Auswertung der Bilder zu kommen werden diejenigen Bildpunkte, die die gleichen Koordinaten im Bezug zum Mittelpunkt und zur Drehung wie die Bildpunkte der Schablone mit dem Wert 255 haben ausgelesen. Dabei wird der Mittelpunkt der Schablone immer auf den berechneten Mittelpunkt der Zelle gelegt. Ihre Helligkeitswerte werden zusammenaddiert und durch die Anzahl der ausgewählten Pixel geteilt. Somit kann jedem Bild in einer Zeitserie der Helligkeitswert der Foci zugeordnet werden. Diese Werte werden in einer Resultattabelle ausgegeben und können vom Benutzer weiterverarbeitet werden.

## Benutzermenü:

Das Benutzermenü bietet dem Bediener des Programms die Möglichkeit, aktiv in die Berechnungen einzugreifen. Je nach Qualität und Beschaffenheit der Bilder bedarf es hier einer Möglichkeit, Randbedingungen zu verändern.




Benutzermenü:

← 1.
← 2.
← 3.

← 4.
← 5.
← 6.

← 7.

1. Randschwellenwert: Hier wird der relative Wert, der als Grenze für die Randbestimmung ermittelt wird, festgelegt. Als Referenz wird die durchschnittliche Helligkeit des gesamten Bildes genommen. Diese wird als 100% für jedes Bild einer Zeitserie aufs Neue festgelegt. Das zu trackende Objekt muss somit heller als die Umgebung sein. Beim Start des Programms wir hier ein Relativwert von 150% vorgeschlagen.
2. Abtastbreite: Bei abweichender Form der Zelle kann hier der Bereich, in dem der Zellrand abgetastet wird, verändert werden. Als Startvorschlag werden 2/3 der Rahmenbreite genommen.
3. Drehung: Da nicht bei allen Zellen eine Rotationsbewegung ermittelt werden kann, da sie zum Beispiel keinerlei Konturen aufweisen oder Unscharf abgebildet sein können, hat man hier die Möglichkeit bei kleinen Drehbewegungen den Rotationsausgleich abzuschalten. Bei einigen Zeitserien war dieser aufgrund nicht vorhandener oder nur geringer Drehbewegungen nicht nötig. Ein Abschalten erhöht die Berechnungsgeschwindigkeit.
4. Schrittweite: Hier kann der Benutzer den Abstand zweier Bilder innerhalb der Zeitserie angeben, bei dem die Bilder verglichen werden sollen. Eine Schrittweite von 20 bedeutet zum Beispiel, dass in der Zeitserie Bild 1 mit 21, 2 mit 22, 3 mit 23 usw. verglichen werden. Als Startwert wird die Hälfte der vorhandenen Bilder einer Serie angegeben, da es hier ein Gleichgewicht zwischen Schrittweite und Anzahl der auswertbaren Bilder gibt. Es empfiehlt sich hier eine Schrittweite zu nehmen, bei der grob die größten Veränderungen zum Anfangsbild auftreten.
5. Schablonenschwellenwert: Damit nicht jede Veränderung der Bilder in die Schablone aufgenommen wird, kann hier ein Grenzwert festgelegt werden, ab dem eine Veränderung registriert werden soll. Hiermit kann

man das natürliche Rauschen der Bilder ausblenden. Beim Start wird ein Wert vorgeschlagen, der dem natürlichen Rauschen des Hintergrunds entspricht.

6. Häufigkeit einer Veränderung: In der Regel kommen nicht erwünschte Störungen nur einmal an einem Ort in der Zeitserie vor. Sind diese Störungen jedoch sehr zahlreich, kann es vorkommen dass statistisch ein Ort mehrmals betroffen ist. Um immer wiederkehrende, nicht statische Veränderungen auszublenden, kann man hier angeben, wie häufig eine Veränderung auftreten soll. Als Startwert werden 5% der Gesamtbilderzahl genommen.
7. Zeichenfunktion: Um zu überprüfen wo das Programm Ränder erkennt und wie stark es ein Bild verdreht kann man sich die Begrenzungen einzeichnen lassen. Hierbei trägt das Programm seine Berechneten werte direkt in die Bilderserie ein. Der Benutzer hat die Möglichkeit die Helligkeit der Linien und den Kontrast der Bilder individuell einzustellen.

