Fakultät für Physik und Astronomie

Ruprecht-Karls-Universität Heidelberg

Diplomarbeit

Im Studiengang Physik

vorgelegt von

Sven Wanner

geboren in Sindelfingen

2010

Interaktives Rendering von Wellendaten windgetriebener Wasseroberflächen und Ereignisklassifizierung

Die Diplomarbeit wurde von Sven Wanner

ausgeführt am

IUP Heidelberg

unter der Betreuung von

Herrn Prof. Bernd Jähne

Interaktives Rendering von Wellendaten windgetriebener Wasseroberflächen und Ereignisklassifizierung:

Die vorliegende Diplomarbeit beschäftigt sich sowohl mit dem interaktiven 3D Rendering als auch mit der automatischen Detektion von Ereignissen auf windgetriebenen Wasseroberflächen. Bei diesen Ereignissen handelt es sich im Wesentlichen um Einschläge von Regentropfen auf der Wasseroberfläche und das Brechen von Wellen in kleinskaligen Wellenfeldern. Hierbei wird unter kombinierter Verwendung eines Visualisierungstool (WaveVis) und einer Software zur interaktiven Segmentierung (Ilastik), durch Training von Beispielereignissen ein *Random-Forest* trainiert, mit dessen Hilfe große Datensätze quantitativ in Bezug auf das jeweils betrachtete Ereignis untersucht werden können. Die Ergebnisse werden dabei zum Vergleich mit früheren Messungen herangezogen. So werden aus Daten einer Messkampagne die genauen Regenraten ermittelt und ein Verfahren zur automatischen Detektion von *Microscale-Breaking-Waves* auf der Basis von Neigungs- und Höheninformation der Wasseroberfläche vorgestellt.

Interactive rendering of data from wind-driven watersurfaces and event classification:

This diploma thesis deals with both, the interactive 3D rendering and with the automatic detection of events on wind-driven water surfaces. It basically acts of impacts of rain drops on the water surface and breaking waves in small-scale wave fields. Under the combined use of the visualization tool (WaveVis) and a software for interactive segmentation (Ilastik), example events are used to train a „Random-Forest“, with whose assistance large data records can be examined quantitatively about the regarded events. By means of our results, the examined method will be compared with pre-existent methods. From data of a measuring campaign the exact rainrates are determined and a procedure for automatic detection of microscale-breaking-waves, exclusively from slope and height data of the water surface, is presented.

Inhaltsverzeichnis

1 Einleitung

In der modernen Wissenschaft haben sich durch den Einsatz von Computern und digitalen Messinstrumenten völlig neue Möglichkeiten zur Erforschung von Naturphänomenen ergeben. Geschwindigkeit und Genauigkeit mit der Daten genommen oder simuliert werden können haben enorm zugenommen. Ebenso Menge und Vielfalt, weswegen Ansätze gefordert sind, um als Mensch diese Datenflut verarbeiten und interpretieren zu können. Ein Ansatz zur Interpretation kann hierbei die Computergrafik [28] sein. Daten für den Betrachter in einer sinnvollen Art und Weise zu visualisieren und ihm die Möglichkeit zu geben mit diesen zu interagieren kann der Schlüssel zum Verständnis der zugrunde liegenden Phänomene sein. Aber nicht nur im Sinne der optischen Repräsentation von Daten hat sich der Computer als sehr nützlich erwiesen, sondern auch bei der Interpretation selbst. So können Methoden des maschinellen Lernens [5, 1] eingesetzt werden, um in großen Datensätzen selbstständig nach Zusammenhängen oder Merkmalen zu suchen und damit detaillierte Analysen oft überhaupt erst ermöglicht werden. Beide Arten der computergestützten Dateninterpretation kommen bei der Analyse von Vorgängen auf windgetriebenen Wasseroberflächen zum Einsatz [30].

Zur Untersuchung physikalischer Effekte auf, in hohem Maße dynamischen, Wasseroberflächen können mittels Computergrafik erzeugte Visualisierungen realer Daten sehr hilfreich sein. Die Möglichkeit, ablaufende Prozesse aus beliebigen Perspektiven und mit beliebiger Abspielgeschwindigkeit betrachten zu können, ist bei der Interpretation der Vorgänge von großem Nutzen. Erforscht werden hierbei Prozesse die den Gasaustausch zwischen Atmosphäre und Ozean beeinflussen, wobei Wind eine entscheidende Rolle spielt. Er ist für die Bildung von Wellen verantwortlich, die ihrerseits auf die Phasengrenze zwischen Luft und Wasser einwirken. Diese Grenzschicht und das dadurch entstehende Konzentrationsgefälle kontrollieren die Diffusion der Gasmoleküle in beide Richtungen [20, 21]. Daher sind auf diese Schicht Einfluss nehmende Prozesse für den Gastransport wichtige Parameter. Es ist intuitiv klar, dass das Erscheinungsbild des Ozeans prägende Vorgänge, wie die Einschläge von Regentropfen auf die Wasseroberfläche und brechende Wellen, zu diesen, die Grenzschicht beeinflussenden Vorgängen zählen. Quantitative und statistische Aussagen, über deren Bedeutung für den Gastransfer machen zu können wird also ein wichtiger Bestandteil eines Verständnisses der Wechselwirkung von Atmosphäre und Ozeanen sein. Um diese Vorgänge zu erforschen, werden neben Feldversuchen auch Experimente unter kontrollierten Bedingungen in Wind-Wellen-Kanälen durchgeführt, um gezielt die Einflüsse von Wind, Wellen, Regen und künstlich aufgebrachten Oberflächenfilmen auf die Gastransferraten zu bestimmen.

Eine wesentliche Rolle spielen dabei oberflächenerneuernde Vorgänge wie brechende Wellen, deren quantitative Untersuchung jedoch experimentelle Schwierigkeiten mit sich bringt. *Microscale-Breaking-Waves* [25], die schon bei relativ niedrigen Windgeschwindigkeiten auftreten, sind brechende Wellen, deren Erscheinungsbild sich nicht durch ausgeprägte Schaumkronen auszeichnet, sondern nur durch eine erhöhte Verwirbelung in den oberen Wasserschichten und damit einhergehenden Umwälzungen. Eine Detektion erfordert daher Methoden zur Visualisierung dieser Wasserumwälzungen und den vorhandenen Strömungsfeldern. Infrarotaufnahmen der Wasseroberfläche (Thermografie) [41, 32] lassen Rückschlüsse auf Temperaturumwälzungen zu. Mittels laserinduzierter Fluoreszenz (LIF) [16, 22] bestimmter Tracerstoffe können Umwälzungen von Gasen beobachtet werden. Strömungsverhältnisse lassen sich durch Eintrag von Partikeln ins Wasser beobachten (PIV) [10, 11]. Über farbcodierte Aufnahmen der Wasseroberfläche (CISG) [29, 30] können mit High-Speed-Kameras zeitlich hoch aufgelöste Neigungsdaten der Wasseroberfläche bestimmt werden. Diese Verfahren lassen sich aber untereinander oft nur schwer kombinieren, was den Informationsgehalt jeder Vorgehensweise immer einschränkt und eine detaillierte Analyse der Vorgänge in Ihrer Gesamtheit schwierig macht.

Doch nicht nur mangelnde Flexibilität der Messmethoden sind ein Problem bei der Untersuchung von Phänomenen auf windgetriebenen Wasseroberflächen, sondern auch die Geschwindigkeit mit der diese ablaufen. Die erwähnten Aufnahmetechniken mittels Hochgeschwindigkeitskameras schaffen hier Abhilfe, allerdings produzieren sie große Mengen an Daten. Bestehen die Resultate wie im Falle der CISG aus mehreren Kanälen wie den Neigungen der Wasseroberfläche in x- und y-Richtung, daraus rekonstruierbaren Höheninformationen und beliebigen anderen Messungen wie zum Beispiel Infrarotaufnahmen, wird es für einen Betrachter sehr schnell unmöglich die Vorgänge in ihrer Gesamtheit zu erfassen. Diesem Problem kann nur durch geeignete Werkzeuge, wie zum Beispiel der erwähnten Computergrafik, begegnet werden. Damit können aus solchen Phänomenen extrahierte Informationen für den Menschen wieder interpretierbar visualisiert werden. Die Zielsetzung dieser Diplomarbeit ist es daher, in Wind-Wellen-Kanälen gewonnene Daten zu visualisieren und zu analysieren. Hierfür wurde ein 3D Viualisierungstool [30] zur Darstellung der windgetriebenen Wasseroberfläche weiterentwickelt, das aus experimentell gewonnenen Neigungs- und Höheninformationen der Wasseroberfläche ein interaktives Rendering ermöglicht um auf den Gasaustausch Einfluss nehmende Vorgänge gezielt untersuchen zu können. So lassen sich mithilfe dieser Software Prozesse wie Einschläge von Regentropfen, brechende Wellen, Wärmetransport, Strömungsverhältnisse oder Langmuir Zirkulation visualisieren. Weiterhin sollen mithilfe des maschinellen Lernens und der Bildverarbeitung Methoden entwickelt werden, um direkt aus den Daten Ereignisse auf der Wasseroberfläche zu segmentieren. Dazu gehören die Einschläge von Regentropfen und eine damit einhergehende Analyse der Regenraten einer Messkampagne sowie die Detektion von *Microscale-Breaking-Waves* aus zeitlich hoch aufgelösten Neigungs- und Höheninformationen.

2 Grundlagen

2.1 Maschinelles Lernen

Der Begriff maschinelles Lernen ist sehr vielfältig und umfasst einen weiten Bereich von Anwendungen. Liegen dem Menschen große Datenmengen vor, die für ihn kaum oder gar nicht in ihren Zusammenhängen überschaubar sind, müssen Methoden gefunden werden automatisch Muster aus diesen zu extrahieren, um damit Teilmengen gleicher Bedeutung zu bilden. Dies entspricht genau den Erfordernissen der in dieser Arbeit verwendeten Daten. Den Prozess in großen Datenmengen ein Beziehungsgeflecht aufzuspüren nennt man auch *Data-Mining* [12]. Es existiert eine Vielzahl von Einsatzmöglichkeiten aus den unterschiedlichsten Anwendungsbereichen. Von der Diagnostik in der Medizin bis zum Aufspüren auffälligen Verhaltens im Börsenhandel ist alles möglich. Doch das maschinelle Lernen geht weit über das Strukturieren von Informationen aus Datenbanken hinaus. Ebenso ist es Teil der künstlichen Intelligenz. „Intelligente“ Systeme müssen in der Lage sein dazu zu lernen, um sich veränderten Bedingungen anzupassen und sich in einer komplexen nicht vollständig vordefinierbaren Umgebung zurecht finden zu können. Ein Teilbereich davon ist die immer besser werdende Gesichts- und Spracherkennung, die ohne maschinelles Lernen nicht denkbar wäre.
Der Überbegriff maschinelles Lernen kann grob in zwei Bereiche unterteilt werden, das überwachte und das unüberwachte Lernen.

2.1.1 Unüberwachtes Lernen

Unter unüberwachtem Lernen versteht man das selbstständige Unterteilen der Daten durch den Lernalgorithmus in sogenannte *Cluster*. Dabei werden Teilmengen gleicher Charakteristiken in den Daten ermittelt und anhand derer ein Zuordnungsschema erstellt, mithilfe dessen auch unbekannte Daten eines der *Cluster* zugeordnet werden können. Ein häufig verwendetes Beispiel zum Auffinden dieser Regionen ist der *k-means Algorithmus* [23]. In diesem werden Schwerpunkte in den Daten zufällig gewählt und mittels einer Abstandsnorm, zum Beispiel der Euklidischen, Datenpunkte den Schwerpunkten zugeordnet. In den so entstandenen *Clustern* werden dann iterativ zuerst die Schwerpunkte zentriert und erneut über den Abstand die *Cluster*-Zugehörigkeit bestimmt. Dies geschieht bis zu einer bestimmten Iterationstiefe oder bis die Schwerpunkte sich nicht mehr bewegen.

2.1.2 Überwachtes Lernen

Beim überwachten Lernen werden Lernalgorithmen vom Menschen trainiert, indem die Zugehörigkeiten einzelner Datenbeispiele zu einer bestimmten Gruppe im Vorfeld festgelegt werden. Anhand dieser, von menschlicher Einschätzung definierten, Abhängigkeiten (Daten↔Datenzugehörigkeit) werden wieder Zuordnungsschemata erstellt, die eine Klassifizierung unbekannter Daten zulässt.

Diese kurze Beschreibung des maschinellen Lernens soll nur eine Begründung für die im folgenden verwendeten Werkzeuge sein. Deshalb wird der Leser für tiefergehende Erläuterungen auf Alpaydin [1] oder Michie [27] verwiesen.

In dieser Arbeit wird der Ansatz des überwachten Lernens gewählt um Ereignisse in Daten aus Wind-Wellen-Kanälen auf der Wasseroberfläche zu detektieren (siehe Kapitel 4). Diese Wahl liegt darin begründet, dass in den Daten nur bestimmte charakteristische Strukturen gefunden werden sollen, was eine vorherige Definition durch den Menschen erfordert. Da die Komplexität der hier verwendeten Daten sehr hoch ist, wird zur besseren Identifizierung der relevanten Charakteristiken der Merkmalsraum mittels Filtern aus der Bildverarbeitung erweitert. Zur nachfolgenden Analyse durch überwachtes Lernen eignet sich dabei der *Random-Forest* (siehe Kapitel 2.1.4), da dieser auch auf sehr großen Daten effizient arbeitet und sehr stabil gegenüber Ausreißern ist.

2.1.3 CART - Classification-And-Regression-Trees

Die folgende Einführung in die Grundlagen des CART-Algorithmus beruht im Wesentlichen auf dem Buch *Classification and Regression Trees* von Breiman, Freidman, Olsen und Stone [6] und soll dem Leser als Zusammenfassung der zentralen Ideen dieses, zu diesem Thema immer wieder zitierten Werkes dienen. *Classification-* und *Regression-Trees* sind die Grundlage des *Random-Forest* 2.1.4 und dieser liegt der interaktiven Segmentierung (siehe Kapitel 2.3 und Kapitel 4) zugrunde. Unter Segmentierung versteht man dabei einen Bereich der Bildverarbeitung und des maschinellen Sehens, in dem zusammenhängende Gebiete in Bildern oder Volumen anhand von bestimmten Kriterien automatisch vom Computer erkannt werden. Für nähere Informationen zur Segmentierung wird der Leser auf Jähne [19] verwiesen. Der Begriff „interaktive Segmentierung“ ist auf die, für Teile dieser Arbeit verwendete Software Ilastik[1] 2.3 bezogen, deren Prinzip und Funktionsweise in den folgenden Kapiteln erläutert werden soll. Für ausführlichere Informationen zu Ilastik und weitere Anwendungsbeispiele sei Sommer [36] genannt.
CART ist eine Klassifikations-Methode auf der Grundlage von Entscheidungsbäumen die einen Satz von unabhängigen Variablen $\mathbf{X}_i$ einer abhängigen Zielvariablen y_i zuordnen. Dabei werden aus vorhandenen und durch den Menschen bzw. Berechnung bereits klassifizierten Daten Entscheidungsbäume konstruiert, die dann

[1] http://hci.iwr.uni-heidelberg.de/download3/ilastik.php

zur Klassifizierung unbekannter Daten verwendet werden können. Dafür werden rekursiv, möglichst homogene Aufteilungen der Grundbeispielmenge gesucht, wobei CART aus allen möglichen Variablen und allen vorhandenen Klassen die jeweils beste Art der Aufteilung wählt.

Classification Trees

Eines der illustrativsten Beispiele in Bezug auf die Verwendung von *Classification-Trees* ist das in Breiman [6] zur Klassifizierung von Herzinfarkt Patienten angeführte. Hintergrund ist eine Studie am San Diego Medical Center der University of California, in der bei Menschen mit Herzinfarkten innerhalb der ersten 24 Stunden nach der Herzattacke 19 verschiedene Variablen bestimmt wurden. Dazu gehörten unter anderem: das Alter, der Blutdruck, Herzrythmus u.v.m. Dadurch sollte eine schnelle Klassifizierung des Patienten in Risikoklassen ermöglicht werden, denn es gibt signifikante Zusammenhänge zwischen dem Zusammenspiel bestimmter Faktoren und der Sterblichkeit innerhalb von 30 Tagen nach einem Infarkt. Dazu wurde in dieser Studie ein Regelnetzwerk aufgestellt, das auf Grund der innerhalb von 24 Stunden nach dem Infark erhobenen Daten, den Patienten einer Hoch-Risiko bzw. einer Nieder-Risikogruppe zuteilt. In Abbildung 2.1 ist ein Teil der Logik eines solchen Abfragenetzwerkes der Studie zu sehen.

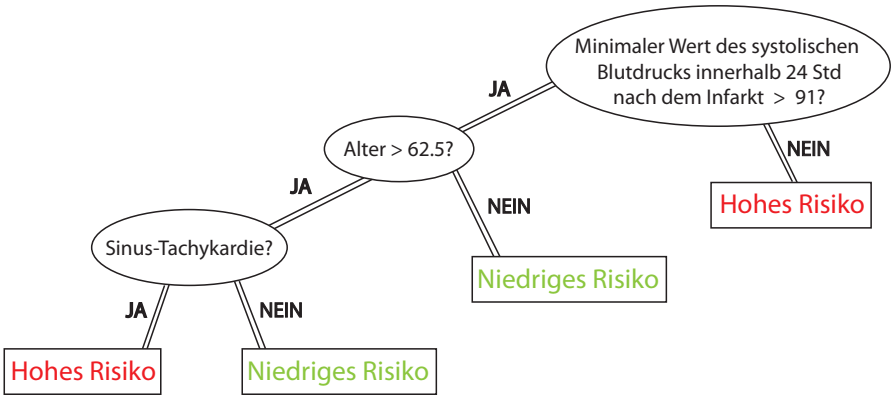

Abbildung 2.1: Klassifizierung der 30-Tage-Mortalität nach Herzinfarkten

Das Ziel solcher Abfragenetzwerke ist einen gegebenen Satz von Daten systematisch einer bestimmten Zielklasse zuzuordnen. Allgemein lässt sich ein solcher Datensatz gemessener Variablen als Vektor $\mathbf{x}$ in einem Raum X, welcher der Raum aller möglicher Messungen ist, darstellen. Im Falle des obigen Beispiels ist X der 19-dimensionale Raum der gemessenen Patientenparameter und die Klassifizierung ordnet jeden Patienten, repräsentiert durch einen Punkt im Raum X, einer Risiko-Klasse zu.

Konstruktion von Klassifikatoren

Wie in der Einleitung schon angedeutet, wird bei der Klassifizierung ein Satz von Parametern als ein Vektor im Raum aller möglichen Messungen der Parameter aufgefasst. Wichtig dabei ist, dass jeder zu klassifizierende Vektor einen Punkt in eben diesem Raum X darstellt. Weiterhin wird ein Satz $\Gamma = 1, 2, ..., M$ von Klassen definiert denen beliebige Vektoren $\mathbf{x} \in X$ in eindeutiger Weise zugeordnet werden können.

Definition Klassifikator: Ein Klassifikator oder eine Klassifizierungsregel ist eine auf X definierte Funktion $d(\mathbf{x})$ in der Art, dass für jedes $\mathbf{x} \in X$, $d(\mathbf{x})$ auf eine der Zahlen $\Gamma = 1, 2, ..., M$ abbildet.

Alternativ kann man auch sagen, dass ein Klassifikator den Raum X in M disjunkte Unterräume aufteilt, deren Vereinigung ganz X ist. In Abbildung 2.2 ist exemplarisch ein Klassifikator als binäre Baumstruktur und die Unterteilung des Raumes (*Feature-Space*) X in Γ Klassen dargestellt. Von der „Wurzel", dem *Root-Node* aus wird an jedem kreisförmigen *Node* dabei solange eine binäre Entscheidung getroffen bis auf diese Weise jedem Vektor $\mathbf{x} \in X$ eine Klasse in Form eines quadratisch dargestellten *Leaf-Nodes* zugewiesen wurde.

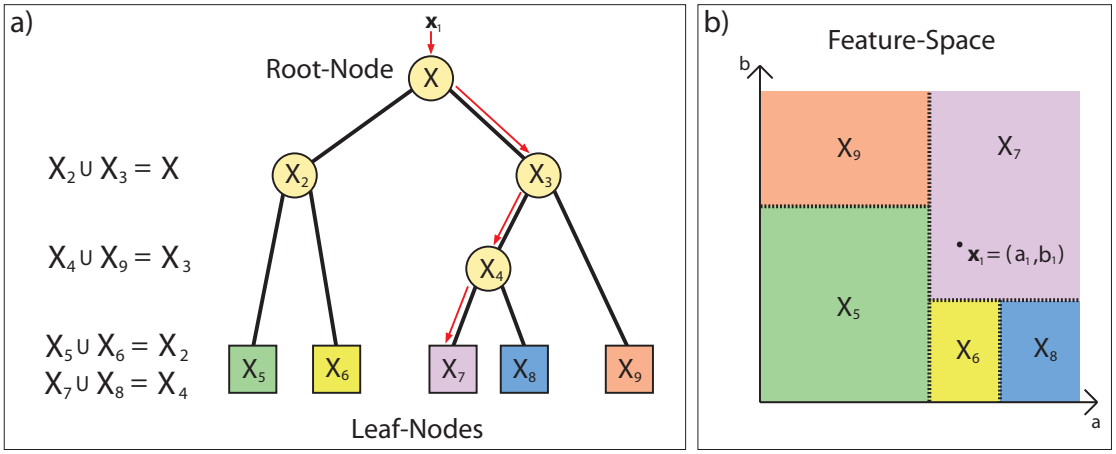

Abbildung 2.2: a) Illustration einer binären Baumstruktur eines *Classification-Trees*, b) Durch die Baumstruktur aus a) definierte Klassenzuweisung jedes Punktes im *Feature-Raum*

Zur Konstruktion von Klassifikatoren dienen wohlbekannte Lernbeispiele. So wurden im Falle der Infarktstudie Daten von 215 Patienten erhoben, die die ersten 24 Stunden ihres Infarkts überlebten. Aus diesen wurde ein Klassifikator konstruiert, der Menschen mit Herzinfarkt einer bestimmten Risikogruppe zuordnet.

Definition Lerndatensatz: Ein Lerndatensatz, bestehend aus N Lernbeispielen, ist definiert als:

$$L = \{(\mathbf{x}_1, \gamma_1), \ldots, (\mathbf{x}_N, \gamma_N)\} \text{ mit } \mathbf{x}_i \in X \text{ und } \gamma_i \in \Gamma \tag{2.1}$$

Entscheidend ist nun die Frage, wie die Struktur eines *Classification-Trees* aus den vorgegebenen Lerndaten gewonnen werden kann. „*It turns out that the class assignment problem is simple. The whole story is in finding good splits and in knowing when to stop splitting.*“[1]

Splitting-Rules

Um die optimale Verzweigung an jedem *Node* (t_{Basis}) zu finden wird das Parameterfeld x_i der Lernbeispiele betrachtet, jede mögliche Verzweigung versucht und dadurch der ideale Schwellwert x_i^T für die Unterteilung gefunden (Vergleiche Abbildung 2.3). Dies wird der Reihe nach für alle x_i, also alle möglichen Parameter, durchgeführt und die Aufteilung mit dem höchsten Maß an Homogenität als Beste gewählt. Der Vorgang wird an den entstandenen *Nodes* (t_L, t_R) wiederholt und so sukzessiv die optimale Baumstruktur ermittelt.

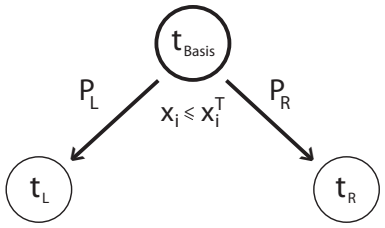

Abbildung 2.3: Illustration der Verzweigung eines *Parent-Node* in zwei *Child-Nodes* (aus Timofeev [37])

Zur Definition der maximalen Homogenität wird eine sogenannte *impurity function* $I(t)$ definiert. $I(t)$ hat dabei die Eigenschaft ihren größten Wert zurückzugeben wenn alle Klassen in einem Node vorhanden sind und ihren Kleinsten bei nur einer enthaltenen Klasse. Dann ist die maximale Homogenität der Verteilung durch die Maximierung der Änderung der *impurity function* gegeben:

$$\Delta I(t) = I(t_{Basis}) - P_L I(t_L) - P_R I(t_R) \tag{2.2}$$

P_L und P_R drücken dabei aus, welcher Anteil der gesamten Lernbeispiele aus t_{Basis} jeweils t_L beziehungsweise t_R zugewiesen werden.

[1] Breiman, Leo: Classification and Regression Trees [6], S. 23.

Zur Entwicklung der gesamten Baumstruktur wird an jedem Node das Maximierungsproblem:

$$\underset{x_i<=x_i^T, i=1,\ldots,N}{\arg\max} \left[I\left(t_{Basis}\right) - P_L I\left(t_L\right) - P_R I\left(t_R\right) \right] \tag{2.3}$$

gelöst. Das konkrete Aussehen der *Impurity-Function* $i\left(t\right)$ hängt ganz von der Problemstellung ab, weshalb hier exemplarisch die zwei Wichtigsten aufgeführt werden sollen. Konkreteres findet sich in [6, 37, 33].

Gini index:

Eine Möglichkeit die *Impurity-Function* zu bestimmen, ist über den Gini Index:

$$i\left(t\right) = 1 - \sum_k p^2\left(k|t\right) \tag{2.4}$$

Hierbei sind $k \in \Gamma$, die Klassenindizes und $p\left(k|t\right)$ die Wahrscheinlichkeit, dass die Klasse k in *Node* t vorkommt. Der *Gini-Index* nimmt ein Maximum an, wenn jede Klasse im Node t mit gleicher Wahrscheinlichkeit vorkommt ($i\left(t\right) = 1/k$), respektive ihr Minimum, wenn alle Fälle in einem Node zur gleichen Klasse gehören ($i\left(t\right) = 0$). Auf das Maximierungsproblem angewandt ergibt sich:

$$\underset{x_i<=x_i^T, i=1,\ldots,N}{\arg\max} \left[-\sum_{k=1}^{M} p^2\left(k|t_{Basis}\right) + P_L \sum_{k=1}^{M} p^2\left(k|t_L\right) + P_R \sum_{k=1}^{M} p^2\left(k|t_R\right) \right] \tag{2.5}$$

Damit arbeitet der Gini-Algorithmus nach dem Prinzip der Klassenselektion. Idealerweise sortiert er die Klassen der Größe nach und wird versuchen, die Größten zuerst aus den Daten zu isolieren. Für die weiteren Betrachtungen des *Random-Forrest* beziehungsweise die interaktive Segmentierungssoftware Ilastik ist auch nur er von Belang.