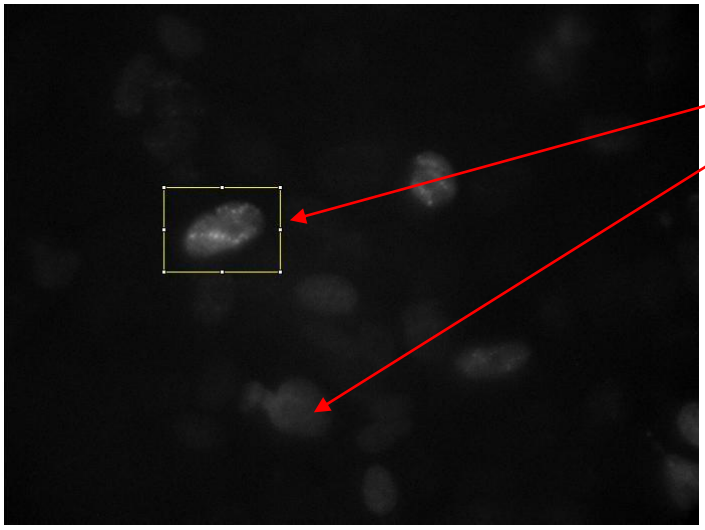
## Durchgeführte Versuchsreihen und Ergebnisse:

Um die Funktionsweise des Programms zu zeigen, standen zwei Bildserien zur Verfügung. Bei der Auswertung der ersten Probe (Probe 4, Position 2 der März/April Strahlzeit diesen Jahres) stellte sich heraus, dass sich die Aufnahmeschärfe des Bildes zyklisch veränderte. Dies hatte zur Folge, dass bei unschärferen Bildern die geringe räumliche Ausdehnung der Foci dazu geführt hat, dass diese nicht mehr erfasst wurden. Um dies zu kompensieren wurde nur etwa jedes dritte Bild der Zeitserie verwendet.

Aus vorangegangenen Experimenten ist bekannt, dass die Gesamthelligkeit mit der Zeit abnimmt. Um dies zu kompensieren, wurden die Werte der zu messenden Zellen mit denen einer unbestrahlten Zelle verglichen.

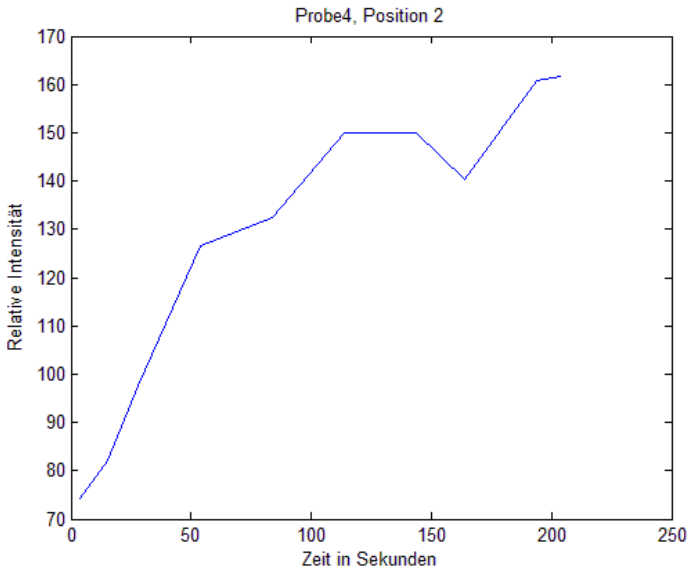
$$I_{Foci} = I_{Foci-Serie} \cdot \frac{I_{Ref\_0}}{I_{Ref-Serie}} - I_{Foci\_0}$$

$I_{Foci\_0}$ = Helligkeitswert der Foci im ersten Bild
$I_{Foci-Serie}$ = Helligkeitswert der Foci im Serienbild
$I_{Ref\_0}$ = Helligkeit der Referenzzelle im ersten Bild
$I_{Ref-Serie}$ = Helligkeit der Referenzzelle im Serienbild

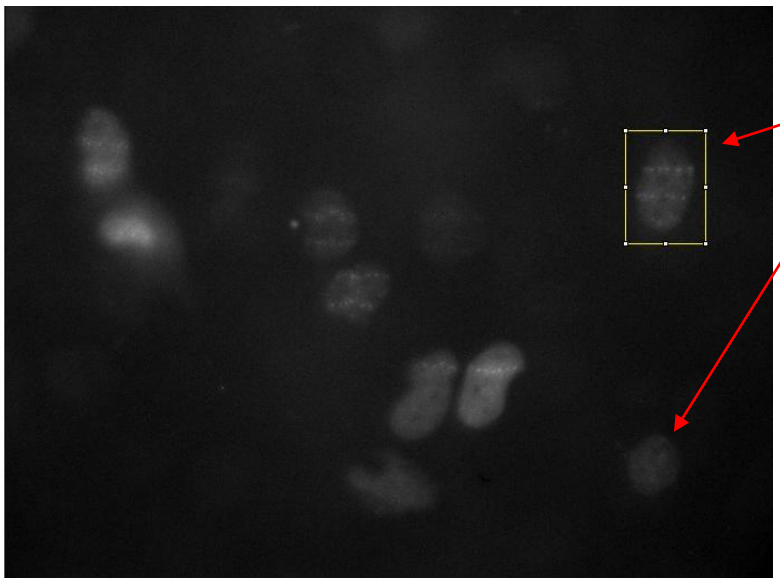


Probe 4 Position 2:
Ausgewertete Zelle
Referenzzelle

Ausgewertet wurden hier 819 Pixel im Bereich der Zellfoci

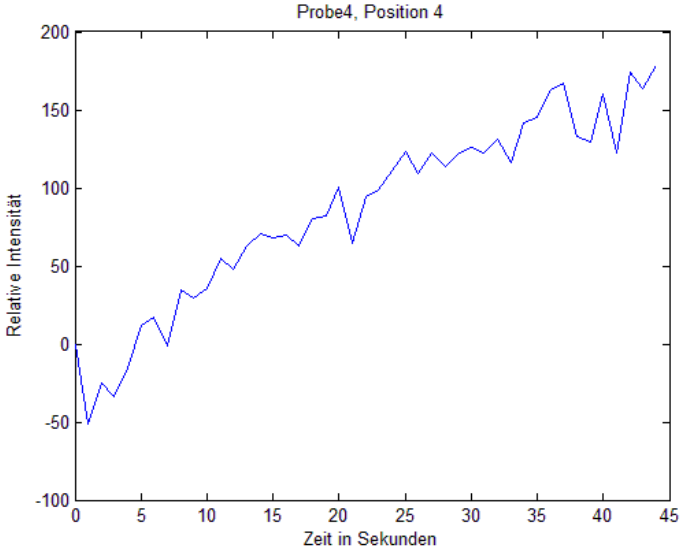



Zeitlicher Verlauf der Helligkeit Probe 4 Position 2



Probe 4 Position 4:
Ausgewertete Zelle
Referenzzelle

Ausgewertet wurden hier 433 Pixel im Bereich der Zellfoci



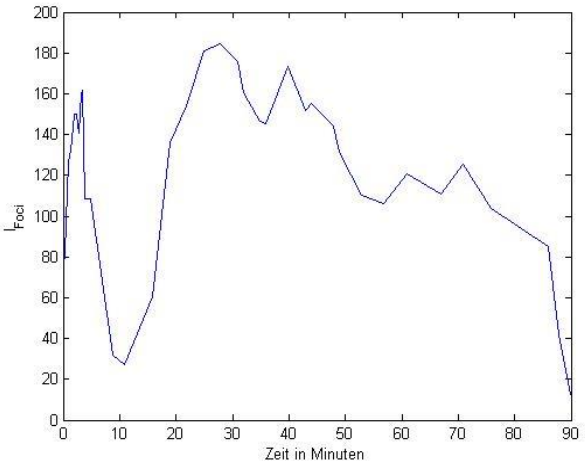

Zeitlicher Verlauf der Helligkeit Probe 4 Position 4

## Auswertung der Ergebnisse:

Beide Auswertungen spiegeln den charakteristischen Anstieg der Proteinkonzentration und somit des Helligkeitsanstieges der Bilder wieder. Trotz der recht hohen Anzahl der ausgewerteten Pixel schwanken die Werte immer noch um etwa 20 Helligkeitspunkte.