Twoing splitting rule:

Der Twoing-Algorithmus sucht im Gegensatz zum Gini-Algorithmus nach zwei Klassen die zusammen mehr als 50% der Daten erfassen. Er hat gegenüber dem Gini-Algorithmus einen Geschwindigkeitsnachteil.

Die Änderung der *impurity function* sieht im Twoing-Algorithmus wie folgt aus:

$$\Delta i\left(t\right) = \frac{P_L P_R}{4}\left[\sum_{k=1}^{M}\left|p\left(k|t_L\right) - p\left(k|t_R\right)\right|\right]^2 \qquad k \in \Gamma \tag{2.6}$$

Weitere Möglichkeiten sind die *Cross-Entropy*, χ^2 oder *maximum deviation rule*. Aber in Breiman [6] wird gezeigt, dass die Wahl der Splitting Methode relativ unbedeutend für die Qualität eines *Classification-Trees* ist. Von sehr viel größerer Bedeutung ist die Optimierung der *Classifier*, auch *pruning procedure* genannt. Es soll an dieser Stelle aber nicht näher darauf eingangen werden, da die *Classification-Trees* im Kontext des *Random-Forest* behandelt werden und in diesem jeder einzelne Baum zur Gänze ausgebildet wird. Für detailliertere Informationen wird deshalb hier auf Hastie et al. [13] beziehungsweise Breiman [7] verwiesen.

Regression-Trees

Der wesentliche Unterschied zwischen Klassifikations- und Regressions-Bäumen besteht in der Art der Zielvariablen y_i, die bei ersteren von kategorischer, bei zweiteren jedoch von kontinuierlicher Natur sind. CART wird also je nach Art der vorliegenden Daten die eine oder die andere Variante wählen. Statt Klassenzuordungen gibt es in *Regression-Trees* Variablen-Vektoren $\mathbf{x}_i \in \mathbb{R}^m$ mit $m \in \mathbb{N}$ und eine Antwort $y_i \in \mathbb{R}$. Um die optimale Aufteilung zu finden, wird ein *squared residuals minimization*-Algorithmus verwendet, da aufgrund der fehlenden vorgegebenen Klassenzuweisungen *Splitting-Rules* wie der *Gini-Algorithmus* hier nicht anwendbar sind.

Splitting Rules

Nimmt man an es existiert ein Lerndatensatz L bestehend aus N_L Fällen

$$\left(\mathbf{x}_1, y_1\right), \ldots, \left(\mathbf{x}_{N_L}, y_{N_L}\right)$$

mit dessen Information ein Klassifikator $d\left(\mathbf{x}\right)$ erzeugt wurde der jedem $\mathbf{x} \in X$ ein $y \in Y$ zuordnet, kann für einen bekannten Datensatz der Größe N der quadratische Fehler $R\left(d\right)$ bezüglich eines Datensatzes Y' definiert werden über:

$$R\left(d\right) = \frac{1}{N}\sum_{n=1}^{N}\left(y'_n - d\left(x_n\right)\right)^2 \tag{2.7}$$

Der Versuch einen Ausdruck der Art $\sum_n \left(y_n - a\right)$ zu minimieren führt auf die Bedingung $a = \frac{1}{N}\sum_n y_n$. Analog lässt sich ein $y\left(t\right)$ finden, das $R\left(d\right)$ minimiert. Dabei

ist $y(t)$ das Mittel der y_n die für (x_n, y_n) *Node* t zugeordnet werden.

$$\bar{y}(t) = \frac{1}{N(t)} \sum_{\mathbf{x}_n \in t} y_n \tag{2.8}$$

Daraus folgt ein Ausdruck $s^2(t)$, als Summe der quadrierten Abweichungen von ihrem Mittelwert aller y_n in einem *Node* t:

$$s^2(t) = \frac{1}{N(t)} \sum_{\mathbf{x}_n \in t} (y_n - \bar{y}(t))^2 \tag{2.9}$$

Definiert man weiter den Anteil der in *Node* t gewanderten Fälle über:

$$p(t) = \frac{N(t)}{N} \tag{2.10}$$

so lässt sich die optimale Verzweigung von *Node* t in die *Nodes* t_L und t_R über das Minimum der gewichteten Varianzen ermitteln:

$$min\left(p_L \cdot s^2(t_L) + p_R \cdot s^2(t_R)\right) \tag{2.11}$$

Aus dieser Optimierung erhält man letzlich den Klassifikator $d(\mathbf{x})$ als eine stufenweise konstante Funktion im m-dimensionalen Merkmalsraum. Die Funktionswerte dieses Klassifikators $d(\vec{x})$ ergeben sich durch das Ergebnis der Minimierung in 2.11 an jedem Node. In Abbildung 2.4 ist ein Beispiel illustriert das einen *Regression-Tree* und seinen zugehörigen Klassifikator für einen 2-dimensionalen Merkmalsraum zeigt.

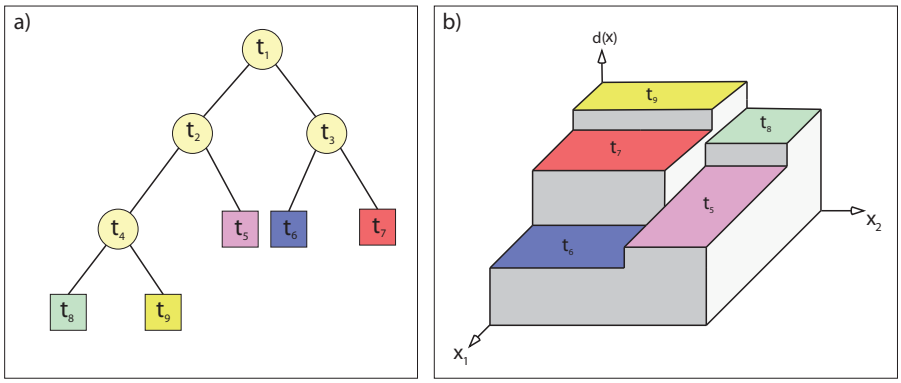

Abbildung 2.4: Illustration eines *Regression-Trees* a) und des zugehörigen Klassifikator $d(\vec{x})$ b) (modifiziert nach Breiman [6])

2.1.4 Random-Forest

Der *Random-Forest*-Algorithmus wurde von Leo Breiman entwickelt und basiert auf dem in den letzten beiden Abschnitten besprochenen Konzept der Entscheidungsbäume. Eine ausführliche Beschreibung würde den Rahmen dieser Arbeit sprengen, weswegen hier nur das Prinzip des *Random-Forest* erläutert werden soll und ansonsten auf Breiman [7], Hastie et al. [13] oder Bishop [5] verwiesen wird.

Bootstrap Aggregation

Bootstrap Aggregation oder *Bagging* wurde von Breiman [8] vorgeschlagen um die Aussagekraft von Klassifikatoren durch zufällig generierte Trainingssets zu erhöhen. Die Idee dahinter ist, da einfache Klassifikatoren zum *Overfitting* der Daten neigen, den Output aus vielen zufälligen Trainingsbeispielen zu mitteln. Dazu wird aus einem Trainingsset der Größe N durch N'-maliges ziehen mit zurücklegen ein zufälliger Trainingssatz der Größe $N' \leq N$ ausgewählt [9].

Training

Im *Random-Forest* wird eine Vielzahl von Entscheidungsbäumen erzeugt. Das prinzipielle Vorgehen beim Training ist dabei wie folgt:

1. Größe des Lerndatensatzes sei N, die Anzahl der Variablen M
2. Wähle einen Trainingssatz der Größe N, durch zufälliges ziehen mit zurücklegen, aus der Menge des Lerndatensatzes aus (*Bootstrapping*).
3. Wähle zur Bestimmung jeder *Node*-Verweigung (Siehe Splitting-Rules in 2.1.3) zufällig m aus M ($m << M$) Variablen aus. Die Anzahl der zufälligen Variablen m bleibt dabei während der Erzeugung des gesamten Waldes konstant.
4. N_T Bäume werden voll ausgebaut. Hierfür werden die Schritte 2 und 3 N_T-mal wiederholt.

Die Größen m und N_T sind dabei die einzig einstellbaren Parameter im *Random-Forest*. Die Genauigkeit hängt dabei im Wesentlichen von zwei Faktoren ab; der Korrelation der einzelnen Bäume untereinander und der Stärke jedes einzelnen Baumes. Diese beiden Faktoren sind Gegenspieler, denn eine wachsende Stärke der Bäume, erreicht durch eine Erhöhung von m, führt zu einer weniger fehlerbehafteten Klassifikation, aber gleichzeitig steigt mit größerem m die Korrelation der Bäume untereinander und damit auch der Fehler. Ein optimales Maß für die Größe m lässt sich aber durch den sogenannten *out-of-bag-error* finden. N_T hat natürlich Einfluss auf die Geschwindigkeit des Algorithmus und seine Genauigkeit.

Out-Of-Bag-Error

Bei dem zufälligen Prozess der Trainigsdatenauswahl (*Bootstrapping*) bleibt im Mittel ein Drittel der N Lernbeispiele unbenutzt. Die Wahrscheinlichkeit, dass sich ein Datenpunkt x im gewählten Ensemble B befindet, berechnet sich nach:

$$P\left(x \in B\right) = 1 - \left(1 - \frac{1}{n}\right)^{n} \approx 0.632 \quad \text{für genügend großes n} \tag{2.12}$$

Diese unbenutzte Datenmenge wird *out-of-bag-data* genannt. Anhand dieser Daten testet sich der *Random-Forest* selbst auf die Güte seiner Klassifikation, indem er seine Vorhersage für jeden *out-of-bag*-Datenpunkt mit seiner vom Nutzer vorgegebenen Klasse vergleicht. Die dabei vom Algorithmus ermittelte prozentuale Fehlerquote ist der *out-of-bag-error* [9].

Vorhersage

Vorhersagen trifft der *Random-Forest*, indem ein Datenpunkt $\mathbf{x}_i$ jedem der N_T Bäume zur Klassifizierung gegeben wird. Dabei ist letztlich die am häufigsten gewählte Klassenzuweisung die für $\mathbf{x}_i$ getroffene Vorhersage γ_i. Je größer der Anteil dieser Klasse aus den N_T Entscheidungen ist, umso sicherer ist die Aussage.

2.2 Bildverarbeitung

In diesem Abschnitt werden die Grundlagen der in dieser Arbeit verwendeten Methoden der Bildverarbeitung erläutert. In späteren Kapiteln kommt eine am HCI[1] entwickelte Software zur interaktiven Segmentierung zum Einsatz, deren Grundprinzipien auf den nachfolgenden Seiten behandelt werden.

2.2.1 Diskrete Faltung

Definition Nachbarschaftsoperation: Betrachtet wird ein Bild $\mathbf{G}$ bestehend aus den Grauwerten g_{mn}. Nachbarschaftsoperationen ändern den Grauwert g_{mn} eines Pixel von $\mathbf{G}$ in Abhängigkeit der Grauwerte eines Gebiets um g_{mn}. Nachbarschaftsoperationen werden mittels Operatoren $A, B, \ldots$ dargestellt:

$$\mathbf{G}' = A\mathbf{G} \tag{2.13}$$

Die mathematische Operation Grauwerte in einer Umgebung gewichtet aufzusummieren wird als Faltung bezeichnet.

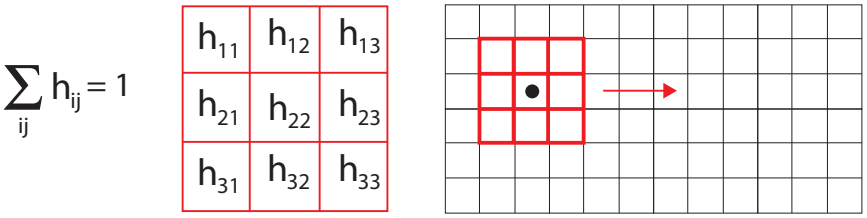

Abbildung 2.5: Veranschaulichung einer diskreten Faltung (modifiziert nach Jähne [19])

Definition Faltung: Betrachtet man ein diskretes Bild $\mathbf{G}$ aus den Grauwerten g_{mn} und eine Faltungsmaske h_{mn} mit $m, n \in [-r, r]$, dann ist eine diskrete Faltung definiert als:

$$g'_{mn} = \sum_{k=-r}^{r} \sum_{l=-r}^{r} h_{kl} \cdot g_{m-k,n-l} \tag{2.14}$$

Häufig ist dabei erwünscht, dass die Koeffizienten der Faltungsmaske eine Summe von Eins ergeben, da sonst die Gefahr des Überschreitens der maximalen Grauwerte besteht. Filter dieser Art, die linear und verschiebungsinvariant sind, nennt man LSI-Filter (*Linear Shift-Invariant*).

[1] Heidelberg Collaboratory for Imageprocessing

Desweiteren gibt es mehrere Möglichkeiten den Rand des Bildes zu behandeln, da der Maske in den Randgebieten Pixel zur Auswertung fehlen:

1. Die exakte Lösung wäre die Randpixel bei der Faltung zu vernachlässigen. Das Problem dabei ist allerdings, dass das Bild nach Anwendung kleiner ist, was bei großen Filtern oder mehrmaliger Faltung unerwünscht sein kann.

2. Eine weitere Möglichkeit ist alles außerhalb des Bildes Null zu setzten. Dies führt aber unter Umständen zu großen Fehlern (Siehe Abbildung 2.6 a).

3. Setzt man die Bildränder zyklisch fort entstehen bei Bildern mit nicht konstanten Grauwerten an allen Rändern ebenfalls Artefakte. (Abbildung 2.6 b).

4. Die besten Ergebnisse lassen sich durch gespiegelte Randbedingungen erreichen. Dabei dient jeder Rand als Spiegelachse (Siehe Abbildung 2.6 c).

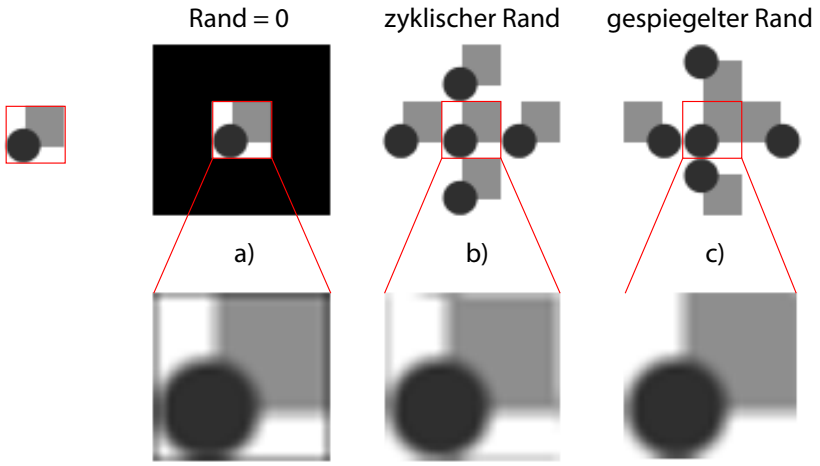

Abbildung 2.6: Faltung mit unterschiedlichen Randbedingungen unter Anwendung eines Glättungsfilters. In a) Null Rand, b) zyklische Fortsetzung, c) gespiegelte Fortsetzung

In Abbildung 2.6 ist anhand eines Testbilds das Auftreten von Artefakten bei unterschiedlichen Behandlungen des Randes nach einer Faltung dargestellt. Es ist leicht zu erkennen, dass jede Art der Randbehandlung zu unterschiedlichen Resultaten führt. Dies verdeutlicht, dass es von Bedeutung sein kann, welche der Methoden im Einzelfall gewählt wird.

2.2.2 Punktantwort und Transferfunktion

Ein Bild der Größe $M \times N$ lässt sich als Linearkombination aus $M \cdot N$ „Delta-Bildern“, also Bilder, die nur einen Punkt enthalten der von Null verschieden ist, darstellen.

Dafür definiert man analog zur Deltafunktion ein Basisbild:

$$ {}^{mn}\mathbf{P} : \ {}^{mn}p_{m'n'} = \begin{cases} 1 & m = m' \wedge n = n' \\ 0 & \text{sonst} \end{cases} \tag{2.15} $$

Ein beliebiges Bild $\mathbf{G}$ lässt sich dann darstellen als:

$$ \mathbf{G} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g_{mn} \, {}^{mn}\mathbf{P} \tag{2.16} $$

Für einen LSI-Filter H kann man daher schreiben:

$$ H\mathbf{G} = \sum_{m} \sum_{n} g_{mn} \left(H^{mn}\mathbf{P} \right) \tag{2.17} $$

Definiton Punktantwort (PSF): Die Punktantwort oder PSF (*point-spread-function*) ist das Resultat eines LSI-Filters angewandt auf ein Basisbild ${}^{mn}\mathbf{P}$.

Nach Gleichung 2.17 ist das Ergebnis eines Filters also vollständig durch seine Punktantwort bestimmt. Nach der Theorie der Fouriertransformation kann man zeigen, dass eine Faltung im Ortsraum im Fourierraum in eine punktweise Multiplikation übergeht (siehe Jähne [19]). Daraus folgt ein zur Punktantwort analoges Interesse für deren Fouriertransformierte.

Definition Transferfunktion (TF): Die Fouriertransformierte der Punktantwort eines LSI-Filters H bezeichnet man als Transferfunktion. Sie charakterisiert den Einfluss des Filters auf die im Bild vorhandenen Frequenzanteile.

Die Transferfunktion ist im allgemeinen eine komplexe Funktion, so dass durch sie bestimmte Frequenzanteile skaliert und in Ihrer Phase verschoben werden können. Daher spiegeln sich in ihr die Symmetrien der LSI-Filter wieder. Reelle Filter mit gerader Symmetrie haben reelle Transferfunktionen und Filter mit ungerader Symmetrie rein imaginäre.

2.2.3 Mittelung

Mittelungsfilter haben die Funktion schnelle Grauwertänderungen zu dämpfen und können deshalb auch als Tiefpassfilter betrachtet werden.

Eigenschaften

- Da ein Mittelungs- oder Glättungsfilter H die Position einer Kante im Bild nicht verändern darf, sollte seine Transferfunktion ($\hat{h}_{\nu,\mu}$) reell sein beziehungsweise die PSF ($h_{m,n}$) symmetrisch.

- Der Mittelwert muss erhalten bleiben, deshalb muss die Summe der Filterkoeffizienten auf Eins normiert sein.

$$\sum_m \sum_n h_{m,n} = 1$$

- Steigende Wellenzahlen sollen gedämpft werden, deshalb muss die TF monoton gegen Null fallen.

$$\hat{h}\left(k_2\right) \leq \hat{h}\left(k_1\right) \quad \text{für} \quad k_2 > k_1$$

- Im Allgemeinen soll die Glättung auch richtungsunabhängig sein, dass heißt PSF und Transferfunktion sind isotrop.

$$h\left(\vec{x}\right) \equiv h\left(|\vec{x}|\right) \quad \text{und} \quad \hat{h}\left(\vec{k}\right) \equiv \hat{h}\left(\left|\vec{k}\right|\right)$$

Rechteckfilter

Die einfachste Konstruktion eines Mittelungsfilters ist der Rechteckfilter R:

$$A_R = \frac{1}{\sum_{ij} h_{ij}} \begin{pmatrix} h_{11} & \dots & h_{1j} \\ \vdots & \ddots & \vdots \\ h_{i1} & \dots & h_{ij} \end{pmatrix} \qquad \text{mit:} \quad h_{ij} = 1 \quad \forall i,j \tag{2.18}$$

Dieser hat aber mehrere Schwächen. Die Fouriertransformierte eines Rechtecksignals ist im eindimensionalen die Sinc-Funktion. Diese oszilliert um die Null und fällt nur sehr langsam (mit 1/k) ab. Stellen, an denen die Transferfunktion kleiner als Null wird, entsprechen einer Phasenverschiebung von Pi. Daher ist der Rechteckfilter weder nullphasig noch isotrop und auch kein guter Tiefpass.

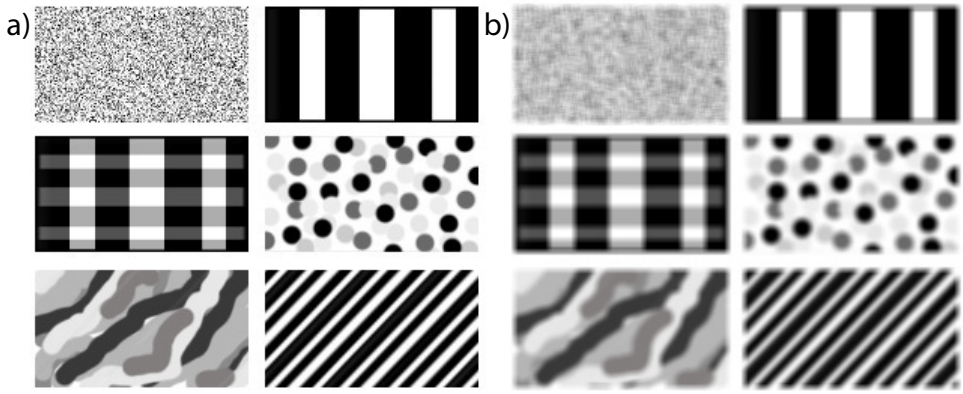

Abbildung 2.7: a) Testbild, b) mit Boxfilter geglättet

Gauß-, Binomialfilter

Filter mit einer Binomialverteilung sind wesentlich bessere Glättungsfilter. Sie ergeben sich aus der Faltung zweier eindimensionaler Binomialfilter, deren Einträge aus dem Schema des *Pascalschen Dreiecks*, normiert auf eine Summe von Eins, folgen. Ein Beispiel für einen 3×3 Binomialfilter ist:

$$A_G = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Diese Art Filter erfüllt das Monotonie-Kriterium und ihre Transferfunktion ist reellwertig und nahezu isotrop.

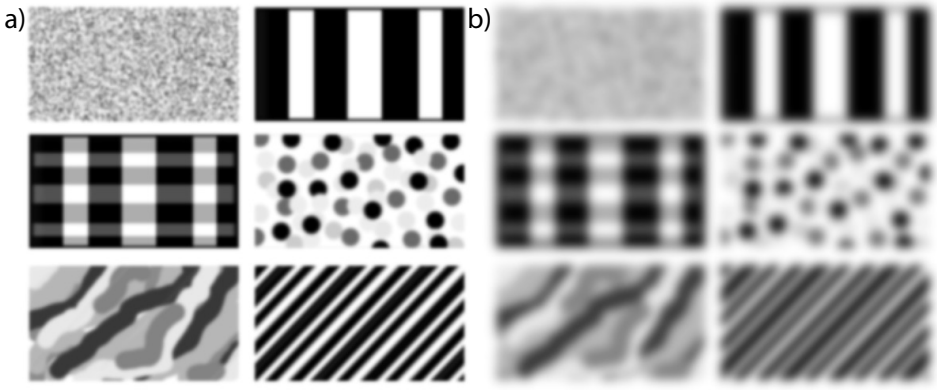

Abbildung 2.8: Testbild aus Abbildung 2.7 mit verschieden großen Gaussfiltern geglättet: a) $\sigma = 1.0$, b) $\sigma = 3.5$

2.2.4 Kanten Detektion

Im Folgenden soll nur die Detektion von Kanten in einem Bild mittels des *Canny-Algorithmus* erklärt werden, da dieser später zur Anwendung kommt. Für eine detaillierte Beschreibung der Kanten-Detektion sei auf Jähne [19] verwiesen.

Sobel-Oberator

Der *Sobel-Operator* besteht aus 2 Operatoren, S_x und S_y. Dabei werden jeweils in x-, beziehungsweise in y-Richtung die ersten Ableitungen berechnet und damit Bereiche starker Grauwertänderung detektiert. Die Resultate (G_x, G_y) der Faltungen eines Bildes g mit dem *Sobel-Operator* sind also Gradientenbilder in den jeweiligen Richtungen.

$$G_x = Sx \cdot g = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} g \tag{2.19}$$

$$G_y = Sy \cdot g = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} g \tag{2.20}$$

Canny-Edge-Detektor

Kantendetektion mittel des *Canny-Edge-Detektors* besteht aus mehreren Schritten:

1. Glättung des Bildes durch einen Gaußfilter um die Sensitivität auf Rauschen im Bild zu verringern.

2. Anwendung des *Sobel-Operators* auf das geglättete Bild ergibt die Gradientenbilder G_x und G_y, sowie die absolute Kantenstärke

   $$|G| = |G_x| + |G_y|$$

3. Bestimmung der Kantenrichtung über

   $$\Theta = tan^{-1}\left(\frac{G_Y}{G_x}\right)$$

   Da für jedes Pixel aber nur 8 benachbarte Pixel existieren, ergeben sich nur 4 Kantenrichtungen: 0°, 45°, 90° und 135° auf die gerundet wird.

4. Im nächsten Schritt werden die Kanten auf maximal einen Pixel Breite reduziert. Dies wird dadurch erreicht, dass der Absolutwert des Gradienten jedes Pixels mit seinen Nachbarn verglichen wird. Dabei darf kein angrenzender Pixel einen größeren Wert aufweisen, es sei, denn er liegt in der selben Kantenrichtung. Wird die Bedingung nicht erfüllt, wird der Grauwert des Nachbarpixels auf Null gesetzt.

5. Im letzten Schritt wird über einen Schwellwert T bestimmt, ab welchem berechneten Wert der Pixel als Kante gezählt werden soll. Dabei wird, um eine Unterbrechung der Kanten durch Schwankungen nahe dem Schwellwert zu vermeiden, ein Hysterese genanntes Verfahren mit zwei Schwellwerten $T_2 > T_1$ eingesetzt. Wird ein Pixelwert größer als T_2 gefunden, läuft der Algorithmus entlang der Kante und markiert alle Werte größer T_1 als solche, alle anderen werden Null gesetzt.

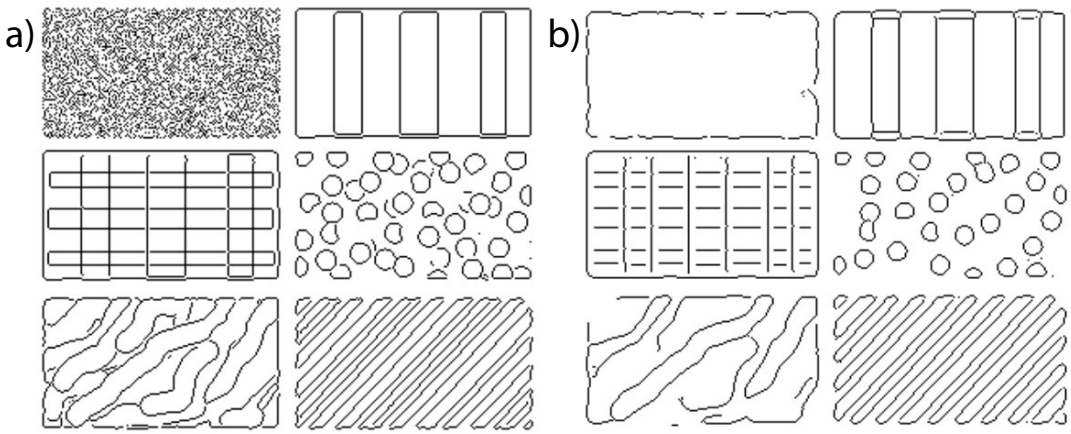

Abbildung 2.9: a) Testbild aus Abbildung 2.7 nach Canny-Edge-Verfahren mit geringer Glättung $\sigma = 0.2$, $T_2 = 9.0$, b) mit stärkerer Glättung $\sigma = 2.5$, $T_2 = 9.0$

In Abbildung 2.9 wird das Ergebnis des *Canny-Algorithmus* anhand des Testbildes aus Abbildung 2.7 mit unterschiedlich starker Glättung demonstriert. Daraus wird oben links ersichtlich wie wichtig das Vermindern von Rauschen vor der Kantendetektion ist.

2.2.5 Textur

In diesem Abschnitt sollen nach einer kurzen Einführung im Wesentlichen nur die Filter besprochen werden, die zur Charakterisierung von Texturen in Ilastik (Kapitel 2.3) verwendet werden. Die Analyse von Texturen in der Bildverarbeitung ist bei weitem vielfältiger. Für eine tiefergehende Einführung wird auf Jähne [19] verwiesen.

Einfache Nachbarschaften

Eine erste Herangehensweise an Strukturen in einem Bild ist die Definition einfacher Nachbarschaften.

Definition Einfache Nachbarschaft: Eine einfache Nachbarschaft ist eine lokale Umgebung, deren Grauwerte sich nur in einer Richtung ändern.

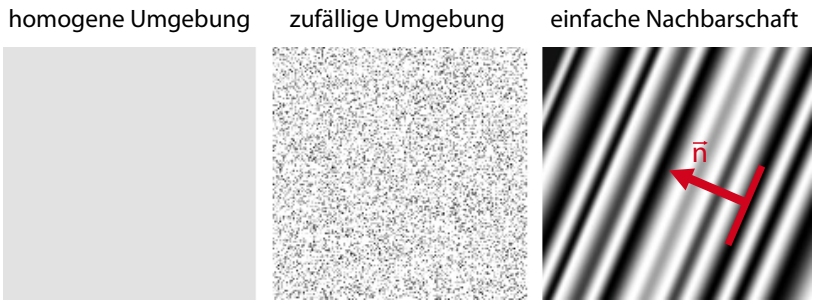

Abbildung 2.10: Mögliche einfache Nachbarschaften (modifiziert nach Jähne [19])

In einer einfachen Nachbarschaft kann eine ausgezeichnete Richtung durch einen Normalenvektor $\vec{n}$ angegeben werden.

Durch eine Vektordarstellung lassen sich einfache Nachbarschaften farbcodiert darstellen. Dabei wird jedem Pixel eine Orientierung durch den Vektor $\vec{n}$ über einen Farbwert zugewiesen und ein Maß für die Bestimmtheit der Orientierung über die Helligkeit. Die Bestimmtheit ist dabei der Betrag der resultierenden Vektorsumme aller Orientierungen einer Umgebung. Daraus folgt, dass zufällige Umgebungen ein kleines Bestimmtheitsmaß aufweisen, orientierte Umgebungen dagegen ein hohes. Nachteil dieser Methode ist, dass nicht zwischen unkorreliertem Rauschen und konstanten Umgebungen unterschieden werden kann. Deshalb bietet sich die Verwendung einer tensoriellen Größe zur Beschreibung der Umgebung eines Pixels an.

Strukturtensor

Die Idee des Strukturtensors basiert auf der Optimierungsaufgabe ein $\vec{n}$ zu finden, das am besten zur Gradientenrichtung des Grauwerts g passt. Entgegengesetz orientierte Gradienten sollen dabei gleichberechtigt sein, da eine Spiegelung eines orientierten Musters keine Veränderung bewirkt.

Daher ist der folgende Ausdruck zu optimieren.

$$\epsilon = \left(\nabla g^T \vec{n}\right)^2 = |\nabla g|^2 \, cos^2 \left(\angle(\nabla g, \vec{n})\right) \tag{2.21}$$

Das ϵ ist dabei maximal wenn Gradient und Orientierungsvektor parallel oder antiparallel sind. Daraus ergibt sich die Maximierung von:

$$\int w\left(\vec{x} - \vec{x}'\right) \left(\nabla g\left(\vec{x}'\right)^T \vec{n}\right)^2 d^W x' \tag{2.22}$$

Dabei wird durch w eine Fensterfunktion eingeführt, die Größe und Form der Umgebung um den Punkt $\vec{x}$ definiert.
Daraus folgt durch Umformungen eine Darstellung des Maximierungsproblems derart, dass:

$$\vec{n}\left(\mathbf{J}\vec{n}\right) \rightarrow max$$

und damit die Definition des Strukturtensors $\mathbf{J}$:

$$J_{pq}\left(\vec{x}\right) = \int w\left(\vec{x} - \vec{x}'\right) \left(\frac{\partial g\left(\vec{x}'\right)}{\partial x'_p} \cdot \frac{\partial g\left(\vec{x}'\right)}{\partial x'_q}\right) d^W x' \tag{2.23}$$

Da der Strukturtensor symmetrisch ist, kann man ihn mit den Mitteln der linearen Algebra auf Diagonalgestalt bringen und erhält über den dafür nötigen Drehwinkel den Orientierungswinkel Θ, den Orientierungsvektor $\vec{\sigma}$ und das Kohärenzmaß c_c nach:

$$tan2\Theta = \frac{2J_{12}}{J_{22} - J_{11}} \tag{2.24}$$

$$\vec{\sigma} = \begin{pmatrix} J_{22} - J_{11} \\ 2J_{12} \end{pmatrix} \tag{2.25}$$

$$c_c = \frac{\sqrt{\left(J_{22} - J_{11}\right)^2 + 4J_{12}^2}}{J_{11} + J_{22}} \tag{2.26}$$

Die Länge des Vektors $\vec{\sigma}$ gibt dabei das Bestimmtheitsmaß der lokalen Orientierung an. Das Kohärenzmaß bildet auf das Intervall $[0, 1]$ ab. Ein Wert von Eins steht dabei für eine ideale Orientierung, die Null steht für eine vollständig isotrope

Grauwertstruktur. Zur Texturanalyse werden in dieser Arbeit die Eigenwerte des Strukturtensors verwendet, deren Bedeutung im folgenden aufgelistet ist:

Beziehung	Rang(J)	Bedeutung
$\lambda_1 = \lambda_2 = 0$	0	konstante lokale Umgebung
$\lambda_1 > 0, \lambda_2 = 0$	1	einfache Nachbarschaft mit idealer Orientierung
$\lambda_1 > 0, \lambda_2 > 0$	2	Grauwerte ändern sich in jeder Richtung. Für $\lambda_1 = \lambda_2$ vollständig isotrope Struktur

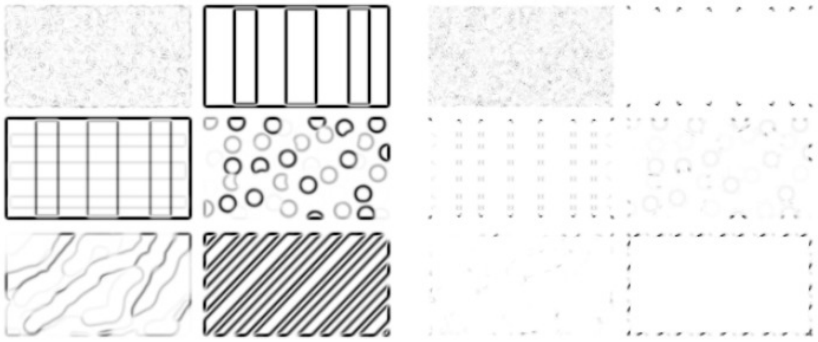

Abbildung 2.11: Eigenwerte des Strukturtensors für das Testbild aus Abbildung 2.7 bei vorheriger Glättung mit $\sigma = 1.0$. Zur besseren Darstellbarkeit wurden diese Bilder invertiert.

Absolutbetrag des Gradienten

Als weiteres Texturmerkmal wird der Absolutbetrag des Gradienten verwendet. Dabei wird aus den gaußgeglätteten Bildern die Ableitung in beiden Richtungen mittels eines Ableitungsoperators (siehe Abschnitt 2.2.4) bestimmt und der Absolutbetrag für jedes Pixel ausgegeben.

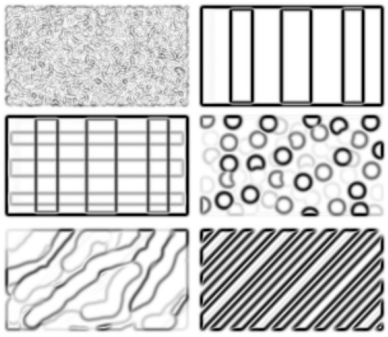

Abbildung 2.12: Absolutbetrag des Gradienten für Testbild 2.7 bei Glättung mit $\sigma = 1.0$. Zur besseren Darstellbarkeit wurde das Bild invertiert.

Eigenwerte der Hessematrix

Als letztes Merkmal zur Texturcharakterisierung der Daten werden die Eigenwerte der Hessematrix H, angewendet auf ein Bild g verwendet:

$$H(g) = \left(\frac{\partial^2 g}{\partial x_i \partial x_j} \right) = \begin{pmatrix} \frac{\partial^2 g}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 g}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} \end{pmatrix} \tag{2.27}$$

Die Einträge der Hessematrix werden dabei durch Faltung mit partiell abgeleiteten Gauskernen bestimmt. Neben der einfachen Kantenerkennung steht mit den Eigenwerten der Hessematrix ein Filter zur Verfügung der auch auf Ecken reagiert. Bei einfachen Kanten wird einer der beiden Eigenwerte einen hohen Wert annehmen. Ecken im Bild zeichnen sich allerdings durch starke Änderungen in beiden Richtungen aus, weswegen an diesen Stellen beide Eigenwerte der Hessematrix einen hohen Wert aufweisen werden.

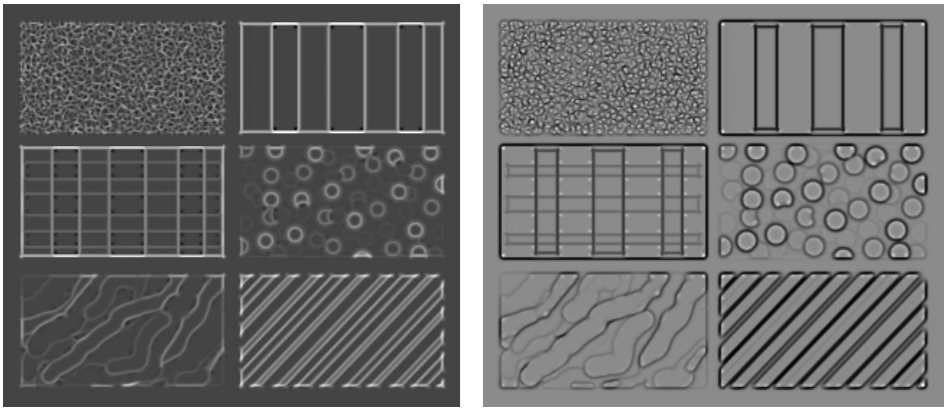

Abbildung 2.13: Eigenwerte der Hessematrix für Testbild 2.7 bei Glättung mit $\sigma = 1.0$.

2.3 Ilastik

Die Software Ilastik[1], entwickelt am HCI[2] von C. Sommer, C Strähle et al. [36], ist eine Software zur interaktiven Segmentierung in der Bildverarbeitung. Sie stellt die Verwendung eines *Random-Forest* innerhalb einer grafischen Benutzeroberfläche bereit und ermöglicht es dem Nutzer, die Definition der Trainingsdaten interaktiv durchzuführen. Verwendete Features sowie die Markierung der Daten können solange bei unmittelbarer optischer Rückmeldung der Ergebnisse angepasst werden, bis der *Random-Forest* optimale Ergebnisse liefert. Anhand des in den vorherigen Abschnitten verwendeten Testbilds soll nun kurz die Funktionsweise dieser Anwendung erläutert werden.

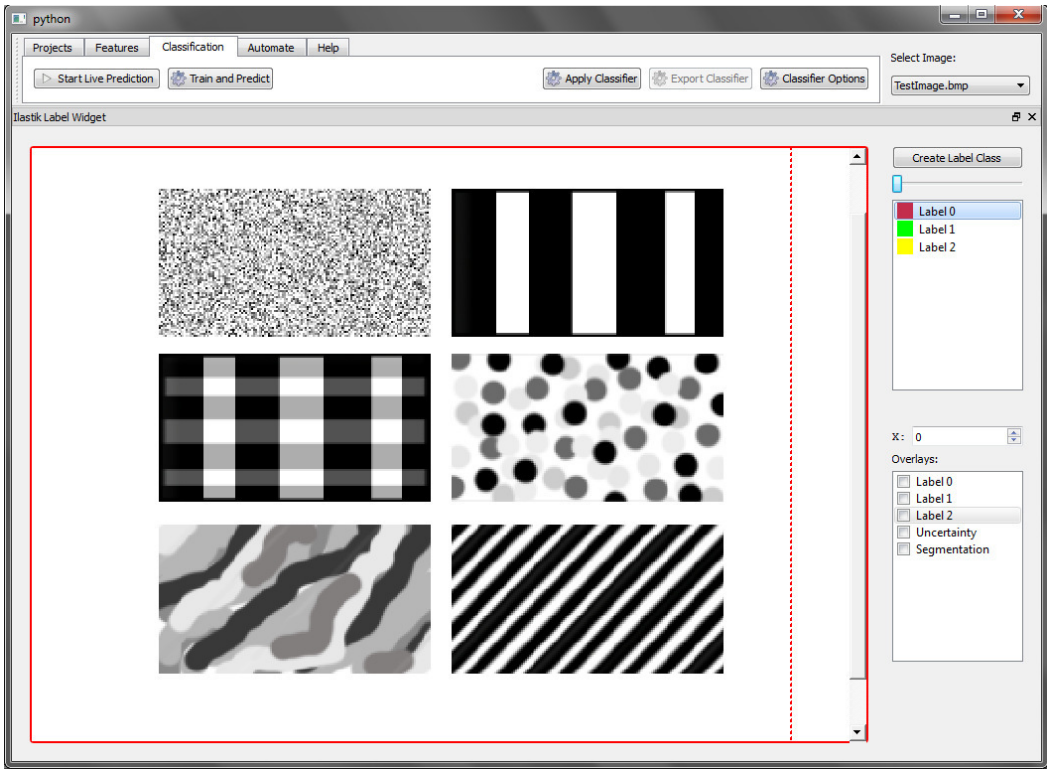

Abbildung 2.14: Ilastik GUI mit geladenem Testbild

2.3.1 Labeling

Die normale Vorgehensweise in Ilastik ist das Anlegen eines Projektes mit den gewünschten zu trainierenden Bilddaten. Zur Veranschaulichung wurde das Testbild aus Abbildung 2.7 als Projekt angelegt.

[1] http://hci.iwr.uni-heidelberg.de/download3/ilastik.php
[2] HCI-Heidelberg Collaboratory for Image Processing

Im oberen Teil der GUI (Vergleiche Abbildung 2.14) befinden sich Reiter zur Auswahl der verschiedenen Arbeitsbereiche. In „Projects“ finden sich die Verwaltung der Projekte und verschiedene Darstellungsoptionen. In „Features“ findet die Auswahl und das Berechnen der zur Klassifikation zu verwendenen Features statt. Im Reiter „Classification“ ist der Interaktivmodus, das Training des *Classifiers* und Optionen hierzu zu finden. „Automate“ bietet die Möglichkeit einen trainierten *Random-Forest* automatisiert auf beliebig viele Daten anzuwenden. In der rechten Leiste der GUI werden Anzahl der gewünschten Klassen und deren Anzeigeoptionen festgelegt. Wie in Abbildung 2.14 zu sehen ist, werden zur Demonstration 3 Klassen definiert und in Abbildung 2.15 zugewiesen.

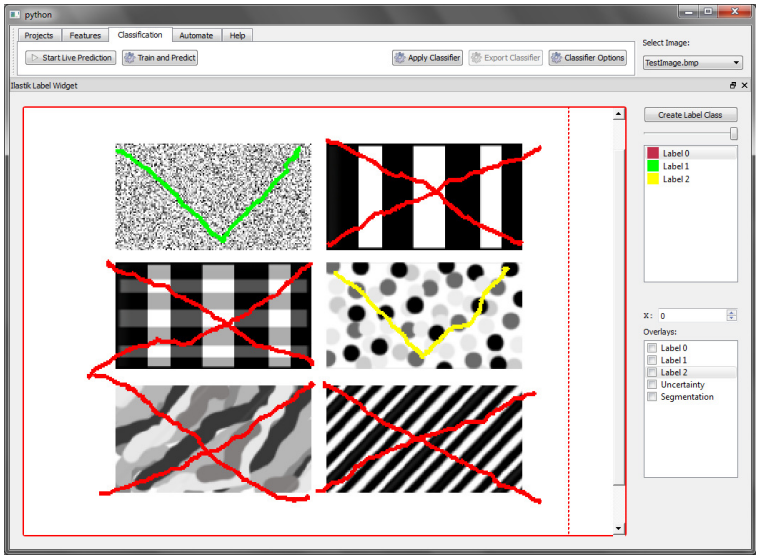

Abbildung 2.15: Labeling des Testbildes mit drei Klassen

Es soll also ein *Random-Forest* trainiert werden, um in anderen Bilddaten das Rauschen oben links (Klasse „Grün“) und das Kreismuster (Klasse „Gelb“) zu finden, wobei alles andere als uninteressant der Klasse „Rot“ zugeordnet wurde. Im nächsten Schritt wurden die gewünschten Features ausgewählt (Abbildung 2.16).

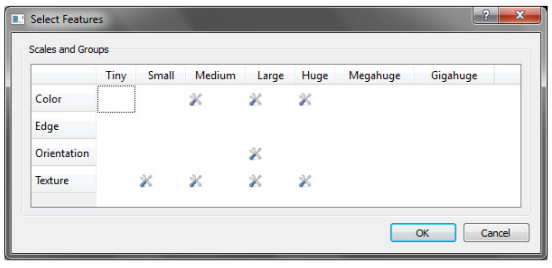

Abbildung 2.16: Featureauswahl

2.3.2 Features in Ilastik

Die Standardmäßig zur Verfügung stehenden *Features* sind:

Feature	**Verwendete Filter**
Color:	Gauß-Filter
Edge:	Laplace-Filter Gauß-Differenz-Filter
Orientation:	Hessematrix Strukturtensor
Texture:	Eigenwerte der Hessematrix Eigenwerte des Strukturtensors Absolutbetrag des Gradienten

Es können zusätzlich auch eigene *Features* geschrieben und als *Plugin* geladen werden, wovon später in Kapitel 4.3 Gebrauch gemacht werden wird.

Jedem Feature kann eine Größe zugewiesen werden, diese bezieht sich auf die Ausdehnung des Filters, wodurch dessen Einfluss auf die Nachbarschaft eines jeden Pixels gesteuert werden kann. Es muss aber immer beachtet werden, dass mit größer werdender Ausdehnung eines Filters die fehlerbehafteten Randbereiche ebenfalls größer werden.

Die Bezeichnungen *Tiny* bis *Gigahuge* entprechen den Werten in folgender Tabelle und beziehen sich in den meisten Features auf das Sigma der verwendeten Gaussmaske.

Tiny	**Small**	**Medium**	**Large**	**Huge**	**Megahuge**	**Gigahuge**
0.3	0.7	1	1.6	3.5	5.0	10.0

2.3.3 Training des Random-Forest

Sind die Features gewählt und berechnet, kann in den interaktiven Modus gewechselt werden, wonach sofort begonnen wird, einen auf Geschwindigkeit optimierten *Random-Forest* zu berechnen. Ist dies geschehen, wird das Ergebnis sofort als Overlay über den eigenen Bilddaten dargestellt (Abbildung 2.17).

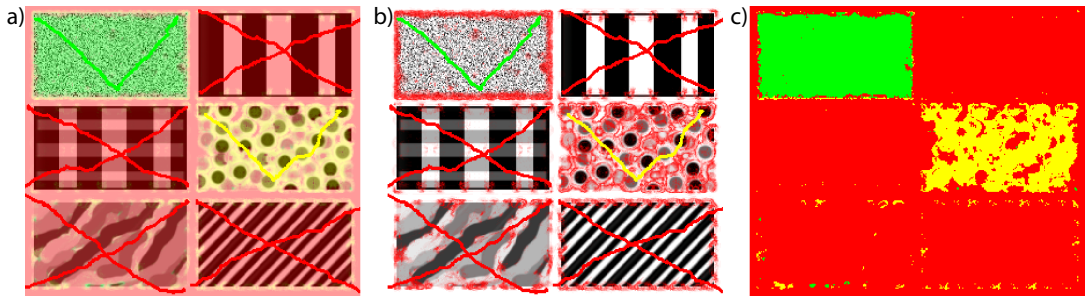

Abbildung 2.17: a) Vorhersage des Random-Forrest, b) Unsicherheiten, c) aus Vorhersagende resultierende binäre Segmentierung

In Abbildung 2.17 sind die drei möglichen Darstellungen des Ergebnisses des ausschließlich mit den im Bild markierten Pixeln trainierten *Random-Forest* zu sehen. Die Möglichkeit, sich die Bereiche hoher Unsicherheiten (Abbildung 2.17 b) des Algorithmus anzeigen zu lassen, ist im Interaktiv-Modus äußerst hilfreich, da sich so sehr gezielt Korrekturen im Trainingsdatensatz hinzufügen lassen. Außerdem wird bei jeder Neuberechnung des *Random-Forest* der *Out-Of-Bag-Error* ausgegeben, was die Wahl der passenden Features sehr komfortabel macht.
Ist man mit dem Ergebnis zufrieden, wird der *Random-Forest* über „Train and Predict“ in seiner vollständigen Genauigkeit ausgebildet und kann danach entweder exportiert und in anderen Projekten verwendet oder direkt im *Batch-Process* auf beliebig viele vorhandene Daten angewendet werden. Die Ausgabe erfolgt dabei im „HDF5“[1] Format. In diesem finden sich nach dem Batch-Process neben den Orginaldaten auch die Vorhersagen für jede Klasse als Wahrscheinlichkeitskarte.
Für das zur Veranschaulichung verwendete Beispiel ergab die Anwendung des vollständig trainierten *Random-Forest* auf ein nicht im Trainingsatz vorhandenes Bild folgende Klassifizierung:

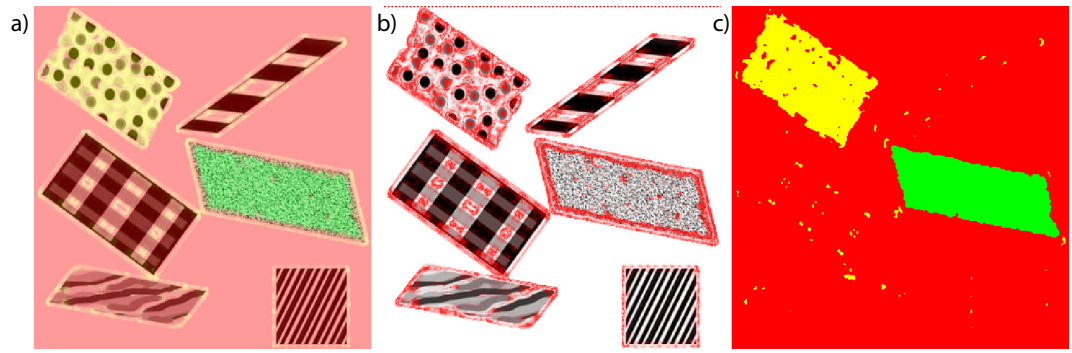

Abbildung 2.18: Ergebnis des Random-Forest bei Anwendung auf nicht trainiertes Bild

[1] http://www.hdfgroup.org/HDF5/

Man erkennt, dass die Segmentierung, trotz der ausgeprägten Unsicherheiten, recht gut die beiden unterschiedlichen Texturen lokalisiert und bei einem *Out-Of-Bag-Error* von knapp über 2% ein akzeptables Ergebnis liefert. Man sieht aber auch, dass für die Kreis-Textur wohl noch nicht die optimalen Features gefunden wurden, da hier die stärksten Unsicherheiten auszumachen sind.
Ilastik ist zudem auch in der Lage mit Volumendaten umzugehen. Hierfür müssen nur statt Trainingsbildern Volumen im H5-Format erzeugt und einem Projekt hinzugefügt werden. Diese Methode kommt in Kapitel 4.3 zum Einsatz.