Um das Auswertungsverfahren weiter zu verbessern, werden in weiteren Arbeiten die Genauigkeit des Trackingverfahrens sowie die Verfahren zur Erstellung der Schablone verbessert.

In weiterführenden Arbeiten soll das Verfahren so weit verbessert werden, so dass damit eine ganze Zeitserie ausgewertet werden kann. Zurzeit stellen länger laufende Experimente noch ein Problem dar, da hier die Qualität und Intensität der Bilder nachlässt. Ziel solle es sein, eine möglichst genaue Aussage über den Helligkeitsverlauf der Foci zu geben.




Dargestellt ist hier der gesamte zeitliche Verlauf einer Zeitserie. Die ersten 4 Minuten können gut erfasst werden. Sie stellen jedoch nur den Bereich des Aufbauens der Foci dar. Der überwiegende, abbauende Teil ist noch sehr schlecht erfasst. Dieses Problem wird in der nächsten Arbeit behandelt.

# Anhang:

## Literatur:

[1] **Volker Hable** 2004. Untersuchung der Dynamik von DNA-Reparationen nach Bestrahlung lebender Zellen am Rasterionenmikroskop SNAKE
Diplomarbeit, Technische Universität München

[2] **ImageJ Macro Language**
http://rsb.info.nih.gov/ij/developer/macro/macros.html

[3] **Built-in Macro Functions**
http://rsb.info.nih.gov/ij/developer/macro/functions.html