2.4 Microscale Breaking Waves

Microscale-Breaking-Waves (siehe Banner und Phillips [4]) sind kleinskalige brechende Wellen (ab $\lambda = 5\,\mathrm{cm}$) die schon bei relativ niedrigen Windgeschwindigkeiten auftreten. Im Gegensatz zu den allgemein bekannten brechenden Wellen im Ozean, die sich durch ausgeprägte Schaumkronen und starken Blaseneintrag auszeichnen (typischerweise ab $\lambda > 1m\,\mathrm{m}$), tritt bei *Microscale-Breaking-Waves* kein Eintrag von Luftblasen auf. Nach Melville [26] gehen aber mit ihrem Auftreten oberflächennahe Turbulenzen einher, die zu Energiedissipation und einem erhöhten Gaseintrag führen (siehe Abbildung 2.19). Durch die Turbulenzen wird Wasser aus tieferen Regionen an die Oberfläche transportiert was durch passive oder aktive Thermographie [15] zu deren Detektion genutzt werden kann (siehe auch Kapitel 2.5.2 und Kapitel 4.3.4). Zudem tritt dabei häufig eine, mit dem Auge erkennbare, charakteristische Struktur an der Wasseroberfläche auf (Vergleiche Abbildung 4.17), die in dieser Arbeit ebenfalls zur Detektion mikrobrechender Wellen verwendet werden soll.

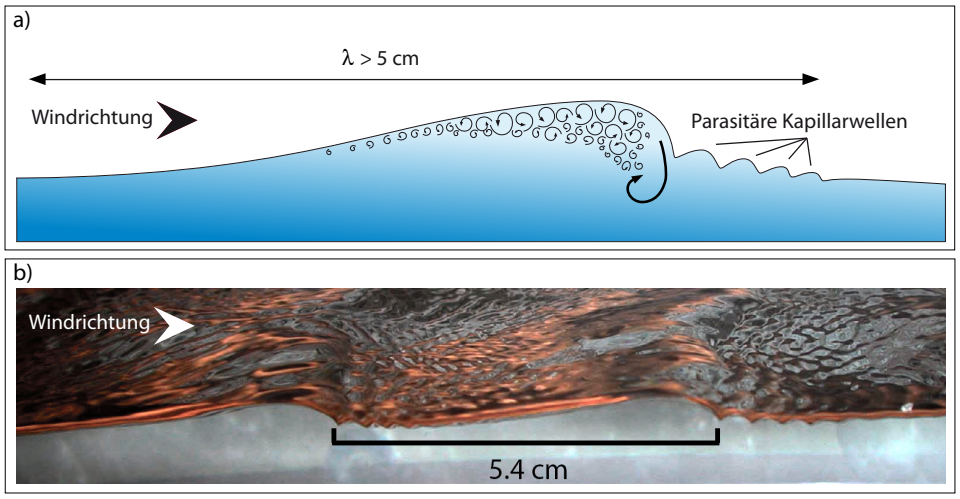

Abbildung 2.19: a) Darstellung der Turbulenzen einer *Microscale Breaking* Welle, b) Aufnahme aus Wind-Wellen-Kanal (modifiziert nach Huhn [18])

In einem idealisierten Modell einer unbewegten Wasseroberfläche stellt sich eine Grenzschicht und mit ihr ein Konzentrationsgefälle zwischen gesättigter Luft- und

Wasserphase ein, an der Gastransport ausschließlich über die sehr langsame molekulare Diffusion stattfinden kann (siehe Jähne [20]). Diese Grenzschicht hat eine Dicke von circa $20 - 200\,\mu$m. Wird diese Grenzschicht durch Material aus den tieferen, gut durchmischten Regionen ausgetauscht, wird sich der Gastransport bis zum erneuten Einstellen der Sättigung beschleunigen. Die Bedeutung von *Microbreaking* in Bezug auf den Gasaustausch zwischen Atmosphäre und Ozean liegt in ihrem Einfluss auf die, den Gasfluss kontrollierende, laminare Grenzschicht. Nach Zappa et al. [39, 40] zeigen Laborexperimente in Wind-Wellen-Kanälen, dass durch *Microbreaking* induzierte Turbulenzen einen signifikanten Einfluss auf die Transfergeschwindigkeiten[1] bei moderaten[2] Windgeschwindigkeiten haben. Möglichkeiten zur Detektion sind neben der bereits genannten Thermographie (Atman et al. [2], Hausecker et al. [14] und Kapitel 2.5.2) auch PIV - *Particle Image Velocimetry* (siehe Chang und Liu [10] oder Fincham und Spedding [11]) oder die Wellenneigung. Die Thermographie nutz dabei Wärme, sowohl passiv als auch aktiv. Diese dient als Tracer um Umwälzungen, die auf oberflächenerneuernde Turbulenzen hinweisen, zu detektieren. PIV ermöglicht eine Rekonstruktion von Strömungsfeldern. Anhand von Aufnahmen ins Wasser eingebrachter Partikel lassen sich Rückschlüsse auf die Turbulenzverhältnisse in einer Welle ziehen und damit ebenfalls *Microbreaking* detektieren. Bei der Wellenneigungsmethode wird mittels CISG (siehe Abschnitt 2.5.1) ab einer gewissen Stärke der Wellenneigung ein Wellenzug als brechend definiert. In Loewen und Siddiqui [25] werden diese Methoden miteinander verglichen.

2.5 Messmethoden

2.5.1 CISG Color-Image-Slope-Gauge

In diesem Abschnitt soll kurz erläutert werden, mit welcher Methode die in dieser Arbeit zum interaktiven Rendern einer realen Wasseroberfläche verwendeten Daten erzeugt wurden. Es gibt viele Möglichkeiten die Wellen einer Wasseroberfläche zu messen, allerdings erweisen sich die meisten als zu langsam oder zu ungenau, um zeitlich beziehungsweise räumlich hoch aufgelöste Daten zu erhalten. Eine Methode, mit der beides gleichzeitig zu erreichen ist, wurde von Rocholz [29] verwendet, um in Wind-Wellen-Kanälen mit einer Frequenz von 312 Hz und einer räumlichen Auflösung der Wellenzahlen im Bereich von $60 - 4500\,\mathrm{rad/m}$ die Neigungen der Wasseroberfläche zu bestimmen. Der Name *Color-Image-Slope-Gauge* kommt daher, dass die Neigungen der Oberfläche über eine Farbkodierung bestimmt werden. Der Aufbau besteht dabei aus zwei Komponenten, einer Farbkamera und einer farbcodierten Lichtquelle. Die Kamera befindet sich über der Wasseroberfläche, schaut vertikal auf diese und kann als unendlich entfernt betrachtet werden. Am Boden des Wind-Wellen-Kanals befindet sich ein Glasfenster, unterhalb dessen eine Linse und die farbcodierte Beleuchtung angebracht sind.

[1] Richtung und Geschwindigkeit mit der Gase von der einen Phase in die andere gelangen.
[2] Windgeschwindigkeiten zwischen 4 und 10 m/s.

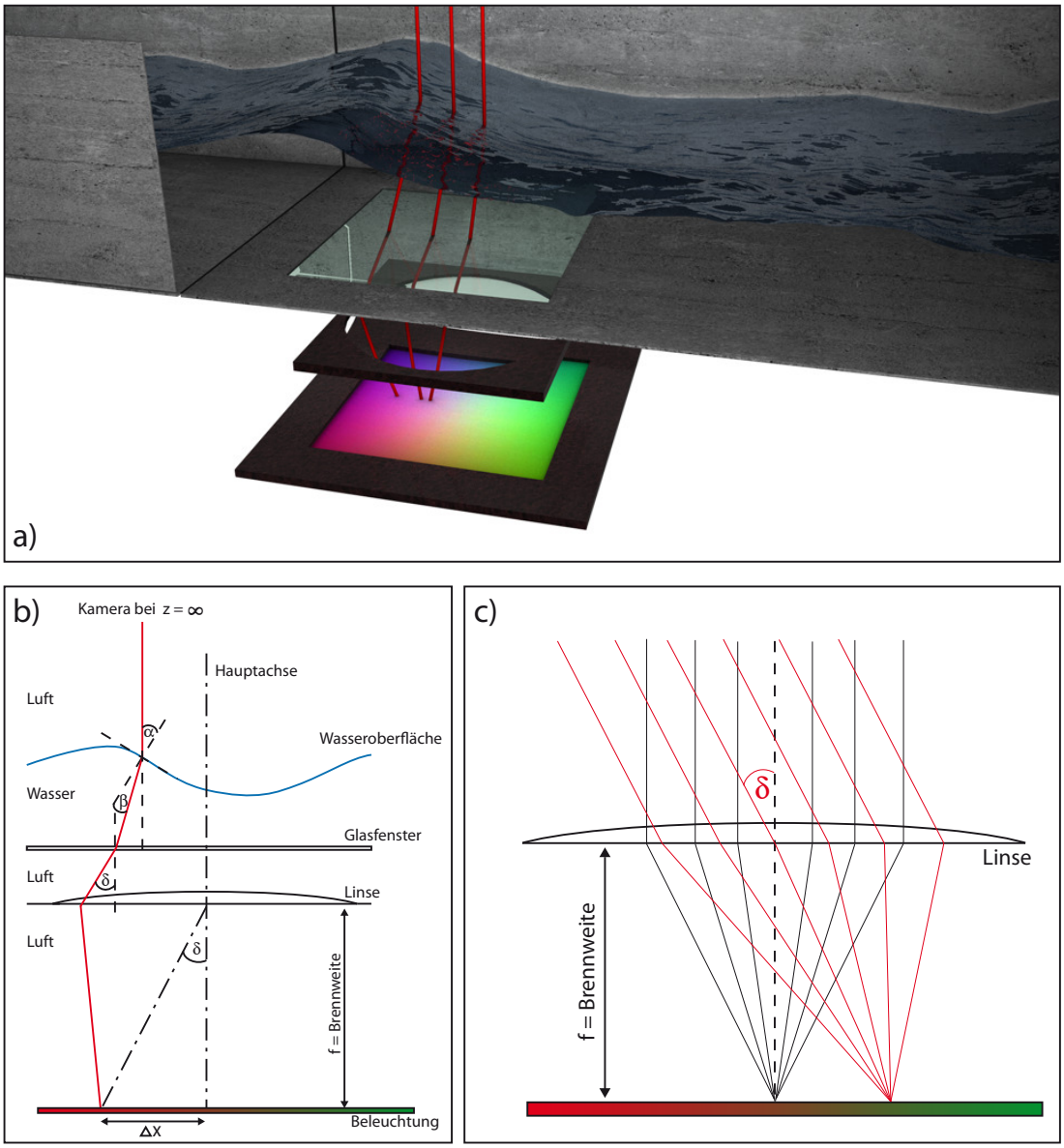

Abbildung 2.20: a) Illustration des CISG-Aufbaus, b) Aufbau schematisch, c) Prinzip telezentrische Beleuchtung (b und c modifiziert nach *Rocholz* [29])

Folgt man in Abbildung 2.20 dem Verlauf eines Lichtstrahls aus der Kamera parallel zur Hauptachse, lässt sich das Prinzip des Aufbaus verstehen. Trifft der Strahl auf die Wasseroberfläche wird er entsprechend deren Neigung gebrochen. Beim Austritt aus dem Fenster des Kanals wird er ein weiteres mal gebrochen, bevor er in die Linse eintritt. Die Fresnel-Linse bildet, da die Lichtquelle sich genau in ihrem Brennpunkt befindet, jeden Strahl unter dem Neigungswinkel δ auf den selben Punkt auf einer Farbmatrix (siehe Abbildung 2.21) ab.

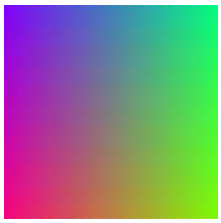

Abbildung 2.21: Farbkodierung

Durch diesen telezentrische Beleuchtung genannten Aufbau wird somit jede Neigung der Wasseroberfläche in beiden Richtungen durch einen eindeutigen Farbwert mit der Position $[X, Y]^T$ auf dem Beleuchtungsschirm codiert. Dies erfolgt über eine von Balschbach [3] eingeführte Methode der Farbkodierung (siehe Abbildung 2.21). Die Kodierung ist eine Superposition dreier linearer Farbkeile in Rot, Grün und Blau.

$$\text{Rot} \propto -\frac{1}{2}X - \frac{1}{2}Y + \frac{1}{2} \tag{2.28}$$

$$\text{Grün} \propto X + \frac{1}{2} \tag{2.29}$$

$$\text{Blau} \propto -\frac{1}{2}X + \frac{1}{2}Y + \frac{1}{2} \tag{2.30}$$

Die Koeffizienten sind hierbei so gewählt worden, dass die Summe der drei Gleichungen konstant und damit unabhängig von der Position auf dem Schirm ist. Daraus lässt sich eine Transformation vom Farbraum zu den Positionen auf dem Schirm ableiten, die über eine Kallibrierung bestimmt wird. Die Steigungen lassen sich dann näherungsweise durch folgende lineare Gleichung bestimmen [29].

$$\begin{bmatrix} s_x \\ s_y \end{bmatrix} \approx \frac{1}{f\,(n_w - 1)} \cdot \begin{bmatrix} X \\ Y \end{bmatrix} \tag{2.31}$$

Hierbei ist f die Brennweite der Linse und n_w der Brechungsindex von Wasser. Aus den beiden durch die Farbbilder gewonnen Steigungskanälen $s_x = \frac{\partial}{\partial x} h\,(x, y)$, $s_y = \frac{\partial}{\partial y} h\,(x, y)$ lassen sich zudem durch Integration die Höheninformation $h\,(x, y)$ der Wasseroberfläche bis auf eine Konstante rekonstruieren. Detailliertere Ableitungen finden sich in [29].
Die Ergebnisse der Neigungsmessung und eine daraus resultierende Höhenrekonstruktion sind anhand eines Beispiels in Abbildung 2.22 dargestellt.

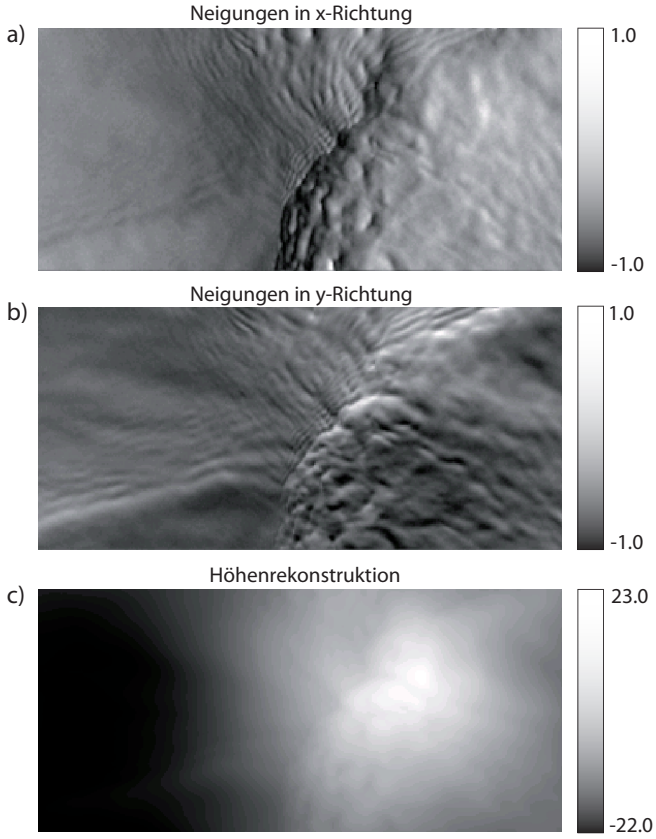

Abbildung 2.22: a) Neigungskanal in x-Richtung, b) Neigungskanal in y-Richtung, c) Höhenrekonstruktion aus a) und b). Die angegebenen Pixelwerte geben die Größenordnungen an und können von Bild zu Bild leicht variieren. In den Darstellungen dieser Arbeit sind alle Grauwertbilder auf Werte zwischen 0 und 255 normiert.

2.5.2 ACFT - Active Thermographie

Da durch Turbulenzen an der Wasseroberfläche ein Austausch der oberflächennahen Schichten stattfindet, lassen sich diese durch den damit einhergehenden Transport von Wärme detektieren. In Feldversuchen wird zur Untersuchung des Wärmetransports im Wasser im Allgemeinen passive Thermographie verwendet (siehe Saunders [31]), da hier eine kühle Oberfläche vorliegt und sich Wasser aus tieferen Regionen durch erhöhte Temperaturen in Infrarotaufnahmen äußern. In Windkanälen ist passive Thermographie allerdings nicht ohne weiteres anwendbar, da es sich im Allgemeinen um geschlossene Systeme mit 100% Luftfeuchtigkeit handelt und deshalb kein latenter Wärmefluss vorhanden ist. Deshalb kommt hier die aktive Thermographie zum Einsatz. Diese besteht aus einem CO_2-Laser, der aktiv die Wasseroberfläche aufheizt und einer Infrarotkamera mit der die Infrarotsignatur beobachtet wird (siehe Abbildung 2.23). Damit können lokale Wärmetransferraten bestimmt werden (Vergleiche Schimpf [32]), was letztendlich Rückschlüsse auf das Vorhandensein oberflächennaher Turbulenzen und damit von *Microscale Breaking* zulässt.

Im Verlauf einer in Kapitel 4.1 beschriebene Messkampagne [29], aus der die in dieser Arbeit verwendeten Daten stammen, wurden verschiedene Varianten der aktiven Thermographie verwendet (siehe Rocholz et al. [30]). Dabei wurde die Art, wie der Laser Wärme auf die Wasseroberfläche bringt, variiert. Der Aufbau bestand aus einem CO_2-Laser, dessen Strahl mittels einer Optik zu einem *Sheet* auffächert und über eine Scan-Einheit gesteuert wurde, einer IR-Kamera, sowie einer Elektronik, die Kamera und Scan-Einheit triggerte. Je nach Modus scannte das Lasersheet (siehe Abbildung 2.23) im vorderen Bereich des CISG-Aufnahmebreichs auf einem schmalen Gebiet, so dass eine dünne Linie entstand, oder es wurde über einen Großteil des Aufnahmebereichs gescannt, um die Wärme möglichst homogen auf einer großen Fläche zu platzieren.

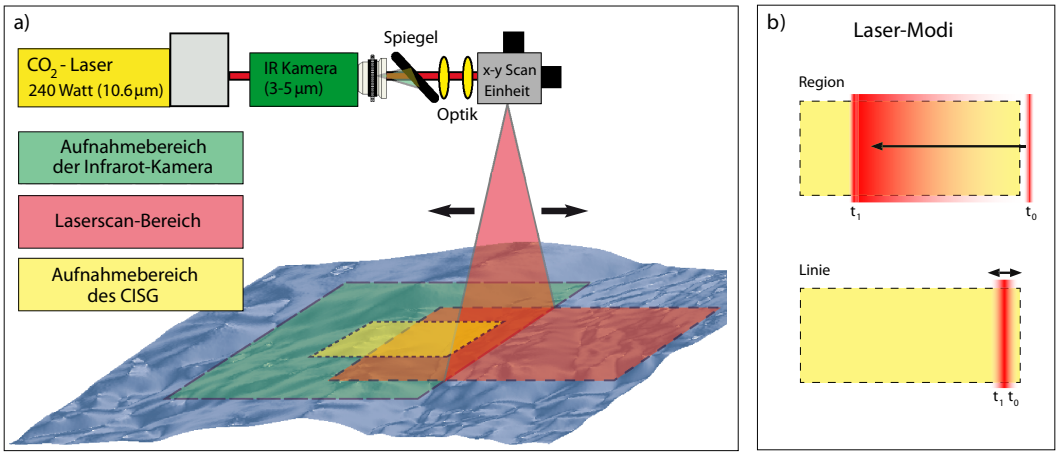

Abbildung 2.23: a) Thermographie Aufbau (Modifiziert nach Rocholz et al. [30]), b) Illustration der Laser-Modi

3 Rendering mit WaveVis

3.1 Einleitung

Das Wellen-Visualisierungs-Tool *WaveVis* wurde entwickelt, um große Datenmengen aus Messkampagnen analysieren zu können. In der 2007 in Hamburg durchgeführten WiSSCy-Kampagne [29] (siehe auch Kapitel 4.1) wurden zeitaufgelöst Neigungen der Wasseroberfläche sowie Thermographiedaten bestimmt. Aus den Neigungen können zusätzlich noch die Höheninformationen rekonstruiert werden (Kapitel 2.5.1). Die Aufzeichnungsfrequenzen betrugen 312.5 Hz und ein Datensatz besteht aus 5000 Bildern, sowie mehreren Kanälen (Steigungen, Höhe, IR-Daten). Aufgrund der Tatsache, dass nur 16 Sekunden Aufnahmezeit 4*5000 Einzelbilder erzeugen wird deutlich das eine Möglichkeit gefunden werden muss die Daten sinnvoll analysieren zu können. Als Lösung hierfür wurde eine Rekonstruktion der Wasseroberfläche mittels Computergrafik und die Möglichkeit von Farb-Overlays zur Darstellung skalarer beziehungsweise 2D-vektorieller Daten gewählt.
WaveVis wurde in seiner ursprünglichen Form von Michael Jung im Rahmen eines Softwarepraktikums[1] geschrieben und im Verlauf dieser Diplomarbeit wesentlich weiterentwickelt. Dabei wurden neben einigen Verbesserungen bezüglich der Benutzerfreundlichkeit, eine einfache Visualisierung von 3D Skalar- und Vektordaten im Wasservolumen sowie ein Modul zur Markierung bestimmter Ereignisse direkt in der Visualisierung zur späteren Verwendung in der Software Ilastik (Siehe Kapitel 2.3) implementiert.

3.2 Beschreibung der Klassen und Routinen

Die Software ist in C++ und OpenGL® [34, 35] sowie Glut [24] verfasst. Der prinzipielle Aufbau geht aus Abbildung 3.1 hervor. Die zentrale Klasse *wave* ist für die Darstellung der Visualisierung und das Erzeugen aller weiteren verwendeten Objekte zuständig. Ihr wird beim Programmeinstieg ein Objekt der Klasse *waveConfig* übergeben, das über eine *parser-function* alle in der Datei *ConfigFile.txt* enthaltenen Parameter aufnimmt. Diese Konfigurationsdateien beinhalten alle nötigen Angaben wie Auflösung, Anzahl der Frames, Pfade zu den nötigen Daten und sonstige Einstellungsoptionen. Näheres zu den Konfigurationsdateien findet sich in Abschnitt 3.3.

[1] http://pille.iwr.uni-heidelberg.de/~wellen01/

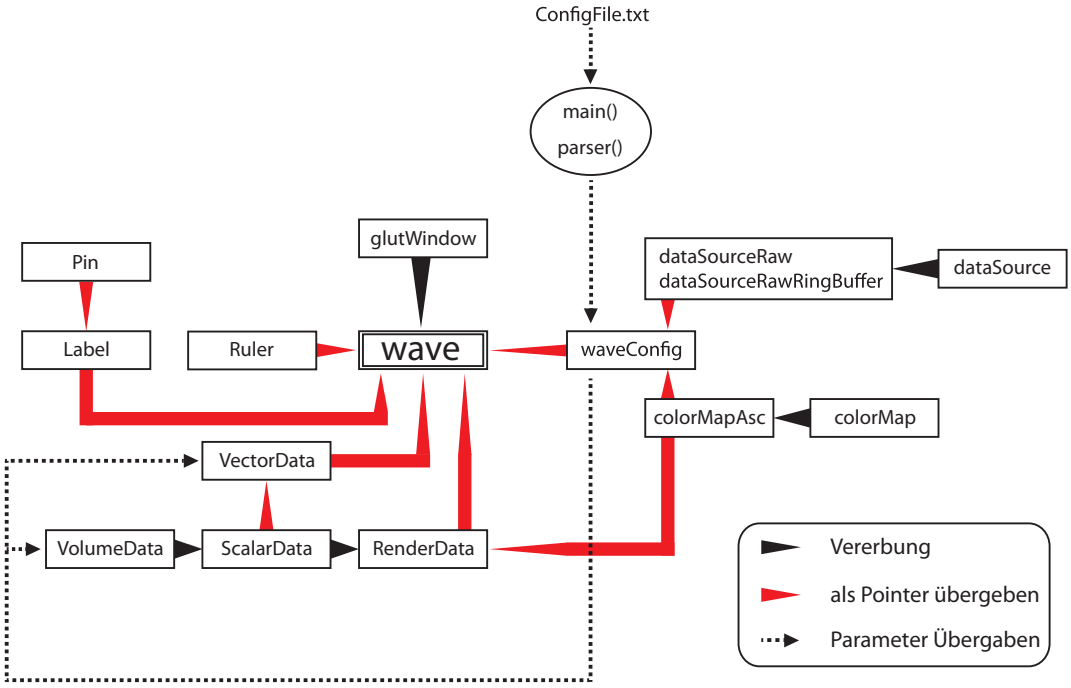

Abbildung 3.1: Klassenstruktur von *WaveVis*

glutWindow: Die Klasse *glutWindow* ist die Elternklasse von *wave* und sorgt für die korrekte Verteilung der Glut-Callbacks auf die jeweiligen Memberfunktionen, indem sie einen Zeiger auf die zugehörigen Dispatcher-Funktion übergibt. Damit werden alle vom Nutzer gegebenen Signale korrekt verarbeitet und die Render-Pipeline aktualisiert.

wave: Die Klasse *wave* ist die zentrale Klasse, welche die graphische Darstellung generiert, die nötigen Objekte erzeugt und in der alle Benutzer-Interaktionen verarbeitet werden. Im Quellcode ist sie auf vier Dateien aufgeteilt. Diese sind die Headerdatei *wave.h*, sowie die Dateien *waveother.cpp*, *waveinteraction.cpp* und *wavedisplay.cpp*. Letztere ist dabei für die 3D-Darstellung zuständig, während in *waveother.cpp* Konstruktor und Destruktor sowie die Steuerung der Framewechsel und wichtige Funktionen zum Overlay-Handling stehen. In *waveinteraction.cpp* findet sich die Behandlung aller Benutzereingaben durch Maus und Tastatur. Beim Programmstart muss einem Objekt der Klasse *wave* ein Zeiger auf ein Objekt der Klasse *waveConfig* übergeben werden, da dieses alle nötigen Parameter für das Rendern enthält. Zeiger auf die anderen in Abbildung 3.1 dargestellten Objekte werden nur bei Bedarf oder bei Aufruf zur Laufzeit erzeugt.

waveConfig: Wird bei Programmstart kein Objekt dieser Klasse an ein Objekt der Klasse *wave* übergeben, so wird das Programm sofort beendet, da in *waveConfig*

alle nötigen Parameter aus der Konfigurationsdatei gespeichert werden. Die Konfigurationsdatei muss der ausführbaren Datei *WaveVis.exe* als Parameter übergeben werden, woraufhin die Parser-Funktion alle nötigen Daten ausliest und in der Klasse *waveConfig* ablegt. Sie beinhaltet eine Vielzahl von get- und set-Funktionen sowie einen „function-pointer“, der von Glut aufgerufen werden kann wenn Resourcen zur Verfügung stehen um zum Beispiel Daten zu streamen.

dataSource: Diese Klasse ist die Basisklasse von *dataSourceRaw* und *dataSourceRawRingBuffer*. Beide sind für das Laden der darzustellenden Daten zuständig. In ihr sind einige virtuelle Funktionen angelegt, die in den abgeleiteten Klassen implementiert werden müssen. So zum Beispiel Funktionen wie *nextFrame(),lastFrame()* usw., die immer dann aufgerufen werden, wenn Daten dargestellt werden sollen und diese in das von OpenGL®zum Rendern verwendete Array geschrieben werden müssen. Außerdem nimmt ein Objekt der Klasse die zur Darstellung von Overlays nötigen Parameter auf und stellt Funktionen zum Zugriff auf diese bereit.

dataSourceRaw: Objekte dieser Klassen laden die Daten vollständig in den Speicher. Die Erzeugung wird über einen Parameter *BUFFERED* (siehe Abbildung 3.2) in der Konfigurationsdatei gesteuert. Bei Speicherproblemen sollte dieser immer zuerst überprüft werden.

dataSourceRawRingBuffer: Erzeugt man ein Objekt dieser Klasse, werden die Daten in einen Buffer geladen. Die Größe des Buffers kann wieder über die Konfigurationsdatei festgelegt werden. Durch das Buffern der Daten können auch Datensätze größer als der zur Verfügung stehende Arbeitsspeicher visualisiert werden.

colorMap: Dies ist die Basisklasse der Colormaps. Auch sie enthält virtuelle Funktionen, die in abgeleiteten Klassen implementiert werden müssen. Darunter die zentrale Methode *lookUp()*, durch die vom Objekt der Klasse *wave* aus die korrekten Werte für die Falschfarbendarstellung aus den Colormaps ermittelt werden.

colorMapAsc: Diese von *colorMap* abgeleitete Klasse liest Lookup-Tabellen aus Ascii-Dateien aus und stellt sie in der Visualisierung bereit. In ihr sind zwei Varianten der Funktion *lookUp* implementiert. Eine skaliert beliebige Werte auf einen Bereich zwischen 0 und 1, die andere dient zum Beispiel der Darstellung von Wahrscheinlichkeiten und lässt die Daten unskaliert. Der Benutzer muss dann aber dafür sorgen, dass nur Werte zwischen 0 und 1 an die Funktion übergeben werden. Die Auswahl geschieht wieder über die Konfigurationsdatei. Näheres hierzu findet sich in Abschnitt 3.4.2.

Ruler: Die Klasse *Ruler* stellt ein Objekt zur Darstellung von Linealen in der Wellenvisualisierung bereit. Es können vom Benutzer selbst Bilder in Form von .gif Dateien erzeugt und geladen werden. Es können Bilder in x und y Richtung

angelegt werden, die die doppelte Auflösung der Daten in der jeweiligen Richtung haben sollten. Siehe auch Abschnitt 3.4.3

Pin und Label: Diese beiden Klassen stellen die Funktionalität des Marker-Moduls bereit. Im Objekt der Klasse *Label* können beliebig viele Objekte der Klasse *Pin* erzeugt werden, die in Form von Zylindern mit kreisrunder beziehungsweise mit elliptischer Grundfläche in jedem Frame der Visualisierung plaziert werden können. Die genauen Orte und Halbachsen, sowie die durch Farben gekennzeichneten Klassenbezeichnungen, werden in *Label* gespeichert und können in einer Textdatei ausgegeben werden. Details siehe Abschnitt 3.4.4.

VolumeData: Die Basisklasse von *ScalarData*. Sie stellt die notwendigen Methoden zum Laden der Volumendaten bereit. Das Modul zur Visualisierung von Daten im Wasserkörper ist im Laufe dieser Arbeit über ein Teststadium leider nicht hinaus gekommen, so dass noch keine allgemeine Einbettung in den Rest des Programmes erfolgt ist und bisher nur Daten in spezieller Form und ungepuffert geladen werden können.

ScalarData: *ScalarData* erbt von *VolumeData*. Hier werden Funktionen bereitgestellt, die aus den Volumendaten die aktiven Projektionsebenen im Volumen lesen. Es können immer nur einzelne Ebenen entweder im xz-Schnitt oder im yz-Schnitt projiziert werden (Siehe 3.4.5), weshalb über Pointerarithmetik immer die korrekten Werte in die darzustellenden Arrays geladen werden müssen.

RenderData: *RenderData* erbt wiederum von *ScalarData* und erzeugt das letztendlich im Objekt *wave* dargestellte Mesh senkrecht zur Wasseroberfläche. Dabei werden die z-Koordinaten der obersten Reihe des darzustellenden Gitters aus den Höhendaten der Wellenoberfläche bezogen und es wird jedem Punkt der gesamten Ebene entsprechend dem Wert der Daten ein Farbwert aus dem *colorMap* Objekt zugewiesen.

VectorData: Die Klasse *VectorData* erzeugt in ihrem Konstruktor drei Objekte der Klasse *ScalarData* und lädt in diesen die Werte der Vektorkomponenten. Eine Methode *drawVector()* übernimmt dann die Darstellung aller Vektoren in einer Ebene der Visualisierung aus den drei *ScalarData*-Objekten.

3.3 Konfigurationsdateien

WaveVis wird mit Konfigurationsdateien angesteuert. Dies sind einfache Text-Dateien, in denen alle für die Visualisierung nötigen Informationen übergeben werden.

3.3.1 Parameter Settings

Im ersten Teil der Konfigurationsdatei werden alle nötigen Parameter gesetzt:

```
################################################################################
############################### parameter settings #############################
################################################################################

#### resolution of data ####
RES_X   320
RES_Y   149

#### number of frames ####
NUM_OF_FRAMES   5000
FRAMES_PER_SEC  30

#### set the initial angles of the camera here ####
CAMERA_PHI      132
CAMERA_THETA    34
CAMERA_DISTANCE 1.4

#### set dimensions ####
X_DIMENSION     22,3cm
Y_DIMENSION     10,4cm

#### set textsize of dimensions, posible values are 12, 18, 24 ####
TEXT_SIZE       18

#### set light position ####
X_LIGHT_POSITION        0.0
Y_LIGHT_POSITION        0.0
Z_LIGHT_POSITION        50.0

#### set this to 1 if the data should be loaded into RAM dynamically
#### if activated be sure to set the max / min values for scalar overlay fields
BUFFERED        1

#### size of the buffer in frames ####
BUFFER_SIZE     100

#### scaling in z direction ####
SCALING 1.43

#### z value for the water box's ground ####
GROUND_Z        -35

#### set wind arrow position ####
#### Angle [0,360] - distance & height measured in pixel ####
WINDARROW_ANGLE 90
WINDARROW_DISTANCE      200
WINDARROW_HEIGHT        30

#### set to 1 if animation shall be enabled at startup, 0 otherwise ####
ANIMATION       0

#### color of the background in numbers between 0.0 and 1.0 ####
BG_RED  1.0
BG_GREEN        1.0
BG_BLUE 1.0

#### user text color ####
TXT_RED 0.0
TXT_GREEN       0.0
TXT_BLUE        0.0

#### water surface color
SURFACE_COLOR_R 0.5
SURFACE_COLOR_G 0.5
SURFACE_COLOR_B 0.9

#### user text size ####
#### this number have to be the number of textlines above ####
TXT_SIZE        3

#### user text lines ####
LINE_1      Hamburg 2007 Run044
LINE_2  Windgeschwindigkeit 6.0 m/s
LINE_3
```

Abbildung 3.2: Konfigurationsdatei - Parameter Settings

- Auflösung der Daten in x- und y-Richtung
- Anzahl der zu verwendenden Frames
- Abspielgeschwindigkeit in Frames pro Sekunde
- Anfangsposition der Kamera
- Anzuzeigende Größe der Visualisierung
- Textgröße
- Anfangsposition der Lichtquelle
- Puffer Modus ein/aus
- Größe des Puffers
- Skalierung der z-Koordinate
- Tiefe des Wasserbeckens
- Position des Windpfeiles
- Entscheidung, ob die Animation bei Start direkt abgespielt werden soll
- Hintergrundfarbe
- Textfarbe
- Farbe der Wasseroberfläche
- Anzahl der Überschrift-Textzeilen
- Überschrift-Text

3.3.2 Render Options

In den Render Optionen wird festgelegt, ob die Screenshot Funktion aktiviert sein soll und wo die gemachten Bilder gespeichert werden. Außerdem können die zu verwendenden Lineale gesetzt werden. Näheres zu beiden Möglichkeiten findet sich in 3.4.3.

```
##########################################################################################
################################ render options ##########################################
##########################################################################################

#### set RENDER_SCENE 1 enable/0 disable screenshots  ####
RENDER_SCENE    1

#### in REC_IMAGE_PATH screenshots are stored with name image_ + frame number in .png ####
REC_IMAGE_PATH  C:\WaveVis\images\image_

#### load ruler ####
RULER_X C:\WaveVis\resources\sideRuler.gif
RULER_Y C:\WaveVis\resources\frontRuler.gif
```

Abbildung 3.3: Konfigurationsdatei - Render Options

3.3.3 Data Files

Hier werden zuerst die zur Wellendarstellung zwingend notwendigen Daten der Höheninformation und der Steigungen in X- und Y-Richtung angegeben.

```
################################################################################
############################### data files #####################################
################################################################################

#### surface data files ####
FILE_HEIGHT     C:\WaveVis\Data\Run044_Height.raw
FILE_SLOPE_X    C:\WaveVis\Data\Run044_Sx.raw
FILE_SLOPE_Y    C:\WaveVis\Data\Run044_Sy.raw


################ overlays ####################

#### the begin and end lines also have to end with a tab ####
#### set OVERLAY_TYPE to 0 if the overlay is a probability ####
OVERLAY_BEGIN
OVERLAY_TYPE    1
OVERLAY_FILE0   C:\WaveVis\Data\Run044_IR.raw
OVERLAY_COLORMAP        C:\WaveVis\resources\Colormaps\jet_colormap.txt

OVERLAY_SIGMA_MIN       1.0
OVERLAY_SIGMA_MAX       1.0

COLORBAR_UNIT   C
OVERLAY_END


############## vector overlays ##################

#### the begin and end lines also have to end with a tab ####
OVERLAY_BEGIN
OVERLAY_TYPE    2
OVERLAY_FILE1   C:\WaveVis\Data\sinus.raw
OVERLAY_FILE2   C:\WaveVis\Data\sinus.raw

#### the following value is needed if buffering is used ####
OVERLAY_VALUE_MAX       5.0
OVERLAY_END


#### you can add other overlays here ####
```

Abbildung 3.4: Konfigurationsdatei - Data Files

Die Daten müssen dabei im Raw-Format als einfacher binärer float stream vorliegen. Hierin werden die Daten in X-Richtung hintereinander abgelegt, so dass ein Stream von (Auflösung X)∗(Auflösung Y)∗Frames entsteht. Im Anschluss können optional Overlays festgelegt werden. Der Parameter $OVERLAY\_TYPE$ definiert dabei die Art des Overlays. Setzt man den Wert auf 0, müssen die Daten im Intervall [0,1] vorliegen und werden absolut angezeigt. Bei einem Parameterwert von 1 werden Mittelwert $\langle x \rangle$ und Standardabweichung σ über die Daten im Puffer berechnet und der Darstellungsbereich auf:

$$max = \langle x \rangle + OVERLAY\_SIGMA\_MAX * \sigma \tag{3.1}$$

$$min = \langle x \rangle - OVERLAY\_SIGMA\_MIN * \sigma \tag{3.2}$$

angepasst. Für Vektor-Overlay's wird $OVERLAY\_TYPE$ 2 gewählt und zwei Datensätze mit 2D-Vektorkomponenten als $OVERLAY\_FILE1/2$ angegeben. Mit dem Parameter $COLORBAR\_UNIT$ wird die Beschriftung der Colorbar festgelegt. Nach dem Block des Vektor-Overlays können auch noch weitere Overlays nach selbem Schema geladen werden.

3.3.4 3D Daten

Im Abschnitt *3D Data* der Konfigurationsdateien wird zuerst die Anzahl der zu verwendenden Volumendatensätze über *NUMBER_OF_LAYERS* festgelegt. Hier sollten die Datensätze analog zu den übrigen in einem binären Raw Format vorliegen, wobei jede Schicht, angefangen bei der untersten am Ende des Dateinamens durchnummeriert sein sollte. Die Volumendaten müssen also in Volumenschichten parallel zur Wasseroberfläche aufgeteilt werden. Jede Volumenschicht enthält dabei die Information in x- und y-Richtung und der Zeit abgelegt als Float-Kolonne, wobei die Daten in x-Richtung aneinander gehängt werden. Die Koordinate jedes Datenpunktes in z-Richtung wird dabei von den Daten vorgegebenen Transformationen bestimmt.

$$x = \frac{6.283185}{128} \cdot (I-1) \quad , \quad y = \frac{6.283185}{128} \cdot (J-1) \tag{3.3}$$

$$z = \left[\frac{tanh\left(1.8417 \cdot \frac{K-1}{128}\right)}{tanh\left(1.8417\right)} \right] \cdot \left[\eta\left(I,J\right) + 5.026548\right] - 5.026548 \tag{3.4}$$

Hierbei ist $\eta\left(I,J\right)$ die Höhe am Pixel (I,J). K ist der Index der Volumenschicht. Eine allgemeine Behandlung von Koordinatensystemen ist bisher noch nicht implementiert worden (siehe Kapitel 5). Die jeweiligen *ENABLED*-Parameter legen dabei über den Boolean-Wert fest, ob die Daten geladen werden sollen. In *SCALAR_DATA_PATH* wird der Ort und der Dateiname ohne die Numerierung festgelegt. Zudem muss bei skalaren Volumendaten ebenfalls eine Colormap definiert werden. Im Abschnitt der Vektordaten kann eine maximale Länge der Vektorpfeile festgelegt werden und analog zur Vorgehensweise bei Skalaren Daten werden drei Pfade mit Dateinamen benötigt, in denen die x-,y- und z-Komponenten zu finden sind.

```
################################################################################
############################### 3D data  #######################################
################################################################################

NUMBER_OF_LAYERS        40

#### scalar data ####
SCALARDATA_ENABLED      0
SCALAR_DATA_PATH
SCALAR_COLORMAP C:\WaveVis\resources\Colormaps\jet_colormap.txt

#### vector data ####
VECTORDATA_ENABLED      0
MAX_VECTOR_LENGTH       0.6
VECTOR_DATA_PATH_U
VECTOR_DATA_PATH_V
VECTOR_DATA_PATH_W
```

Abbildung 3.5: Konfigurationsdatei - 3D Data

3.4 Anwendung

Das Wellen-Visualisierungs-Tool *WaveVis* wurde Entwickelt, um mit Hilfe der CISG (siehe Abschnitt 2.5.1) gewonnene Daten analysieren zu können. Eine Rekonstruktion der Vorgänge auf der Wasseroberfläche dreidimensional aus jeder Perspektive und mit beliebiger Wiedergabegeschwindigkeit betrachten zu können eröffnet die Möglichkeit, interessante Ereignisse einfacher in den Daten ausfindig zu machen und teilweise überhaupt erst zu erkennen, dass es sich um solche handelt. Denn Vorgänge, die sich in der Zeit entwickeln, lassen sich nur schwer in unbewegten Einzelbildern erfassen. Ist die Information noch dazu auf mehrere Kanäle verteilt, wird dies nahezu unmöglich. *WaveVis* leistet in diesem Fall Abhilfe, da sich damit vier Kanäle, in für den Menschen optimaler Weise, gleichzeitig betrachten lassen.

3.4.1 Wellen-Visualisierung

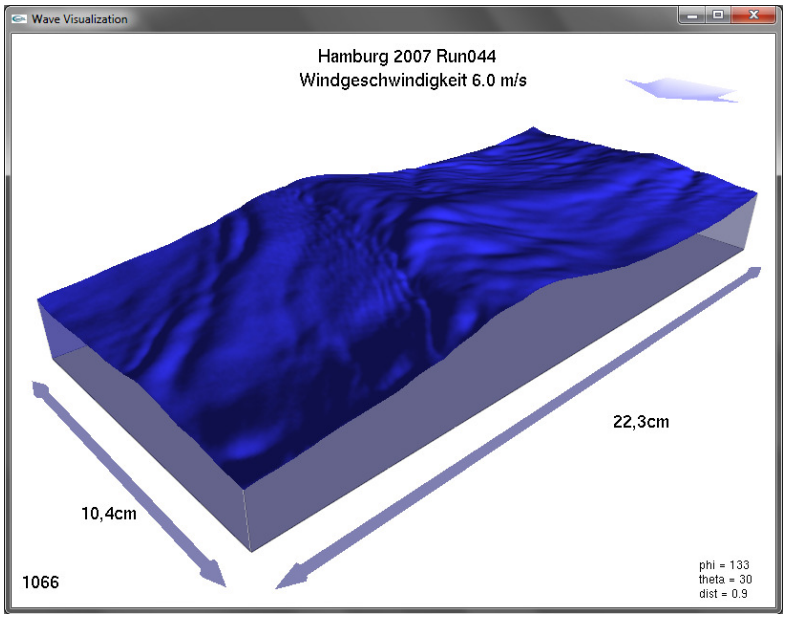

Abbildung 3.6: WaveVis

In Abbildung 3.6 ist mit *WaveVis* ein Datensatz aus der WiSSCy-Messkampagne geladen worden. Man erkennt die Darstellung der Welle sowie die Frame-Nummer (unten links), die Angaben zur tatsächlichen Größe des Wasserauschnitts, die Windrichtung und die Informationen zur aktuellen Kameraposition (unten rechts). Das Wasservolumen lässt sich beliebig mit Hilfe der Maus im Raum drehen, verschieben und verkleinern beziehungsweise vergrößern. Informationen über die Steuerung der Visualisierung erhält man sowohl über eine schnelle Bildschirmhilfe mittels F1, als auch über einen integrierten ausführlicheren Hilfe-Browser wie in Abbildung 3.7 zu sehen.

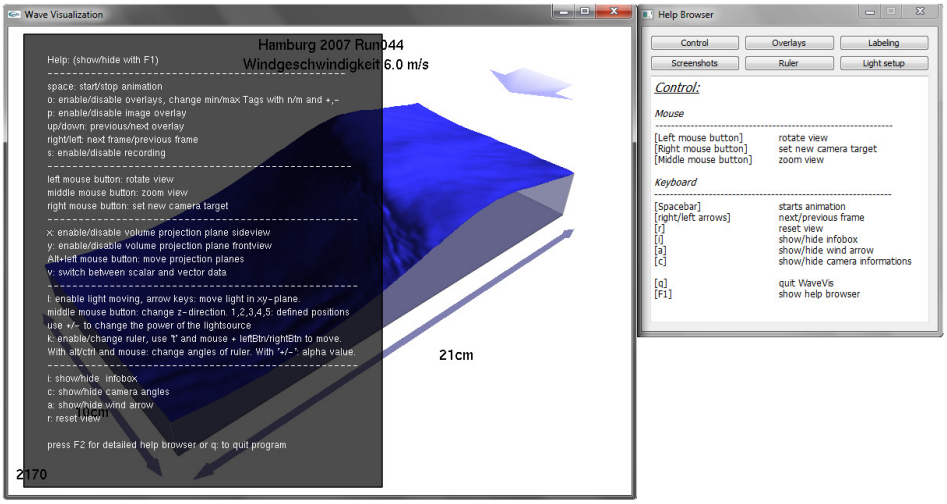

Abbildung 3.7: WaveVis Hilfe Optionen

3.4.2 Overlays

In *WaveVis* gibt es drei verschiedene Arten Falschfarben-Overlays darzustellen. Prinzipiell muss aber nur zwischen skalaren und vektoriellen Overlays unterschieden werden. Bei skalaren Daten gibt es den Modus 0, in welchem die Daten unskaliert auf das Intervall zwischen $[0, 1]$ in den durch die geladene Colormap definierten Falschfarben dargestellt werden. Die Darstellung eignet sich zum Beispiel für Wahrscheinlichkeitsdatensätze. Im Modus 1 werden ebenfalls skalaren Daten Farbwerte auf der Wasseroberfläche zugewiesen, allerdings wird hierbei aus den geladenen Werten im Speicher Mittelwert und Standardabweichung berechnet und das Darstellungsintervall nach den Gleichungen 3.1/ 3.2 angepasst. Die vom Benutzer anfänglich in der Konfigurationsdatei festgelegten Faktoren OVERLAY_SIGMA_MIN und OVERLAY_SIGMA_MAX zur Definition des Darstellungsbereichs um den Mittelwert können in der laufenden Visualisierung jederzeit geändert werden.

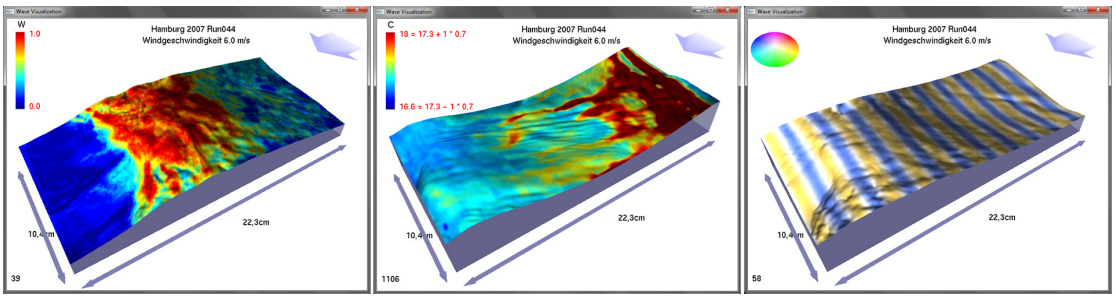

Abbildung 3.8: WaveVis Overlay Möglichkeiten, von links nach rechts: Modus 0 (absolut) zum Beispiel Wahrscheinlichkeiten, Modus 1 (Mittelwert) Temperatur, Modus 2 (Vektoriell) für Strömungen

Die dritte Art des Daten-Mapping ist der Modus 2, welcher es erlaubt, zwei Datensätze jeweils mit x- und y-Komponenten zu laden und so 2D-Vektorfelder auf die Wasseroberfläche zu projizieren. Hierbei werden Richtung und Länge der Vektoren über den HSI-Farbraum visualisiert. Die Richtung wird durch die Farbe und die Länge mit Hilfe der Sättigung codiert. In Abbildung 3.8 ist rechts beispielhaft ein Vektordatensatz dargestellt, dessen x- und y-Komponenten ein sinusförmiges Verhalten aufweisen. Im Vektoroverlay-Modus können zusätzlich *Streaklines* gesetzt werden. Diese entsprechen beispielsweise den Bahnen von Farbpartikeln, die an dieser Stelle in das Strömungsfeld gesetzt würden. *WaveVis* kann immer nur eine Overlay-Art darstellen, es können aber mehrere geladen und zur Laufzeit gewechselt werden.

3.4.3 Funktionen von WaveVis

Folgende Funktionen zur besseren Analyse, Darstellung und Benutzerfreundlichkeit wurden implementiert:

Lineale

Es besteht die Möglichkeit, Bilder senkrecht zur Wasseroberfläche in der Visualisierung einzublenden. Dazu werden für jede Seite des Wasservolumens .gif Dateien mit der doppelten Auflösung der Daten mittels der Konfigurationsdatei geladen und können dann bei Bedarf verwendet werden. Vorrangig dient diese Funktion dem Einblenden von Linealen, die zur Analyse verwendet werden können.

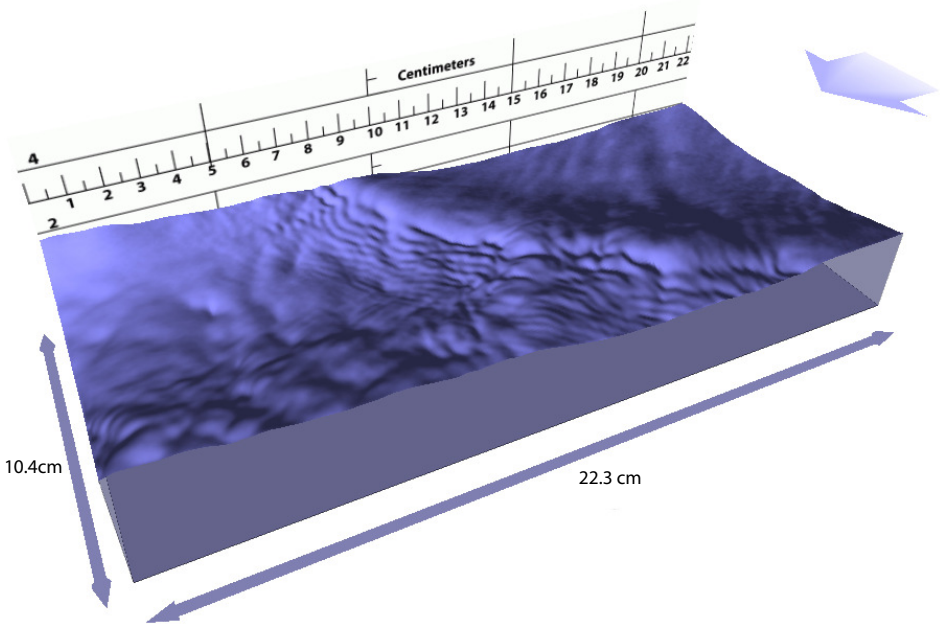

Abbildung 3.9: WaveVis Lineale

Lichtverhältnisse

Durch die hohe Dynamik der Amplituden in den Wellendaten ist eine Darstellung der kleinskaligen Strukturen wie den Kapillarwellen über reine Meshmodellierung nicht zu erreichen. Daher ist in der Wellenvisualisierung eine Darstellung durch Abschattung, das sogenannte *Shading* nötig. Dabei werden die gemessenen Steigungen zur Berechnung der Schattierungen der Oberfläche, je nach Standort der Lichtquelle, verwendet. Die Ausleuchtung der Visualisierung lässt sich vom Benutzer frei verändern. Sowohl die Position als auch die Intensität der Lichtquelle kann zur besseren Darstellung verändert werden.