[4] **Index of /ij/ macros**
http://rsb.info.nih.gov/ij/macros/

## Quelltext:

```
current_add=1.5                    // Schwellenwert der Zellenränder über dem Durchschnitt
thickness=50                       // Abtastbreite in Pixeln (halber Wert)
ink=0                              // Einzeichnen der Messungen?
brightness=2000                    // Helligkeit der Markierers
Step=18                            // Schrittweite zur Erstellung einer Maske
AufH=0                                    // Kontrastlevel zur Aufhellung der Maske
Mask_threshold=50                  // Schwellenwert für die Makse
incidence=2                        // Häufigkeit der Veränderung bis sie als Punkt in die
Maske ausgewählt wird
execute_rotation=1                 //Drehung berechnen


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

////////////////////////////////////
// Dialogfeld erstellen //
////////////////////////////////////


                        //Berechnen optimaler Werte als Vorgabe
        getSelectionBounds(X, Y, width, height);
        if(selectionType()!=0){
                showMessage("Kein Ziel Ausgewählt");
                exit;
        }

if(1==1){
        thickness=floor(minOf(width,height)/3)      ;
        Step=nSlices/2;

        rustling=newArray(8);
        for(n=1;n<=nSlices;n++){
                setSlice(n);
                rustling[0]=maxOf((abs (rustling[4]-getPixel(X,Y))),rustling[0]);
                        rustling[4]=getPixel(X,Y);
                rustling[1]=maxOf((abs (rustling[5]-getPixel(X+width,Y))),rustling[1]);
                        rustling[5]=getPixel(X+width,Y);
                rustling[2]=maxOf((abs (rustling[6]-getPixel(X,Y+height))),rustling[2]);
                        rustling[6]=getPixel(X,Y+height);
               rustling[3]=maxOf((abs  (rustling[7]-getPixel(X+width,Y+height))),rustling[3]);
                        rustling[7]=getPixel(X+width,Y+height);

                if (n==1){
                        rustling[0]=0;
```

```
                        rustling[1]=0;
                        rustling[2]=0;
                        rustling[3]=0;
                }

        }

        Mask_threshold=rustling[0];
                if (rustling[1]<Mask_threshold) {
                        Mask_threshold= rustling[1];
                }
                if (rustling[2]<Mask_threshold) {
                        Mask_threshold= rustling[2];
                }
                if (rustling[3]<Mask_threshold) {
                        Mask_threshold= rustling[3];
                }
        Mask_threshold=Mask_threshold*1.1;

        setSlice(1);

        incidence=maxOf(2,round(nSlices/20));

}




do{

                        //Dialogbox erstellen
        Dialog.create("Zellenauswertung");

        Dialog.addMessage("Abtasten der Ränder und Tracken eines Objektes.");
        Dialog.addNumber("                Schwellenwert für die Randbemessung in % über
dem Imagedurchschnitt",current_add*100);
        Dialog.addNumber("Abtastbreite in Pixeln",thickness);
        Dialog.addCheckbox("Drehung berücksichtigen?",false);

        Dialog.addMessage("\n Bildervergleich und suchen nach Unterschieben.");
        Dialog.addNumber("Schrittweite zu Erstellung einer Maske",Step);
        Dialog.addNumber("Schwellerwert zur Erkennung von
Veränderungen",Mask_threshold);
        Dialog.addNumber("Häufigkeit einer Veränderung bis sie ausgewählt
wird",incidence);
```

```
	Dialog.addMessage("\n Zur Überprüfung des Programs können die ermittelten Werte \n graphisch dargestellt werden. \n Achtung! Hierbei wird direkt in ein Image hineingezeichnet \n und dieses damit verändert.");
	Dialog.addCheckbox("Messwerte einzeichnen?",false);
	Dialog.addNumber("Helligkeit der Linien",brightness);
	Dialog.addNumber("Aufhellung der Bilder",AufH);


	Dialog.show();


			// Werte erfassen
	current_add=Dialog.getNumber()/100;
	thickness=Dialog.getNumber();
	Step=Dialog.getNumber();
	Mask_threshold=Dialog.getNumber();
	incidence=Dialog.getNumber();
	brightness=Dialog.getNumber();
	AufH=Dialog.getNumber();
	execute_rotation=Dialog.getCheckbox();
	ink=Dialog.getCheckbox();


			// Werte überprüfen
all_items_correct=1;

	if(current_add<=1){
		showMessageWithCancel("Randschwellenwert zu gering");
		all_items_correct=0;
	}

	if(thickness<=0 || thickness >=minOf(width,height)){
		showMessageWithCancel("Abtastbreite außerhalb der Markierung");
		all_items_correct=0;
	}

	if(Step<=0 || Step >=nSlices-2){