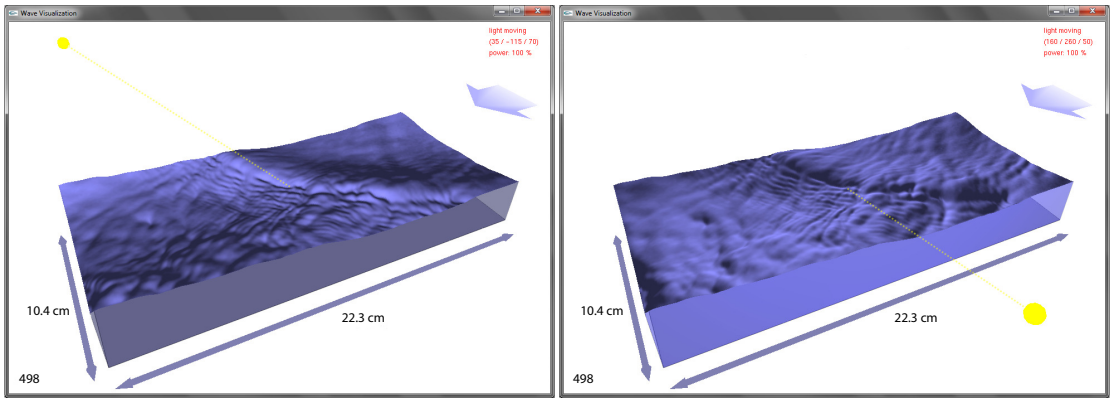

Abbildung 3.10: WaveVis Lichtverhältnisse

Screenshots

Auf Basis der *Corona*-Bibliothek wurde ein Screenshot-Mechanismus in *WaveVis* integriert. Der Zweck ist das einfache Erzeugen von Videos einer Datenvisualisierung. Daher kann in der Konfigurationsdatei über *RENDER_SCENE* die Aufnahmefunktionalität aktiviert und über *REC_IMAGE_PATH* ein Speicherort festgelegt werden. Dabei wird der Pfad des Ordners und der gewünschte Dateiname angegeben. Die Nummerierung erfolgt automatisch und die Ausgabe geschieht im Png-Format. Danach kann in *WaveVis* jederzeit durch betätigen der Aufnahmetaste ein Screenshot der momentanen Darstellung gemacht werden. Bleibt die Aufnahme aktiviert, wird bei jeder Änderung der Szene ein weiterer Screenshot gemacht und gespeichert, so dass jede Veränderung des Wasservolumens durch den Benutzer sowie jeder Zeitschritt gespeichert wird. Mit OpenSource-Software wie zum Beispiel *VirtualDub* kann aus den Bildsequenzen sehr leicht ein Video erzeugt werden.

3.4.4 Labeln mit WaveVis

In *WaveVis* wurde die Möglichkeit geschaffen, Ereignisse direkt in der Visualisierung zu labeln. Dies geschah im Hinblick auf die spätere Nutzung der Daten in der interaktiven Segmentierungs-Software *Ilastik* (siehe hierzu 2.3). Da Ereignisse wie *Microbreaking* in der Visualisierung sehr viel besser zu erkennen sind, ist es sinnvoll, zum Training zu verwendende Beispiele direkt in der 3D-Umgebung zu markieren. Hierfür können im *Label-Modus* Zylinder beziehungsweise Zylinder mit elliptischer Grundfläche gesetzt und einer bestimmten Klasse durch Wahl der Farbe zugewiesen werden.

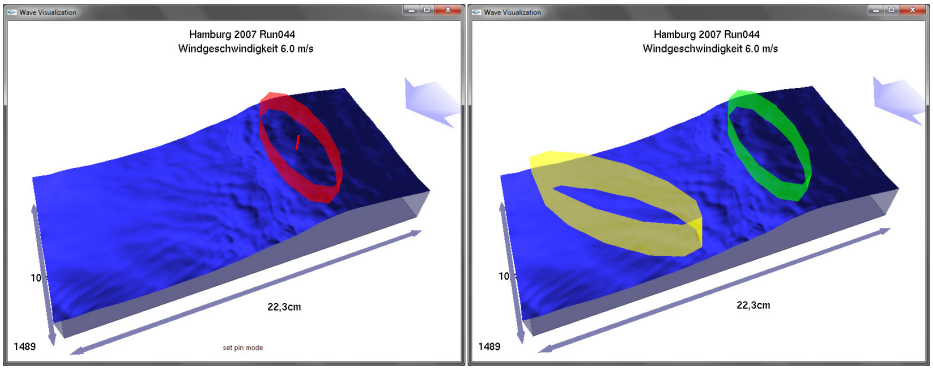

Abbildung 3.11: WaveVis Label-Modus

WaveVis legt die nötigen Informationen wie Framenummer, Position, Größe der Halbachsen und Drehwinkel auf Wunsch in einer Datei ab, die im Anschluss mit einer für diesen Zweck entwickelten Anwendung zur Konvertierung der Daten geöffnet werden kann. Mithilfe des *LabelViewer* lassen sich die gesetzten Label's betrachten und zusammen mit den Daten direkt in die von *Ilastik* benötigten *HDF5*-Formate konvertieren. Im Folgenden fett gedrucktes bezieht sich immer auf einen Button aus Abbildung 3.12

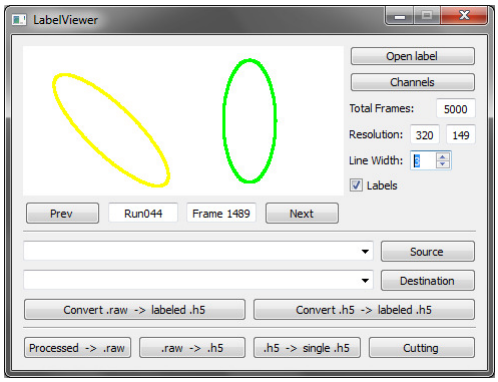

Abbildung 3.12: WaveVis LabelViewer

Vor der Konvertierung mit ***Convert .raw -> labeled .h5*** können Strichstärke und Kanäle gewählt werden. Es empfiehlt sich aus Geschwindigkeitsgründen zuerst die *Raw*-Formate mit der gewünschten Anzahl Kanäle mittels ***.raw -> .h5*** in eine *HDF5*-Datei zu schreiben, die im Anschluss in einzelne Frames über ***.h5 -> single .h5*** zerlegt werden kann. Die Daten in einzelne Frames zu zerlegen hat sich an vielen Stellen als nützlich erwiesen. Nun kann der *LabelViewer* sehr schnell die einzelnen markierten Frames über ***Convert .h5 -> labeled .h5*** in das von *Ilastik* benötigte Format bringen. Zusätzlich gibt es die Möglichkeit, mit ***Processed -> .raw*** die von *Ilastik* berechneten Dateien zurück in das für *WaveVis* lesbare *Raw*-Format zu bringen oder mittels ***Cutting*** beliebige Stücke aus den *Raw*-Dateien zu schneiden.

3.4.5 Volumendaten

Anhand simulierter Daten (nach Wu-Ting Tsai, Li-Ping Hung [38]) wurde *WaveVis* um die Funktionalität der Darstellung von Volumendaten erweitert. Die Daten stellen die Höheninformation der Wasseroberfläche, zwei Sätze skalarer Volumendaten für Temperatur und Druck und drei Datensätze für die Vektorkomponenten der Strömungsverhältnisse im kompletten Raum bereit. Auf deren Grundlage wurde die Möglichkeit geschaffen, skalare (Abbildung 3.13) und vektorielle (Abbildung 3.14) Daten im Wasserkörper zu visualisieren. Die Darstellung erfolgt dabei in einer vom Nutzer in beide Richtungen frei verschiebbaren Ebene senkrecht zur Wasseroberfläche. Skalare Daten werden an jedem Voxel über den zugehörigen Farbwert einer Colormap dargestellt. Vektorfelder werden über in der Länge variierende Vektorpfeile visualisiert. Sowohl die Wahl der *Colormap* im Falle skalarer Daten, als auch die maximale Länge der Vektorpfeile sind vom Nutzer festlegbar. Es können beide Arten von Daten gleichzeitig geladen und die Darstellungen zur Laufzeit gewechselt werden.

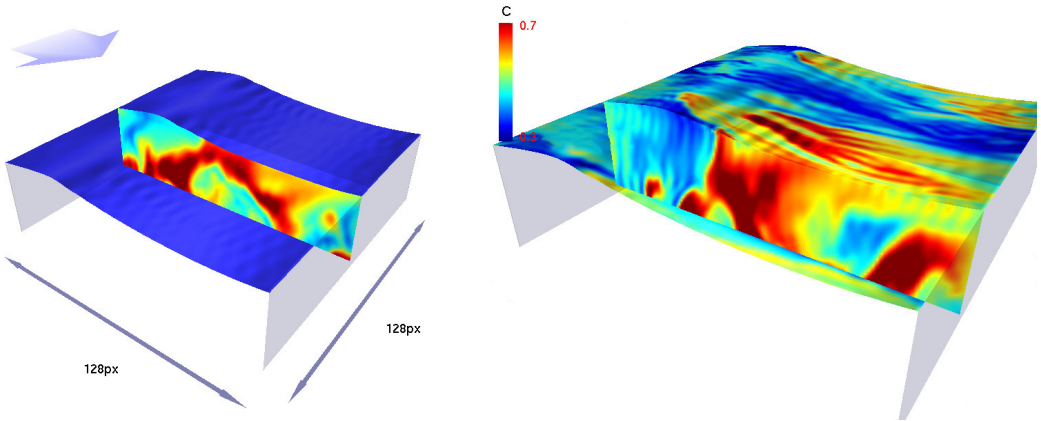

Abbildung 3.13: Visualisierung skalarer Volumendaten in WaveVis

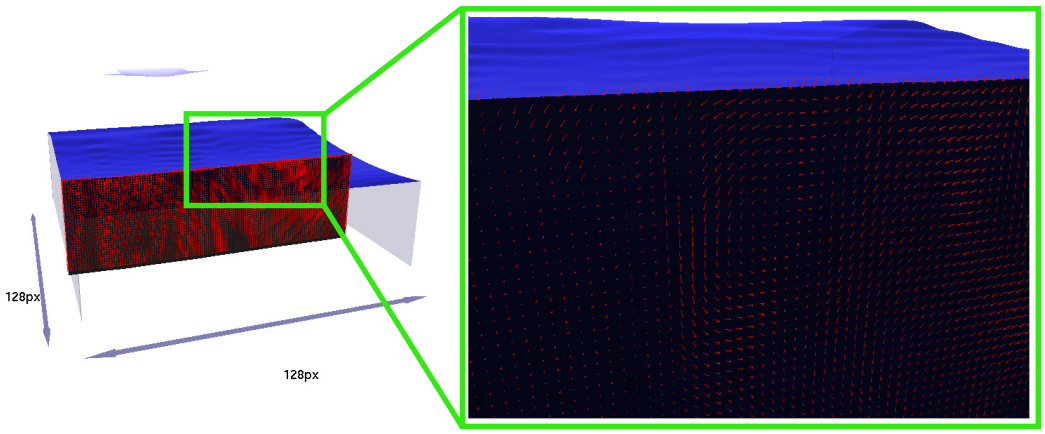

Abbildung 3.14: Visualisierung vektorielle Volumendaten in WaveVis

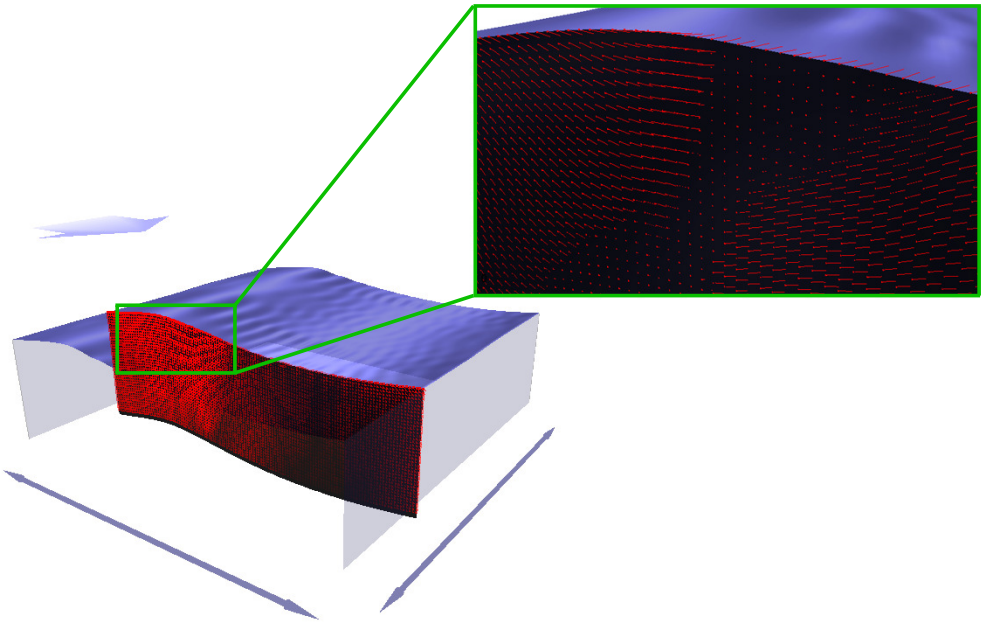

Abbildung 3.15: Visualisierung vektorielle Volumendaten in WaveVis

4 Interaktive Segmentierung

4.1 WiSSCy-Messkampagne

Die in dieser Arbeit verwendeten Daten stammen alle aus einer 2007 in Hamburg durchgeführten Messkampagne [29]. Diese trägt den Namen *Impact of Wind, Rain and Surface Slicks on Air-Sea CO_2 Transfer Velocity - Tank Experiments*, auch *WiSSCy-Project* genannt. Das Projekt war eine Zusammenarbeit der Forschungsgruppen um Prof. Dr. Detlef Stammer, Universität Hamburg, und Prof. Dr. Bernd Jähne, Universität Heidelberg. Zielsetzung war die Untersuchung des Gasaustausch zwischen Atmosphäre und Ozean unter den Einflüssen von Wind, Regen und Oberflächenfilmen. Durchgeführt wurden die Experimente in einem linearen Windkanal der Universität Hamburg. Sowohl die Messung der Wellenneigungen mittels CISG (Kapitel 2.5.1) als auch Infrarot-Aufnahmen der Wasseroberfläche zur Bestimmung der Wärmetransferraten, fanden in einem Regenturm des Windkanals (siehe Abbildung 4.1) statt.

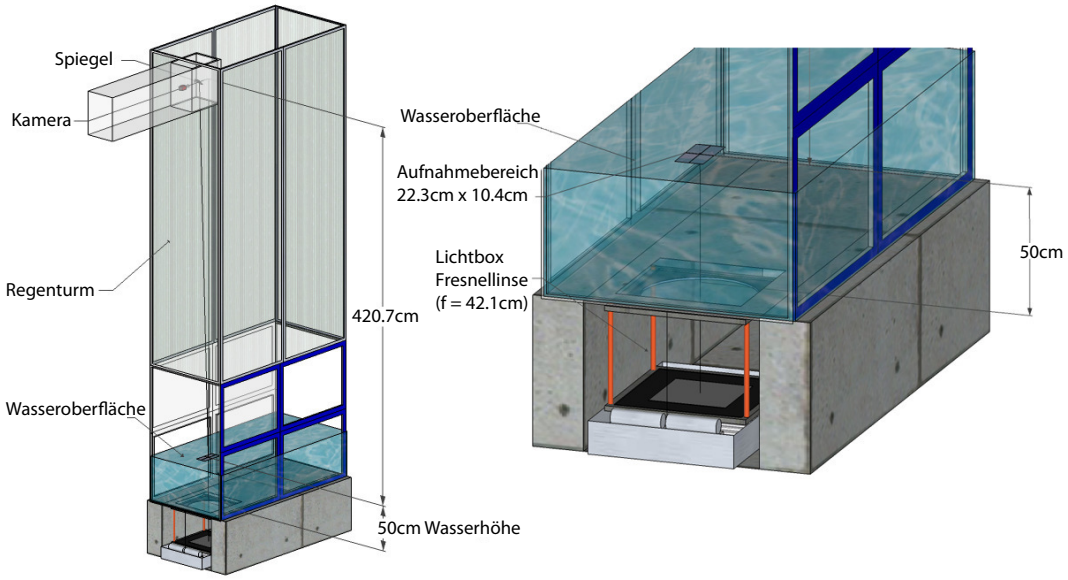

Abbildung 4.1: Regenturm am Windkanal Hamburg (modifiziert nach *Rocholz* [29])

Der in Abbildung 4.1 rechts zu erkennende *Image-Sector* ist der durch CISG aufgenommene und damit in WaveVis dargestellte Bereich der Wasseroberfläche. Die Daten der 150, unter verschiedenen Bedingungen durchgeführten, kontinuierlichen

Messungen wurde bei einer Aufnahmefrequenz von 312Hz zu je 5000 Frames gewonnen.
Am oberen Ende des Regenturms befand sich die Anlage zur Erzeugung des Regens. Dieser in Abbildung 4.2 schematisch dargestellte Aufbau wurde über eine Wasserzufuhr (rechts im Bild) befüllt. Die Regenrate konnte über die Füllstandshöhe der Tanks kontrolliert werden, was durch einen in der Höhe verstellbaren Abfluss (links im Bild) erreicht wurde. Die Regentropfen wurden durch im Boden eingefasste einheitliche Injektionsnadeln erzeugt, wodurch eine konstante Tropfengröße erreicht werden sollte.

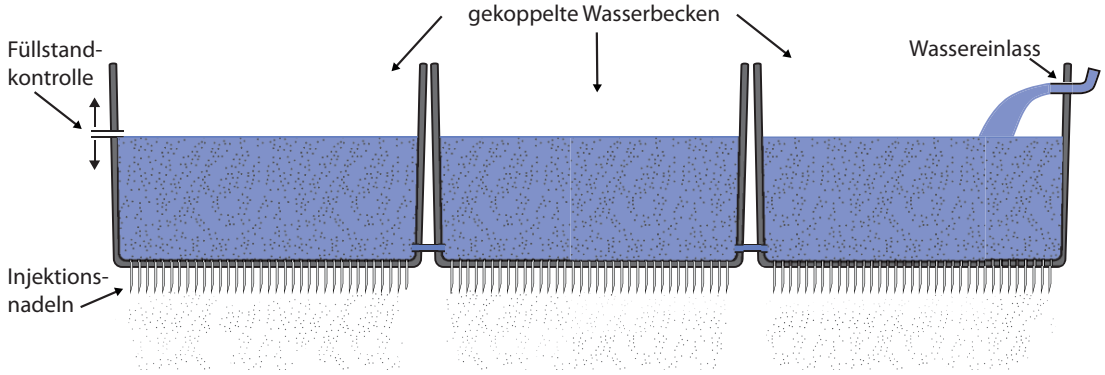

Abbildung 4.2: Schematisch - Beregnungsanlage in Hamburg

4.2 Segmentierung von Regentropfen

4.2.1 Einleitung

Der Versuch Regentropfen zu segmentieren liegt darin begründet, dass im Laufe der WiSSCy-Messkampagne in Hamburg mehrere Datensätze unter Regenbedingungen aufgenommen wurden, da der Messaufbau aber (siehe Abbildungen 4.1) keine direkten Daten zu Regenraten lieferte. Es wurden bei Beginn der Versuchsreihe Regenraten von 110 $\frac{mm}{h}$ bestimmt, die aber im Laufe der Zeit vermutlich Schwankungen unterlagen und gegen Ende der Kampagne nach einer weiteren Messung nur noch zu 57 $\frac{mm}{h}$ bestimmt wurden. Aufgrund dieser Tatsache gab es Unsicherheiten in der Deutung bestimmter Auswertungen, da ein Einfluss schwankender Regenintensität nicht auszuschließen war.
Exemplarisch sei hier die Auswertung der aus den Neigungsdaten bestimmten Werte für das mittlere Steigungsquadrat $\langle s^2 \rangle$ der Wasseroberfläche (*Mean-Square-Slope, mss*) genannt. Hintergrund ist, dass eine Parametrisierung der Gastransfergeschwindigkeit über $\langle s^2 \rangle$ aussagekräftiger ist als über die Windgeschwindigkeit [29]. Unter Einfluss von Regen und der damit einhergehenden Erhöhung von $\langle s^2 \rangle$ verliert aber auch diese Parametrisierung ihre Aussagekraft. Geht man nach Ho et al. [17] davon

aus, dass die durch Regen verursachten Ringwellen keinen Beitrag zur oberflächennahen Turbulenz leisten, sollte sich der Effekt bei Bestimmung des Beitrages dieser Wellenart zum $\langle s^2 \rangle$ herausrechnen lassen. Dafür wird in Rocholz [29] zwischen „in Windrichtung“ (*Lee*) und „gegen die Windrichtung“ (*Luv*) propagierenden Wellen unterschieden. Da angenommen werden kann, dass windgetriebene Wellen nahezu ausschließlich in Windrichtung laufen werden, sollte sich dadurch der Beitrag der Ringwellen ermitteln lassen, wenn der Lee-Anteil bzw. Luv-Anteil von $\langle s^2 \rangle$ über $\langle s^2 \rangle$ geplottet wird (siehe Abbildung 4.3).

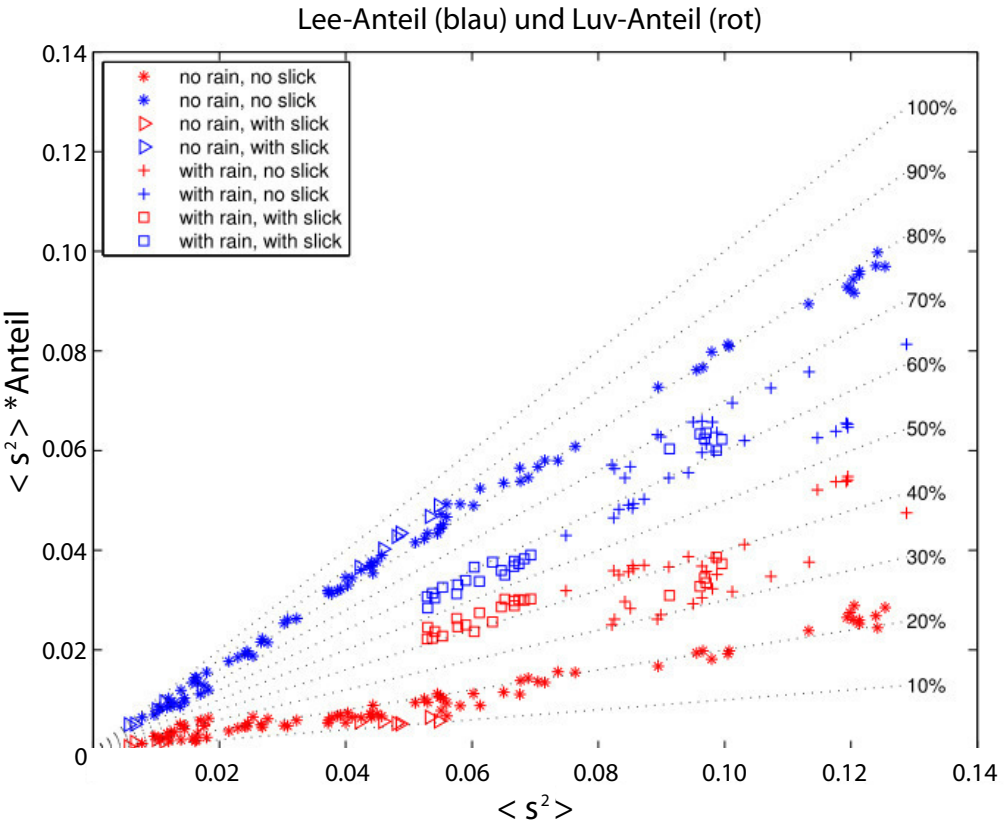

Abbildung 4.3: Lee-Luv Anteil der mittleren quadratischen Neigung gegen $\langle s^2 \rangle$ (aus *Rocholz* [29])

Zu erwarten ist, dass mit steigender Windgeschwindigkeit der Wert für $\langle s^2 \rangle$ wächst, was aber bei Messungen ohne Regen keinen Einfluss auf den Anteil der in Windrichtung propagierenden Wellen haben sollte [29]. Genau dieses Verhalten spiegelt Abbildung 4.3 wieder. Allerdings lässt sich bei den unter Regeneinfluss gewonnenen Datenpunkten ein starkes Streuverhalten erkennen. Ob diese breite Varianz von den im Experiment nicht gleichbleibenden Regenraten herrührt wird im Folgenden untersucht.

4.2.2 Segmentierung und Bestimmung der Regenraten

Zur Segmentierung von Regentropfeneinschlägen auf der Wasseroberfläche wurden die drei Standardkanäle, Höhe und Steigungen der WiSSCy-Daten verwendet. Ein Trainingssatz bestand aus 10 Beispielen verschiedener Messreihen. Nur 0.2% eines Datensatzes wurden als Trainingsdaten verwendet. Ein Beispiel aus dem in Ilastik benützen Trainings-Satz ist in folgender Abbildung zu sehen.

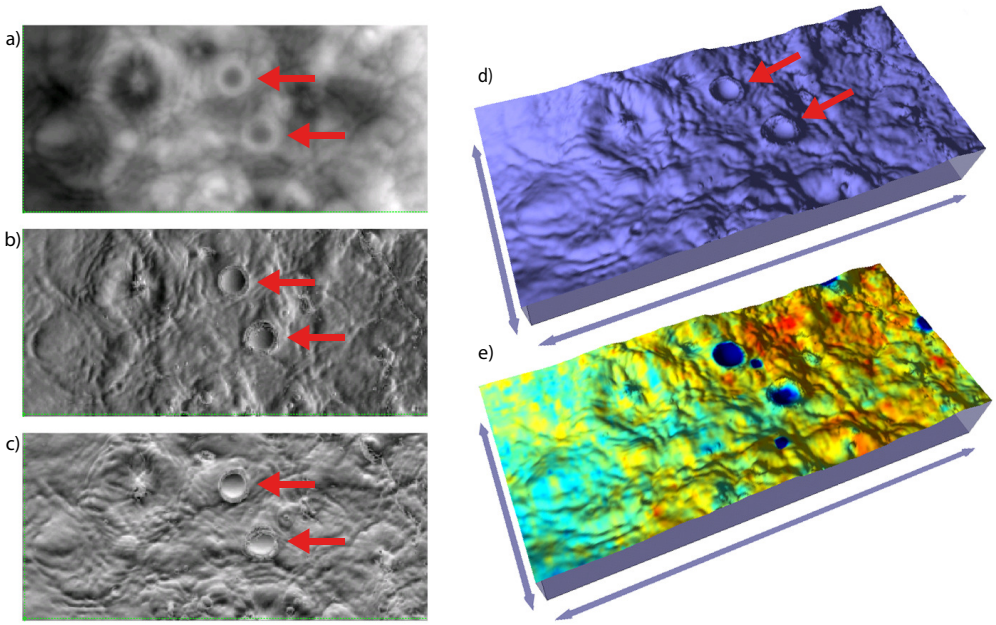

Abbildung 4.4: Für Training verwendete Kanäle: in a) Höhenkarte, b) Steigungen in x- und in c) Steigungen in y-Richtung. In d), Darstellung der drei Kanäle in *WaveVis*. In e) Infrarot-Kanal als Overlay in *WaveVis*. Die roten Pfeile weisen auf Regentropfeneinschläge

Die von Regentropfen verursachten Krater lassen sich, wie man sieht, in den Steigungs-Kanälen (Abbildung 4.4 b und c) sehr gut erkennen, weswegen in diesen auch bevorzugt die Trainingsklassen mittels Ilastik markiert wurden. Zusätzlich wurde derselbe Frame auch in WaveVis dargestellt (4.4 d und e). Man sieht, dass die Regentropfeneinschläge auch im Infrarot-Kanal eindeutig zu erkennen sind. Die Segmentierung der Ereignisse wurde aber ganz bewusst ohne diesen Kanal durchgeführt, um Methoden unabhängig von gleichzeitig aufgenommenen IR-Daten zu entwickeln.

Es wurden zwei Klassen gewählt, Regentropfeneinschlag (rot), im Folgenden häufig auch als Ereignis bezeichnet, und Nicht-Ereignis (grün). Als Features wurden Color, Texture und Orientation in den Stärken *Large* und *Huge* gewählt (siehe Kapitel 2.3). Die daraus resultierende Wahrscheinlichkeitskarte für die Klasse Regentropfeneinschlag ist für je einen Beispiel-Frame mit diesem Ereignis (Abbildung 4.5 a) und einen ohne den Einschlag eines Tropfens (Abbildung 4.5 b) dargestellt.

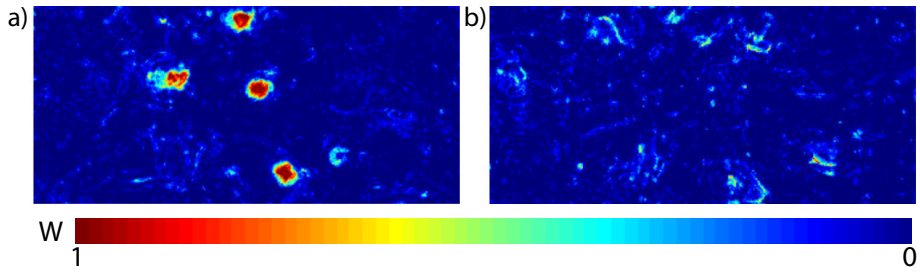

Abbildung 4.5: Original Segmentierung, Wahrscheinlichkeit für das Ereignis Regentropfeneinschlag, in a) mit 4 Ereignissen, in b) ohne Regentropfeneinschläge. Gut zu erkennen ist ein, durch Unsicherheiten des *Random-Forest* verursachtes, Rauschen.

Es ist gut zu erkennen, dass in a) hohe Wahrscheinlichkeiten für vier durch Regen verursachte Krater berechnet wurden. Abbildung 4.5 b) hingegen weist keine größeren zusammenhängenden Bereiche hoher Wahrscheinlichkeiten auf, allerdings erkennt man in Beiden eine Art Rauschen, dass dadurch verursacht wird, dass sich Pixel einer bestimmen Charakteristik im Merkmalsraum nicht eindeutig vom Unterraum der Ereignisse „Einschlag“ trennen lassen und deshalb mit einer gewissen Wahrscheinlichkeit diesem Raum zugeordnet werden.

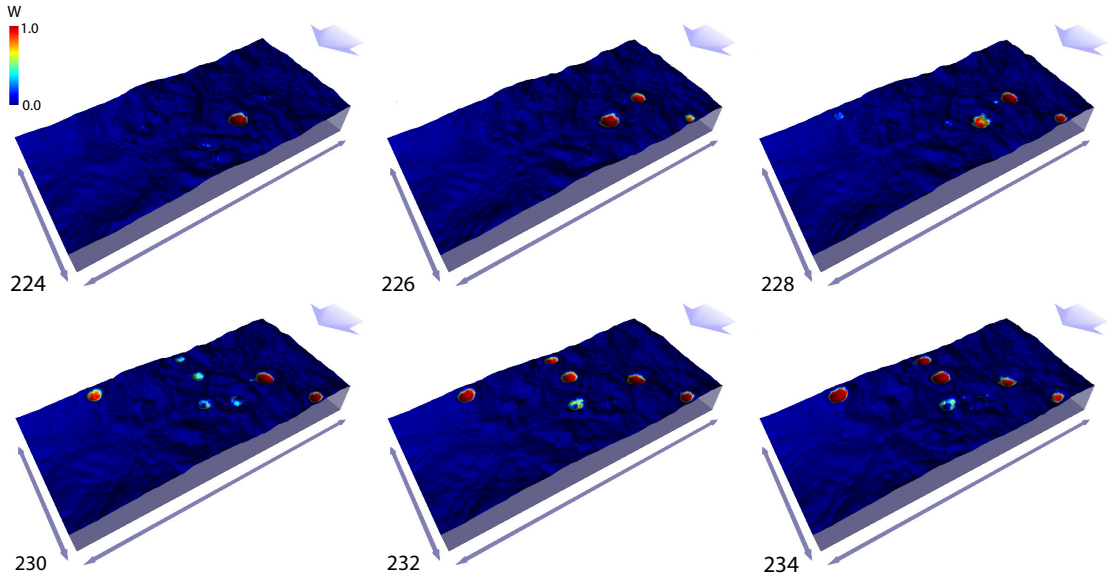

Abbildung 4.6: Exemplarisches Ergebnis einer Regensegmentierung, dargestellt in WaveVis über insgesamt 10 Frames ($\approx 0.03\,\mathrm{s}$) in Schritten von 2 Frames. Man erkennt den Einschlag von insgesamt 6 Regentropfen

Im Mittel sind die Einschlagereignisse über acht bis zehn Bilder hinweg in der Segmentierung zu erkennen. Die Idee, zusammen gehörende Gebiete hoher Wahrscheinlichkeit als von Regentropfen verursacht zu erkennen und zu zählen, bestand darin,

Volumen mit der Zeit als dritter Dimension zu bilden. Dies sollte ermöglichen, die Segmentierung der Einschlagskrater als ellipsoide Gebilde im Orts-Zeit-Raum zu identifizieren. Der hierfür verwendete Algorithmus lässt sich als Pseudocode folgendermaßen beschreiben:

Pseudocode Zählalgorithmus:

1. Definiere Gesamtzahl der zu berechnenden Frames: $totalFrames$
2. Definiere Volumengröße: $volSize$
3. Definiere: $vol\_index = 0$
4. Definiere gesamte Anzahl an Tropfen: $n = 0$
5. Erzeuge Volumen von Frame vol_index bis Frame $volSize \rightarrow volume$

6. Wiederhole für i=0 bis $i = \frac{totalFrames}{volSize} - 1$: ($\frac{totalFrames}{volSize} \in \mathbb{N}$)
   a) Zähle Ereignisse in letzter Ebene des Volumens: n_{last}
   b) Wende $connectedComponents()^*$ auf Volumen an
   c) Setze $n = n + max(connectedComponents(volume))$
   d) Wenn $i < totalFrames/volSize - 1$:
   e) Erhöhe vol_index um $volSize$
   f) Erzeuge Volumen von vol_index bis $vol\_index+volSize \rightarrow volume$
   g) Zähle Ereignisse in erster Ebene des neuen Volumens: n_{first}
   h) wenn $n_{last} > 0$:

      wenn $n_{last} >= n_{first}$:

      $n = n - n_{first}$

      sonst:

      $n = n - n_{last}$

(*) Mit $connectedComponents()$ ist dabei die Methode $vigra.analysis.labelImage()$, beziehungsweise $vigra.analysis.labelVolume()$[1] gemeint, die zusammenhängende Gebiete erfasst und durchnummeriert. Der größte Index des resultierenden Arrays ist dementsprechend die Anzahl der zusammenhängenden Gebiete.
In Punkt 6h) des Pseudocodes werden die Ränder der beiden Volumen auf überschneidende Ereignisse überprüft und die Anzahl der gezählten Tropfen dementsprechend korrigiert. Dabei wird allerdings der Fall vernachlässigt, in dem Ereignisse genau an der einen Volumengrenze enden und nicht in das nächste, beziehungsweise das vorherige Volumen hineinragen. Dadurch werden zu viele Überschneidungen

[1] Bestandteil der Bildverarbeitungsbibliothek Vigra: http://hci.iwr.uni-heidelberg.de/vigra/

„gesehen“ und damit insgesamt zu wenig Ereignisse gezählt. Um diesen Fall mit einzuschließen, müssten die Positionen der Ereignisse ermittelt werden und ein Vergleich über Anzahl und Ort der Ereignisse an den Rändern stattfinden. Allerdings wurde der durch diese Vernachlässigung möglicherweise entstehende Fehler bei der großen Anzahl an Ereignissen (im Mittel 750), der geringen Anzahl an betrachteten Volumen (es wurde in 8 Teilvolumen unterteilt) und der damit einhergehenden niedrigen Wahrscheinlichkeit des Auftretens dieses Falles, als sehr gering eingeschätzt (Vergleiche Abschnitt 4.2.4).

Wie in Abbildung 4.7 ersichtlich, ist ein aus stark verrauschten Daten gebildetes Volumen allerdings nicht sehr erfolgversprechend. Das in Abbildung 4.5 dargestellte Beispiel einer Segmentierung ist eine Teilebene des Volumens aus Abbildung 4.7.

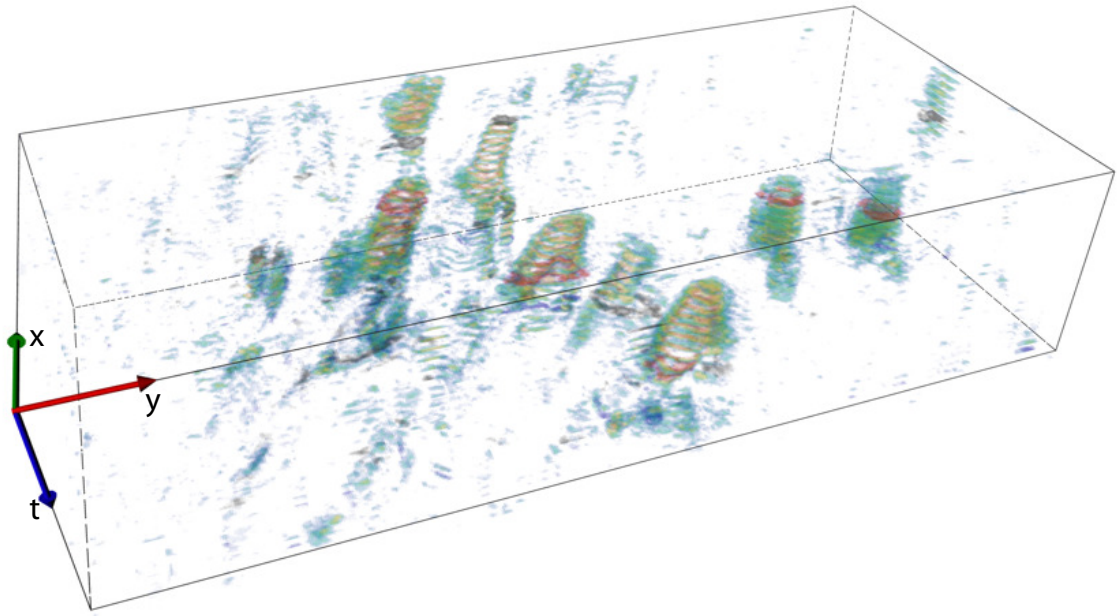

Abbildung 4.7: Orts-Zeit-Volumen aus original Segmentierung

Daher wurde die Erzeugung des Volumens (in obigem Pseudocode Schritt 5 und 6f) in eine Klasse ausgelagert, deren Aufgabe es ist, dass zur Prozedur verwendete Volumen so aufzubereiten, dass nur noch die reinen Ereignisse im Volumen vorhanden sind. Dafür werden die einzelnen Frames geladen und in einem ersten Schritt über einen Schwellwert T_1 binärisiert. Der Schwellwert wurde bei einer 40-prozentigen Wahrscheinlichkeit für die Zugehörigkeit zur Klasse Regentropfeneinschlag gesetzt, um nicht zu viel der zusammenhängenden Ereignisgebiete zu verlieren. Das Ergebnis nach diesem ersten Schritt ist in Abbildung 4.8 unter Verwendung des Beispiels aus Abbildung 4.5 zu sehen.

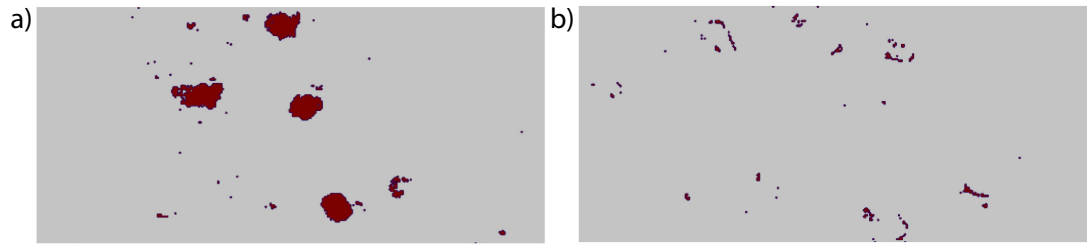

Abbildung 4.8: Über Schwellwert $T_1 = 0.4$ binärisierte Segmentierung aus Abbildung 4.5, in a) Beispiel mit Einschlag-Ereignissen, in b) ohne Regentropfeneinschläge

Durch diesen Schritt wurde das Rauschen noch nicht vollständig unterdrückt, weswegen in einem nächsten Schritt wieder die Methode der *connectedComponents*(), allerdings im zweidimensionalen Fall, zum Einsatz kam. Wie oben besprochen ist der Rückgabewert dieser Funktion ein Array, dessen Einträge die zusammenhängenden Pixel einer Zahl zuordnen. Dadurch lassen sich die Größen der Gebiete ermitteln und durch einen zweiten Schwellwert T_2 kleine Regionen aus dem Bild entfernen. Hierfür wurden die Flächen der Ereignisse und des Rauschen analysiert. Die Ereignisse weisen danach Ausmaße von 760 ± 254 Pixeln, die Rausch-Regionen 134 ± 98 Pixel auf. Daher wurde ein Schwellwert $T_2 = 300$ gewählt.

Das Ergebnis der Größenselektion, angewendet auf die Beispiele aus Abbildung 4.8, ist in den Bildern 4.9 zu sehen.

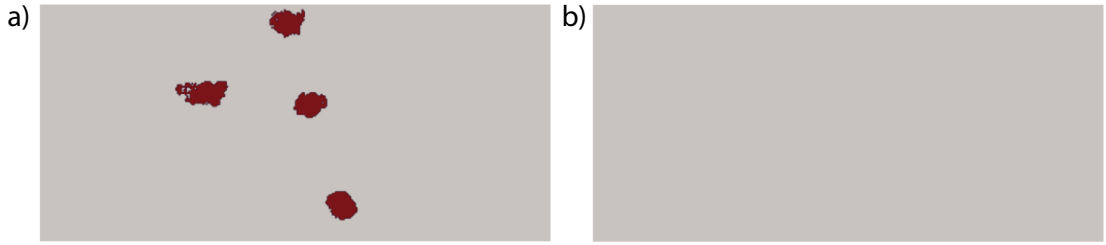

Abbildung 4.9: Über Größenselektion nachbearbeitete, binäre Segmentierung aus Abbildung 4.8, a) Beispiel mit Einschlag-Ereignissen, b) ohne Regentropfeneinschläge

Aus den auf diese Weise nachbearbeiteten Einzelbildern wird das Volumen gebildet. In Abbildung 4.10 ist exemplarisch ein, nach der beschriebenen Prozedur erzeugtes, Volumen dargestellt. Die blauen Linien dienen dabei der Orientierung des Betrachters und helfen, die Ereignisse räumlich beziehungsweise zeitlich zuzuordnen. Sie sind jeweils in den Schwerpunkten der letzten Ebene eines Ereignisses gesetzt und Enden am Rande des gesamten Volumens in einer Kreisfläche. Das Volumen wurde aus 24 Frames gebildet und entspricht damit einem Zeitraum von 76.9 ms.

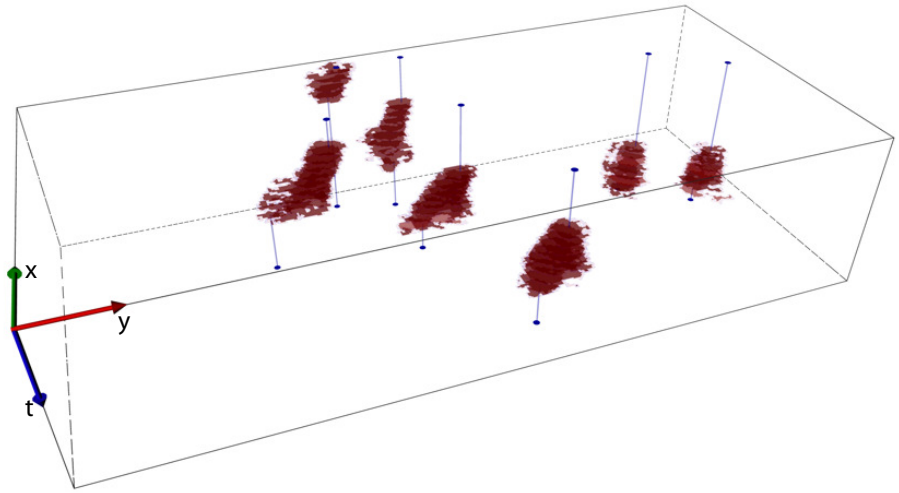

Abbildung 4.10: Zur Zählung der Ereignisse aufbereitetes Volumen. Die Abbildungen 4.5, 4.8 und 4.9 sind eine Ebene des Volumens (Die blauen Linien dienen nur zur optischen Orientierung)

Der Algorithmus arbeitet nach beschriebenen Schema $(totalFrames/volSize) - 1$ gebildete Volumen durch und addiert die gefunden Ereignisse auf. Die Regenraten pro Datensatz wurden dann aus der Anzahl Tropfen n, des Durchmessers eines Tropfens[1] d_o und der beregneten Fläche $A_\square$ wie folgt bestimmt:
Volumen eines Tropfens in mm^3:

$$V_o = \frac{\pi}{6} * d_o^3 \quad \text{mit:} \qquad d_o = 2.9\,\mathrm{mm} \tag{4.1}$$

Regenrate in $\frac{\mathrm{mm}}{\mathrm{h}}$:

$$R = \frac{nV_o}{A_\square t} \quad \text{mit:} \quad A_\square = 223.4 \cdot 104\,\mathrm{mm}^2 \quad \text{und} \quad t = \frac{Frames}{312\,\mathrm{Hz} \cdot 3600\frac{\mathrm{s}}{\mathrm{h}}} \tag{4.2}$$

Grundsätzlich könnte die Ursache für eine Abnahme der Regenrate im Laufe der mehrtägigen Messkampagne auch in einer Verkalkung der Kanülen, einhergehend mit einer Verringerung der Tropfengröße, zu finden sein. Dies würde den Ansatz nach den Gleichungen 4.1, 4.2 nicht rechtfertigen und zu falschen Ergebnissen führen. Um dies auszuschließen, wurde ein Vergleich der durchschnittlichen Kratergrößen zu Beginn und am Ende der Kampagne durchgeführt. Da keine diesbezügliche Abnahme festzustellen war kann davon ausgegangen werden, dass der die Krater verursachende Impuls sich nicht mit der Zeit verändert hat.

[1] Nachzulesen in Rocholz [29] S.112

4.2.3 Ergebnis

Das beschriebene Verfahren wurde im Weiteren auf die Daten der *WiSSCy-Kampagne* angewandt und das Ergebnis analog zu Abbildung 4.3 erneut geplottet (Abbildung 4.11). Allerdings mit dem Unterschied, dass nur der Luv-Anteil betrachtet wird und das in jedem Datenpunkt zusätzlich die Information über die Windgeschwindigkeit durch die Farbe und die Größenordnung der berechneten Regenrate durch das Symbol dargestellt wird.

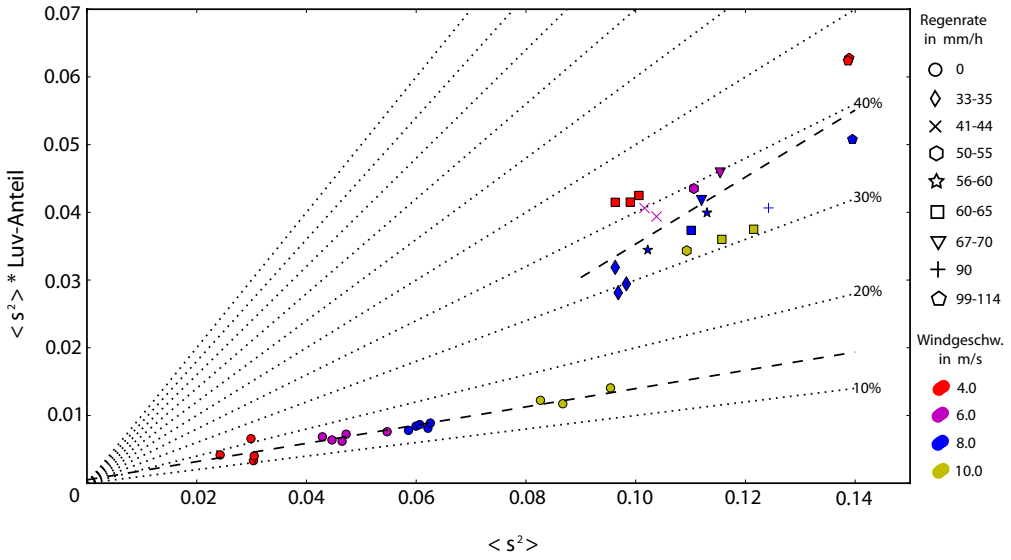

Abbildung 4.11: Luv-Anteil bei unterschiedlichen Regenraten und Windgeschwindigkeiten. Im Bereich von 15% finden sich Daten ohne Beregnung. Bei Daten unter Beregnung aufgenommen lässt sich eine breite Streuung von 30-45% erkennen.

Aus dieser Darstellung lässt sich zuerst einmal grundsätzlich erkennen, dass bei Abwesenheit von Niederschlag der Wert von $\langle s^2 \rangle$ mit der Windgeschwindigkeit zunimmt, der Anteil an Wellen entgegen der Windrichtung allerdings, wie zu erwarten, konstant bei circa 15% liegt. Unter Regenbedingungen lässt sich für die breite Streuung der Daten in Richtung des Luv-Anteils eine Abhängigkeit von der Windgeschwindigkeit erkennen. Die Ursache hierfür ist, dass nach Kapitel 2.4 $\langle s^2 \rangle$ proportional zur Energie der Kapillarwellen ist und der prozentuale Anteil dieser, bei hohen Windgeschwindigkeiten, in Windrichtung stärker ins Gewicht fällt als bei niedrigen. Die Information der Regenraten zeigt eine deutliche Tendenz zu einer Erhöhung von $\langle s^2 \rangle$ bei hohen Regenraten, was klar ist, da die Wasseroberfläche durch die von den Regentropfen herrührenden Ringwellen stark aufgeraut wird. Das eine nicht gleichbleibende Regenrate allerdings zu einer Verbreiterung des Luv-Anteils beiträgt, kann aus diesem Ergebnis nicht klar erkannt werden. Die nicht eindeutig zu interpretierende Verteilung der Datenpunkte in Abhängigkeit von den Regenraten kann ihre Ursache darin haben, dass im Laufe der Messkampagne auch unter Einfluss von Oberflächenfilmen (siehe Kapitel 2.4) gemessen wurde, was die Wellenbildung unter

Umständen stark dämpfen kann. Ist das Wasser für nachfolgende Messreihen nicht vollständig frei von den verwendeten Substanzen gewesen, wird dies Einfluss auf die Werte von $\langle s^2 \rangle$ nehmen und eine Deutung der Messergebnisse schwierig machen.