		showMessageWithCancel("Schrittweite außerhalb der Parameter");
		all_items_correct=0;
	}

	if(incidence<=0 || incidence>nSlices){
		showMessageWithCancel("Unterschiedshäufigkeit außerhalb der Parameter");
		all_items_correct=0;
	}

} while(all_items_correct!=1);

print("\\Clear");
```

```
print("Analyse gestartet");



////////////////////////////////////////////////////////////////////////////
// Verrfolgen der Zelle über die einzelnen Bilder//
////////////////////////////////////////////////////////////////////////////


X_center=newArray(nSlices+1);
Y_center=newArray(nSlices+1);
rotation=newArray(nSlices+1);
circle_alt=newArray(361);


print("");

Image_1=getImageID();
for(n=1;n<=nSlices;n++){
setSlice(n);
getSelectionBounds(X, Y, width, height);

sum_alt=0;
count_alt=0;
ink_s=0;



                        // bestimmen des Helligkeitsdurchschnittes

for(x=0;x<width;x++){
        for(y=0;y<height;y++){
                sum_neu=sum_alt+getPixel(x,y);
                sum_alt=sum_neu;
                count_neu=count_alt+1;
                count_alt=count_neu;
        }
}
current=sum_neu/count_neu;



                        // linke Randbemessung
x_bounds_left=-1;
for(x=0;x<width;x++){
        sum_line_alt=0;
        for(i=0;i<(thickness*2);i++){
                sum_line_neu=sum_line_alt+getPixel(x+X,Y+height/2-thickness+i);
```

```
                sum_line_alt=sum_line_neu;
        }
        current_line_value=sum_line_neu/(thickness*2);

        if(current_line_value<current*current_add){

        }else{
                if(ink==1){
                setColor(brightness);
                drawLine(X+x,Y+height/2-thickness,X+x,Y+height/2+thickness);
                }
                x_bounds_left=x;
        x=width-1;
        }
}


                        // rechte Randbemessung

x_bounds_right=-1;
for(x=width;x>0;x--){
        sum_line_alt=0;
        for(i=0;i<(thickness*2);i++){
                sum_line_neu=sum_line_alt+getPixel(x+X,Y+height/2-thickness+i);
                sum_line_alt=sum_line_neu;
        }
        current_line_value=sum_line_neu/(thickness*2);

        if(current_line_value<current*current_add){

        }else{
                if(ink==1){
                setColor(brightness);
                drawLine(X+x,Y+height/2-thickness,X+x,Y+height/2+thickness);
                }
                x_bounds_right=x;
        x=1;
        }
}



                        //obere Randbemessung

y_bounds_upper=-1;
for(y=0;y<height;y++){
        sum_line_alt=0;
        for(i=0;i<(thickness*2);i++){
                sum_line_neu=sum_line_alt+getPixel(X+width/2-thickness+i,y+Y);
```

```
                sum_line_alt=sum_line_neu;
        }
        current_line_value=sum_line_neu/(thickness*2);

        if(current_line_value<current*current_add){

        } else {
                if (ink==1){
                setColor(brightness);
                drawLine(X+width/2-thickness,Y+y,X+width/2+thickness,Y+y);
                }
                y_bounds_upper=y;
        y=height-1;
        }
}


                        // untere Randbemessung

y_bounds_lower=-1;
for(y=height;y>0;y--){
        sum_line_alt=0;
        for(i=0;i<(thickness*2);i++){
                sum_line_neu=sum_line_alt+getPixel(X+width/2-thickness+i,y+Y);
                sum_line_alt=sum_line_neu;
        }
        current_line_value=sum_line_neu/(thickness*2);

        if(current_line_value<current*current_add){

        } else {
                if (ink==1){
                setColor(brightness);
                drawLine(X+width/2-thickness,Y+y,X+width/2+thickness,Y+y);
                }
                y_bounds_lower=y;
        y=1;
        }
}


                        // Bestimmung der Mitte


if(x_bounds_left==-1 || x_bounds_right==-1 ||  y_bounds_upper==-1 || y_bounds_lower==-
1){
        showMessage("Bild "+n+" Kein Objekt gefunden");
        exit;
```

```
}


if(n==1){
	X_center[0]=(x_bounds_left+x_bounds_right)/2;
	Y_center[0]=(y_bounds_upper+y_bounds_lower)/2;
}

X_center[n]=((x_bounds_left+x_bounds_right)/2)+X;
Y_center[n]=((y_bounds_upper+y_bounds_lower)/2)+Y;

if(ink==1){
	drawLine(X_center[n]-5,Y_center[n],X_center[n]+5,Y_center[n]);
	drawLine(X_center[n],Y_center[n]-5,X_center[n],Y_center[n]+5);
	drawLine(X_center[n],Y+y_bounds_upper,X_center[n],Y+y_bounds_upper-5);
	drawLine(X_center[n],Y+y_bounds_lower,X_center[n],Y+y_bounds_lower+5);
	drawLine(X+x_bounds_left,Y_center[n],X+x_bounds_left-5,Y_center[n]);
	drawLine(X+x_bounds_right,Y_center[n],X+x_bounds_right+5,Y_center[n]);
}

if(n>1){
	setSelectionLocation(X_center[n]-width/2,Y_center[n]-height/2);
}




////////////////////////////////////////////////////////////////////
// Drehung durch Vergleich ganzer Bereiche//
////////////////////////////////////////////////////////////////////

if(execute_rotation==1){



	circle_neu=newArray(361);
	circle_diff=newArray(361);
	distance=0.75*minOf((x_bounds_right-x_bounds_left)/2,(y_bounds_lower-
y_bounds_upper)/2);


	for(rot=1;rot<=360;rot++){
		circle_neu[rot]=Scan(X_center[n],Y_center[n],distance,(rot*3.14/180));
	}

	if(n==1){
		circle_alt=circle_neu;
	}
```

```
	for(diff_case=0;diff_case<=360;diff_case++){
		for(rot=0;rot<=360;rot++){
			if(diff_case+rot<=360){
				circle_diff[diff_case]=circle_diff[diff_case] +
abs(circle_alt[rot]-circle_neu[rot+diff_case]);
			}
			else{
				circle_diff[diff_case]=circle_diff[diff_case] +
abs(circle_alt[rot]-circle_neu[rot+diff_case-360]);
			}
		}
	}

	circle_alt=circle_neu;


	circle_min=100000;
	circle_rot_min=0;
	for(rot=0;rot<=360;rot++){
		if(circle_diff[rot]<circle_min){
			circle_min=circle_diff[rot];
			circle_rot_min=rot;
		}
	}

	rotation[n]=rotation[n-1]+circle_rot_min;

	if(ink==1){
		turn_indicator_x=sin(rotation[n]*3.14/180)*(Y+y_bounds_upper-Y_center[n]);
		turn_indicator_y=(-1)*sin(rotation[n]*3.14/180)*(X+x_bounds_left-
X_center[n]);

	drawLine(X_center[n]+turn_indicator_x,Y+y_bounds_upper,X_center[n]+turn_indica
tor_x,Y+y_bounds_upper-10);
		drawLine(X_center[n]-turn_indicator_x,Y+y_bounds_lower,X_center[n]-
turn_indicator_x,Y+y_bounds_lower+10);
		drawLine(X+x_bounds_left,Y_center[n]+turn_indicator_y,X+x_bounds_left-
10,Y_center[n]+turn_indicator_y);
		drawLine(X+x_bounds_right,Y_center[n]-
turn_indicator_y,X+x_bounds_right+10,Y_center[n]-turn_indicator_y);
	}
}

print("\\Update:verfolge Bewegung. Bild: "+ n+"/"+(nSlices));

}
```

```
////////////////////////////////////////////////////////////////////////////
// Erstellen eines neuen Images nur mit der Zelle //
////////////////////////////////////////////////////////////////////////////


print("Ziel kopieren und drehen");
print("");
newImage("Cells", 16,width,height,n-1);
Image_2=getImageID();
setLocation(0,0);

for(n=1;n<=nSlices;n++){
	selectImage(Image_1);
	setSlice(n);
	makeRectangle(X_center[n]-width/2,Y_center[n]-height/2 , width, height);
	run("Copy");
	selectImage(Image_2);
	setSlice(n);
	run("Paste");
	run("Arbitrarily...", "interpolate slice angle="+rotation[n]);
	}



run("Enhance Contrast", "saturated="+AufH);
updateDisplay();
selectWindow("Log");



//////////////////////////////////////
// Erstellen einer Maske//
//////////////////////////////////////

do{

jump=0;

	selectImage(Image_2);


	E=newArray(width*height*(nSlices+1));
	D=newArray(width*height);



	for(n=1;n<=(nSlices);n++){
		setSlice(n);
			for(x=0;x<=width-1;x++) {
```

```
                    for(y=0;y<=height-1;y++) {
                        E[x+y*width+n*(width*height)]=getPixel(x,y);
                }
                }
}


for(n=1;n<=(nSlices-Step);n++){
        A=newArray(width*height);
                for(x=0;x<=width-1;x++) {
                    for(y=0;y<=height-1;y++) {
                        A[x+y*width]=E[x+y*width+n*(width*height)];
                }
                }


        B=newArray(width*height);
                for(x=0;x<=width-1;x++) {
                    for(y=0;y<=height-1;y++) {
                        B[x+y*width]=E[x+y*width+(n+Step)*(width*height)];
                }
                }

        C=newArray(width*height);
        for(i=0;i<(width*height);i++){
                C[i]=B[i]-A[i];
                if(A[i]==0){
                    C[i]=0;
                }
                if(C[i]<Mask_threshold){
                    C[i]=0;
                }
                if(C[i]>=Mask_threshold){
                    C[i]=1;
                }
        }

        for(i=0;i<(width*height);i++){
                D[i]=D[i]+C[i];

        }
print("\\Update:Vergleiche durchgeführt "+ n+"/"+(nSlices-Step));
}

C=newArray(width*height);
counter=0;
for(i=0;i<(width*height);i++){
                if(D[i]>=incidence){
                    C[i]=1;
```

```
                        counter=counter+1;
                }
        }

        if(counter==0){
                showMessageWithCancel("Keine Unterschiede gefunden");
        }


        print("erstelle Maske");
        newImage("Maske", 16 , width, height, 1);
        Image_3=getImageID();
        setLocation(width+20,0);
        selectImage(Image_3);
        setSlice(1);
        for(x=0;x<=width-1;x++) {
                        for(y=0;y<=height-1;y++) {
                                setPixel(x,y,C[x+y*width]*255);
                }
                }

        run("Enhance Contrast", "saturated=0");
        updateDisplay();


        jump=getBoolean("Mit anderen Randwerten noch einmal versuchen?");
                // Möglichkeit zum Neuversuch mit anderen Maskenwerten

                if(jump==1){
                        selectImage(Image_3);
                        close();

                        selectImage(Image_2);

                        do{
                                Dialog.create("Zellenauswertung");

                                        Dialog.addMessage("\n Bildervergleich und suchen nach
Unterschieben.");
                                        Dialog.addNumber("Schrittweite zu Erstellung einer
Maske",Step);
                                        Dialog.addNumber("Schwellerwert zur Erkennung von
Veränderungen",Mask_threshold);
                                        Dialog.addNumber("Häufigkeit einer Veränderung bis
sie ausgewählt wird",incidence);

                                Dialog.show();

                                Step=Dialog.getNumber();
```

```
                        Mask_threshold=Dialog.getNumber();
                        incidence=Dialog.getNumber();

                all_items_correct=1;

                        if (Step<=0 || Step >=nSlices-2){
                                showMessageWithCancel("Schrittweite außerhalb der
Parameter");
                                all_items_correct=0;
                        }

                        if (incidence<=0 || incidence>nSlices){
                                showMessageWithCancel("Unterschiedshäufigkeit
außerhalb der Parameter");
                                all_items_correct=0;
                        }

                } while(all_items_correct!=1);

        }

}while (jump!=0);



        // Möglichkeit zur Nacharbeitung der Maske
work=getBoolean("Maske manuel bearbeiten? \n    Beenden mit "Umschalttaste" ");

if(work==1){

    ready=0;
    setBackgroundColor(0,0,0);
    setForegroundColor(255,255,255);
        do{
            //setTool(18);
                do{
                }while(isKeyDown("shift")!=1);

        ready=getBoolean("Bearbeitung Beenden?");

        }while(ready!=1);

    //setTool(0);

    counter=0;
    C=newArray(width*height);

    selectImage(Image_3);
```

```
	setSlice(1);
	for(x=0;x<=width-1;x++) {
		for(y=0;y<=height-1;y++) {
			if (getPixel(x,y)>=200){
				C[x+y*width]=1;
				counter=counter+1;
			}
		}
	}

}


////////////////////////////////////////////////////////////////////
// Auswerten der Zellen anhand der Maske//
////////////////////////////////////////////////////////////////////

write("Slice	Wert				Anzahl der Ausgewerteten Pixel:
"+counter+"\n  ");

selectImage(Image_2);
for(n=1;n<=nSlices;n++){
	setSlice(n);
	value=0;
		for(x=0;x<=width-1;x++) {
			for(y=0;y<=height-1;y++) {
				value=value+getPixel(x,y)*(C[x+y*width]);

		}
		}
	write(n+"	"+value/counter);
}

print("Vorgang erfolgreich");

exit;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

	//////////////////////
	// Funktionen//
	//////////////////////


function Scan(x_location,y_location,scan_distance,scan_angle){
	largeness_x=1;		//1/2 * Breite des Scanfeldes
	largeness_y=1;		//1/2 * Höhe des Scanfeldes
```

```
                i_scan_alt=0;
                sum_scan_alt=0;
                x_angle=sin(scan_angle)*scan_distance;
                y_angle=cos(scan_angle)*scan_distance;

                for(y_scan=-largeness_y;y_scan<=largeness_y;y_scan++){
                        for(x_scan=-largeness_x;x_scan<=largeness_x;x_scan++){
                                sum_scan_neu = sum_scan_alt +
getPixel(x_location+x_angle+x_scan,y_location+y_angle+y_scan);
                                sum_scan_alt=sum_scan_neu;
                                i_scan_neu = i_scan_alt+1;
                                i_scan_alt=i_scan_neu;
                                if(ink_s==1){

        setPixel(x_location+x_angle+x_scan,y_location+y_angle+y_scan,brightness);
                                }
                        }
                }
                value=(sum_scan_neu/i_scan_neu);

                return value;

}
```