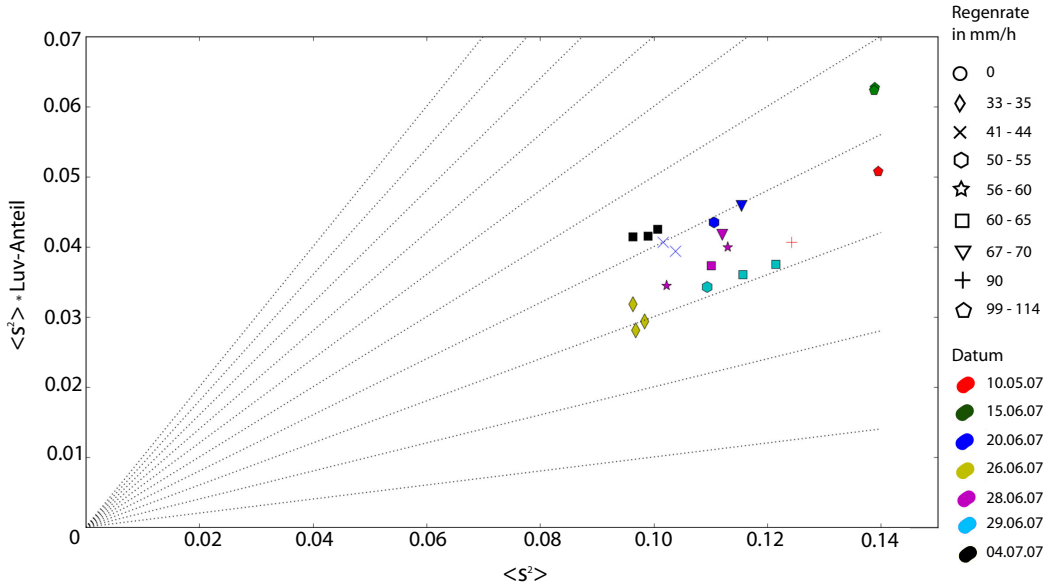

Abbildung 4.12: Luv-Anteil bei unterschiedlichen Regenraten im Verlauf der Messkampagne.

In Abbildung 4.12 wurden deshalb die verschiedenen Regenraten nochmals nach dem Zeitpunkt der Messung aufgeschlüsselt, woraus ersichtlich wird, dass sich die Messwerte stark nach dem Datum ihrer Messung gruppieren. Es muss also davon ausgegangen werden, dass die Messbedingungen an den jeweiligen Tagen der Kampagne hier einen stärkeren Einfluss zeigen als die Regenraten selbst.

4.2.4 Genauigkeit des Verfahrens

Fehler in der Segmentierung

Den niedrigsten *Out-Of-Bag-Error* (siehe Kapiel 2.3) erzielte die Feature-Kombination von Color, Orientation und Texture jeweils in *large* und *huge*. Der ausgegebene Fehler lag bei 4.3%.

Fehler durch Volumenunterteilung

Eine weitere Abschätzung erfolgte hinsichtlich der Stabilität der Zählung je nach Größe der dafür verwendeten Teilvolumen und damit des Fehlers, der durch die unvollständige Behandlung der Randbereiche zweier Teilvolumen entsteht. Auf Rechnern mit genügend Arbeitsspeicher lassen sich Daten mit 5000 Bildern und der

hier vorliegenden Auflösung am Stück in den Speicher laden, wodurch keine Fehler durch Randeffekte auftreten können. Steht jedoch nicht genug Speicherkapazität zur Verfügung, muss wie im Pseudocode beschrieben das Gesamtvolumen unterteilt werden. Dabei kann dahingehend ein Fehler in der Zählung der Ereignisse auftreten, dass nach Abschnitt 4.2.2 eine zu geringe Anzahl an Ereignissen ermittelt wird. Um das Ausmaß dieses Fehlers festzustellen, wurden die Gesamtvolumen mehrerer Datensätze in immer kleinere Volumenpartitionen unterteilt. Aus diesen unterteilten Volumen wurden die Tropfenanzahlen bestimmt und jeweils auf den als fehlerfrei angenommenen Wert aus dem unpartitionierten Volumen normiert. Daraus wurde die prozentuale Abweichung für die verschiedenen Partitionierungen beziehungsweise Größen der Teilvolumen bestimmt.

Partitionen	Volumengröße	Fehler
25	200	0.071%
50	100	0.139%
100	50	0.21%
500	10	1.05%

Für Volumengrößen über 200 Frames konnte aus den zur Verfügung stehenden Datensätzen keine Abweichung zur Zählung im kompletten Volumen gefunden werden.

Güte des Algorithmus

Um die Güte des Algorithmus beschreiben zu können wurde untersucht, welche Menge an Einzelbildern benötigt werden um eine gute Auskunft über die durchschnittliche Regenrate zu bekommen. Ein Datensatz besteht aus 5000 Frames und man erhält daher natürlich die beste Aussage, wenn alle zur Zählung der Tropfeneinschläge herangezogen werden. Allerdings benötigt der Algorithmus für einen vollständigen Datensatz recht lange. Kann aber angenommen werden, dass die Regenintensitäten über die Dauer eines Datensatzes nicht zu stark schwanken, sollte man schon bei weniger als 5000 Bildern mit guten Ergebnissen rechnen können. In Abbildung 4.13 wird für die 21 verwendeten Datensätze die Abhängigkeit der Regenraten von der Anzahl der zu deren Bestimmung verwendeten Frames dargestellt. Wie man sieht, stellen sich etwa ab 2000 zur Zählung verwendeter Frames relativ stabile Regenraten ein.

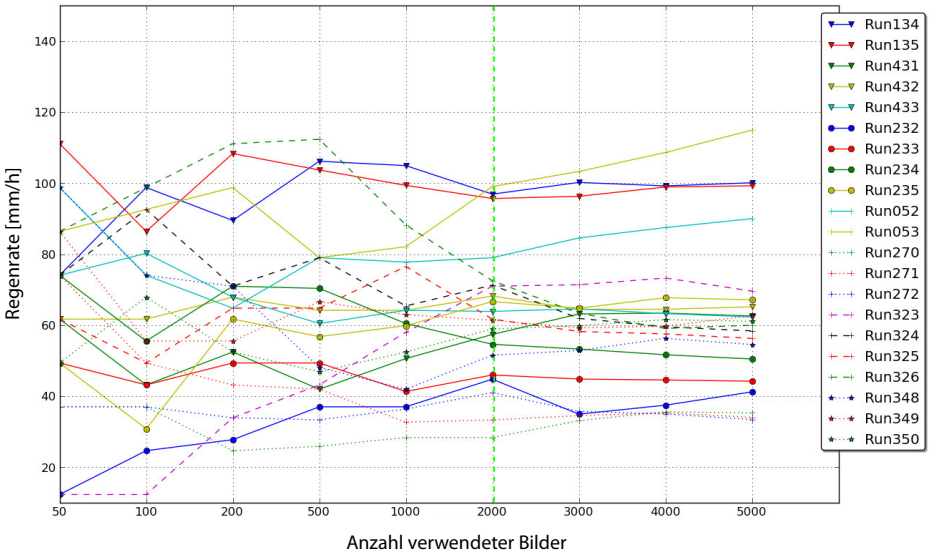

Abbildung 4.13: Ermittelte Regenrate je nach Anzahl verwendeter Frames.

Als zusätzliche Information lässt sich aus Abbildung 4.13 außerdem auch die eingangs beschriebene Vermutung einer tendenziellen Abnahme der Regenrate im Laufe der Messkampagne erkennen. Die Datensätze mit den höchsten Regenraten waren Aufnahmen zu Beginn der Messkampagne; im Gegensatz zu der zweiten Gruppe in diesem Plot, die allesamt bei Regenraten unter 75 $\frac{mm}{h}$ liegen. Noch deutlicher wird dies in Abbildung 4.14. Hier wurde aus den verwendeten Datensätzen für jeden Tag der Messkampagne Mittelwert und Standardabweichung dargestellt.

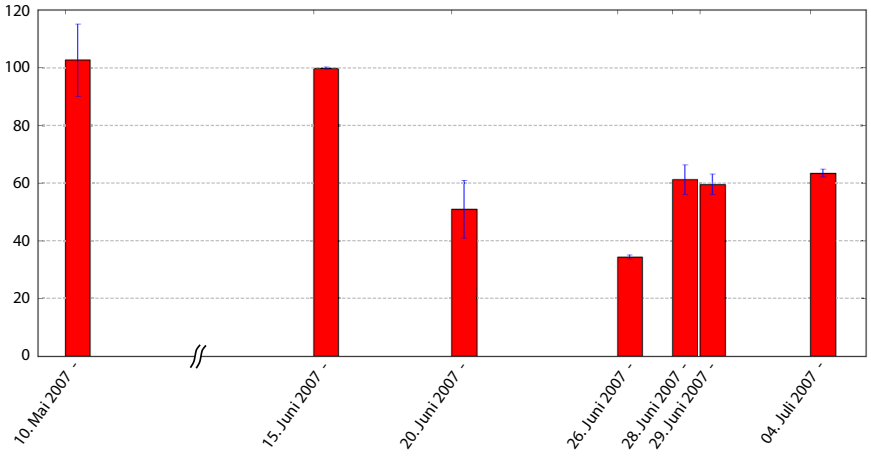

Abbildung 4.14: Schwankung der Regenraten im Verlauf der Messkampagne. Man erkennt, dass die Regenraten nicht durch starke Ausreißer bestimmt werden sondern gemäßigt um den Mittelwert schwanken. Im Anhang finden sich die Schwankungen aller verwendeten Daten.

Weiterhin wurde überprüft, ob die ausgegebenen Regenraten dahingehend zuverlässig sind, dass sie durch genügend homogene Beregnung bestimmt wurden. Dafür wurde für jeden Datensatz die Anzahl der Tropfeneinschläge je 200 Frames bestimmt und exemplarisch für einen frühen (Abbildung 4.15 a) und einen späten Datensatz (Abbildung 4.15 b) der Messkampagne um den Mittelwert und die Standardabweichung geplottet. Hiernach lässt sich beurteilen, ob die ermittelten Regenraten als zuverlässig betrachtet werden können, wenn ihre Werte gleichmäßig um den Mittelwert schwanken und nicht ein durch starke Ausreißer entstandener Mittelwert die Regenrate bestimmt.

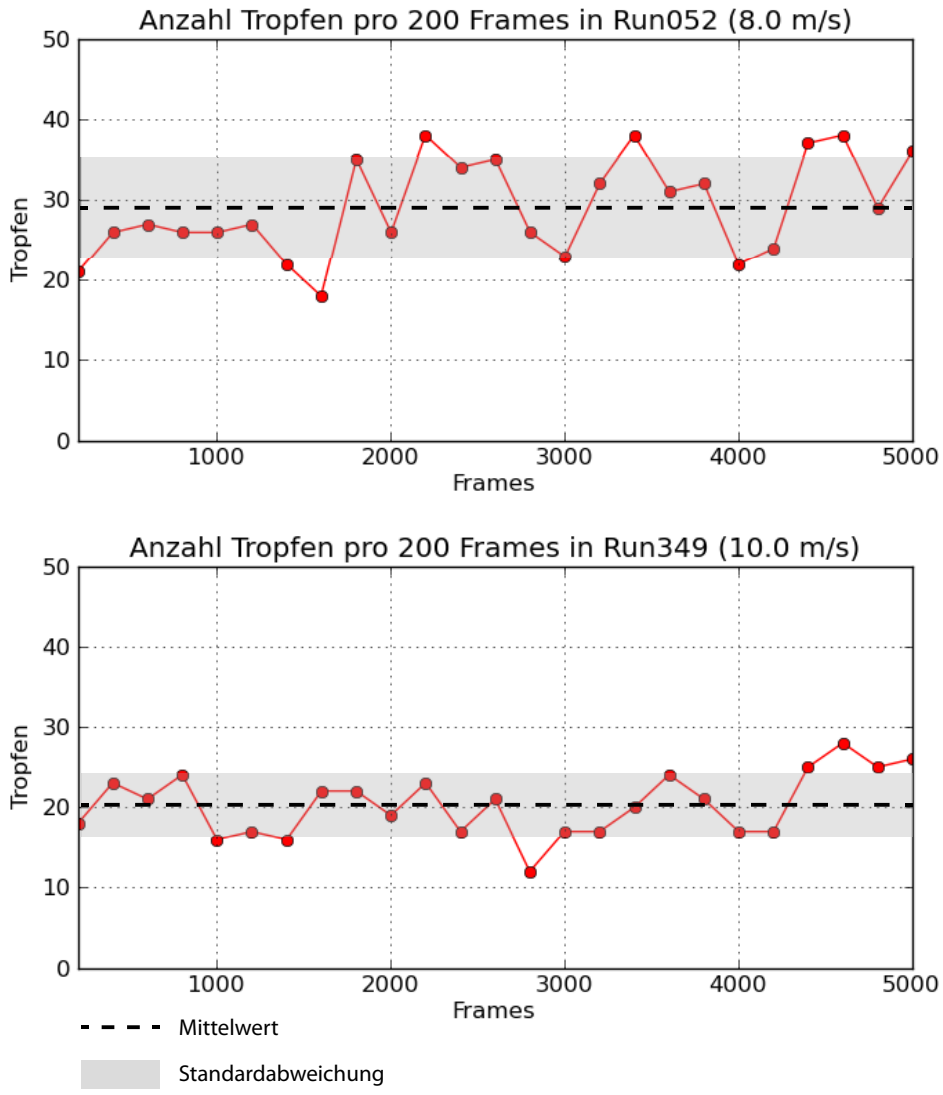

Abbildung 4.15: Schwankung der Regenintensität in Abschnitten von 200 Frames

4.3 Segmentierung von Microscale-Breaking-Waves

4.3.1 Einleitung

Wie in Kapitel 2.5.2 erläutert, beruhen bisherige Verfahren zur Detektion von *Microscale-Breaking* im Wesentlichen auf Thermographie. Die Grundannahme ist bei den vorgestellten Verfahren immer eine erhöhte Umwälzung eines *Tracers* (wie die durch Laser aufgebrachte Wärme) wenn eine brechende Welle das Gebiet der Messung passiert.

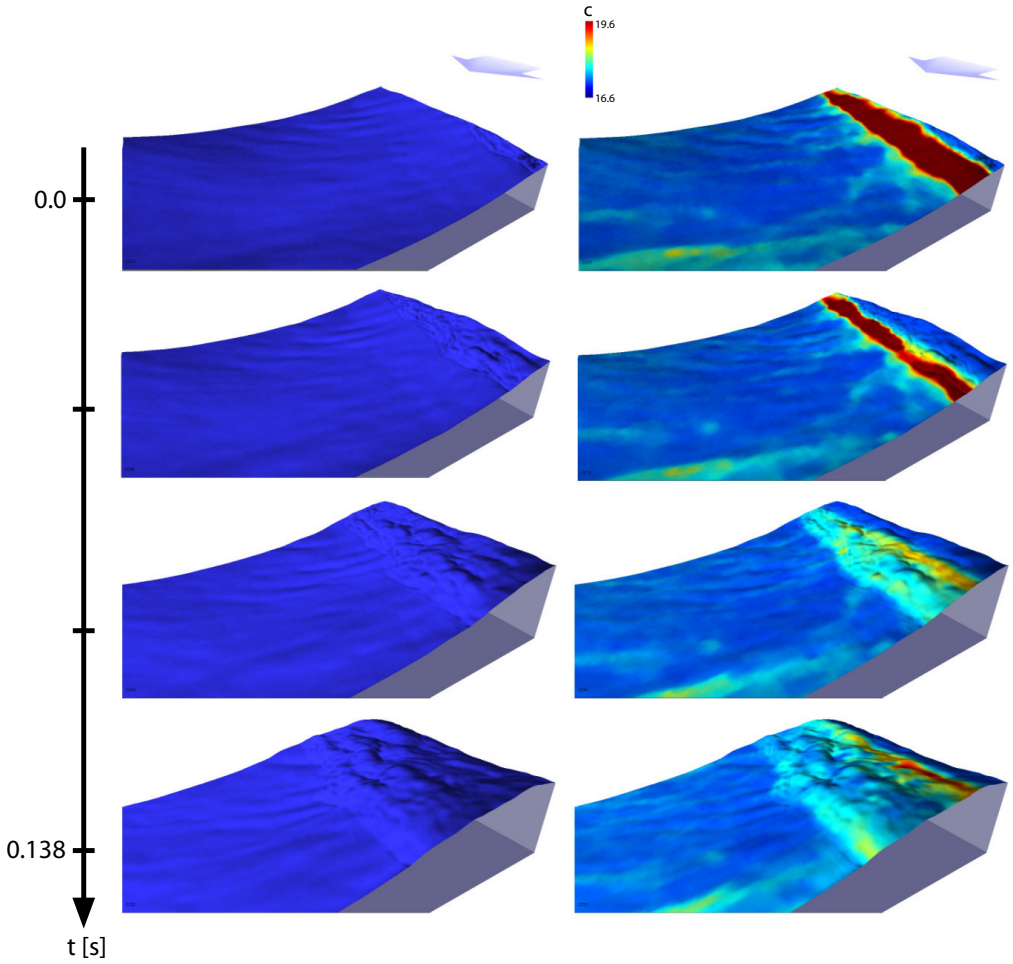

Abbildung 4.16: *Microscale Breaking Wave* identifiziert durch IR-Signatur in WaveVis, links Normalmodus, rechts IR-Overlay

Durch den in Abschnitt 2.5.2 erläuterten Linien-Modus bei der aktiven Thermografie lassen sich die *Microscale-Breaking-Waves* mithilfe von WaveVis (siehe Kapitel 3)

sehr gut identifizieren, da beim Vorüberziehen einer solchen Welle die Wärmesignatur der Laserlinie deutlich abgeschwächt wird (Vergleiche Abbildung 4.16). Untersucht man die Datensätze auf diese Art, stellt man fest, dass sich Wellenzüge, die einen eindeutig erhöhten Wärmetransport in den Wasserkörper verursachen, auch optisch von solchen unterscheiden die diese Signatur nicht aufweisen. Die Turbulenzzonen, die solche Wellen nach sich ziehen, machen sich in den Neigungskanälen durch eine stark variierende Struktur bemerkbar. In Abbildung 4.17 ist ein Beispiel dargestellt. Die von rechts ins Bild laufende Welle weist sehr starke Strukturen auf, die in der am linken Bildrand auslaufenden Welle hingegen nicht zu erkennen sind.

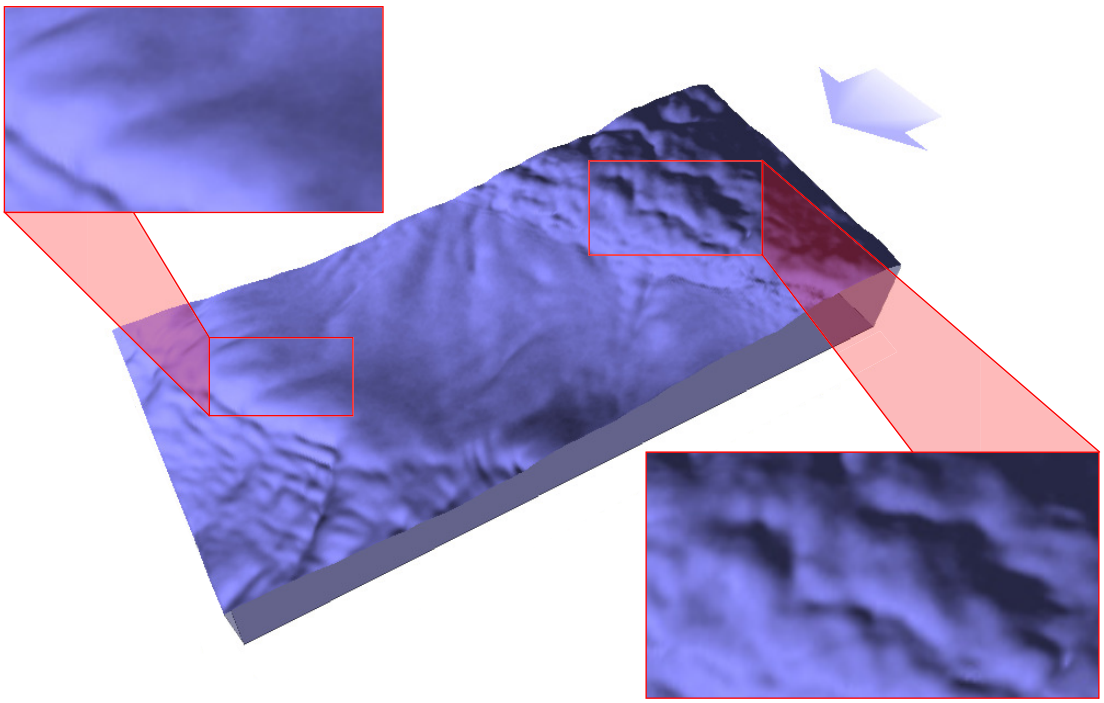

Abbildung 4.17: Strukturunterschiede von nichtbrechender (links) und brechender Welle (rechts)

So eindeutig wie hier ist der Unterschied allerdings nicht immer. Alle Abstufungen zwischen den beiden Extremen sind möglich und häufig lässt sich die durch Turbulenz verursachte Struktur in einem einzelnen Bild kaum ausmachen. In der Animation mittels WaveVis lassen sich im zeitlichen Ablauf die Strukturen mit dem Auge aber meist sehr gut identifizieren.

4.3.2 Segmentierung mit Einzelbildern

Aus der Abbildung 4.17 wird ersichtlich, dass es mit den in Kapitel 2 beschriebenen Verfahren des Random-Forest (Abschnitt 2.1.4) und Ilastik (Abschnitt 2.3) möglich

sein sollte, die Ereignisse *Microscale-Breaking* von anderen Wellen zu unterscheiden. Hierfür wurden aus bis zu 20 Einzelbeispielen (je 10 ohne *Microscale-Breaking*-Ereignis und 10 mit) die Höhen und Neigungsinformation aus verschiedenen Datensätzen ausgewählt und mittels Ilastik ein *Random-Forest* trainiert.

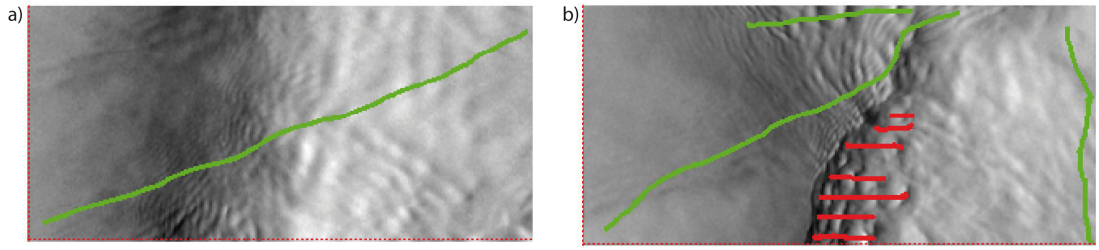

Abbildung 4.18: Zwei gelabelte Beispiele aus dem Trainingssatz, a) ohne Ereignis, b) mit brechender Welle. Die grünen Markierungen sind Trainingslabels für Pixel ohne Ereignis, die roten definieren Pixel im Kontext der gesuchten Struktur.

Es zeigte sich jedoch, dass die von Ilastik standardmäßig bereitgestellten Features zu keinem zufriedenstellenden Ergebnis führten. Die Vielfalt an ähnlichen Strukturen ist in windgetriebenen Wellenfeldern zu groß, so dass eine reine Strukturanalyse durch die Features *Color* und *Texture* (siehe Abschnitte 2.3 und 2.2) nicht ausreichend ist (Die Features *Orientation* und *Edge* bringen bei dieser Art von Daten keinen merklichen Vorteil!). Deshalb wurde zusätzlich zu den Features *Color* und *Texture*, die beide jeweils in den Größen *Large* und *Huge* (siehe Kapitel 2.3) verwendet wurden, ein weiteres Feature entwickelt.

Wellenfront Detektion

Eine Ursache der schlechten Klassifizierungseigenschaften des *Random-Forest* ist der mangelnde Bezug zur Wellenfront. Bereiche stark variierender Grauwertstrukturen treten an vielen Stellen der Wellendaten auf, befinden sich jedoch nicht zwangsläufig in der unmittelbaren Umgebung hinter Wellenfronten. Deshalb wurde zur besseren Detektionsfähigkeit ein zusätzliches Feature in Ilastik implementiert, das den Bezug zu einer vorhandenen Wellenfront herstellt, da die gesuchten Strukturen nur im Kontext einer solchen auftreten können. Hierfür wurde im ersten Schritt ein Canny-Edge-Filter (siehe Kapitel 2.2.4) auf das Orginalbild angewandt. Dieser reagiert, je nach eingestelltem *Threshold*, auf starke Grauwertänderungen und legt senkrecht zu den Gradienten eine, ein Pixel breite Linie (vergleiche Abbildung 4.19, zweite Spalte). Der mittlere Threshold wurde dabei aus der Grauwertdifferenz mehrerer Wellenfronten ermittelt. Im einem zweiten Schritt werden kleine Kanten, die nur wenige Pixel Länge aufweisen entfernt, damit möglichst nur durch Wellenfronten entstandene Grauwertabstufungen mitgenommen werden. Im letzten Schritt wird auf das

Canny-Edge-Bild eine *Distance-Transform* angewendet. Diese codiert einfach nur die Entfernung von jedem Kantenpixel mit einem mit der Entfernung steigenden Grauwert. Der Merkmalsraum wird somit um eine Bedeutungszone in der Gegend von Wellenfronten erweitert. Zur Erhöhung der Detektionsfähigkeit werden Ilastik insgesamt fünf verschiedene Varianten, mit um den Mittelwert verteilten *Thresholds*, zurück gegeben. Dies macht das Feature auf ein breiteres Spektrum von Wellenfronten empfindlich und gibt stark ausgeprägten ein höheres Gewicht, da diese in allen fünf Varianten erfasst werden. In Abbildung 4.19 sind zur Veranschaulichung die Resultate für den niedrigsten und den höchsten *Threshold* für je ein Beispiel mit und eines ohne Ereignis dargestellt.

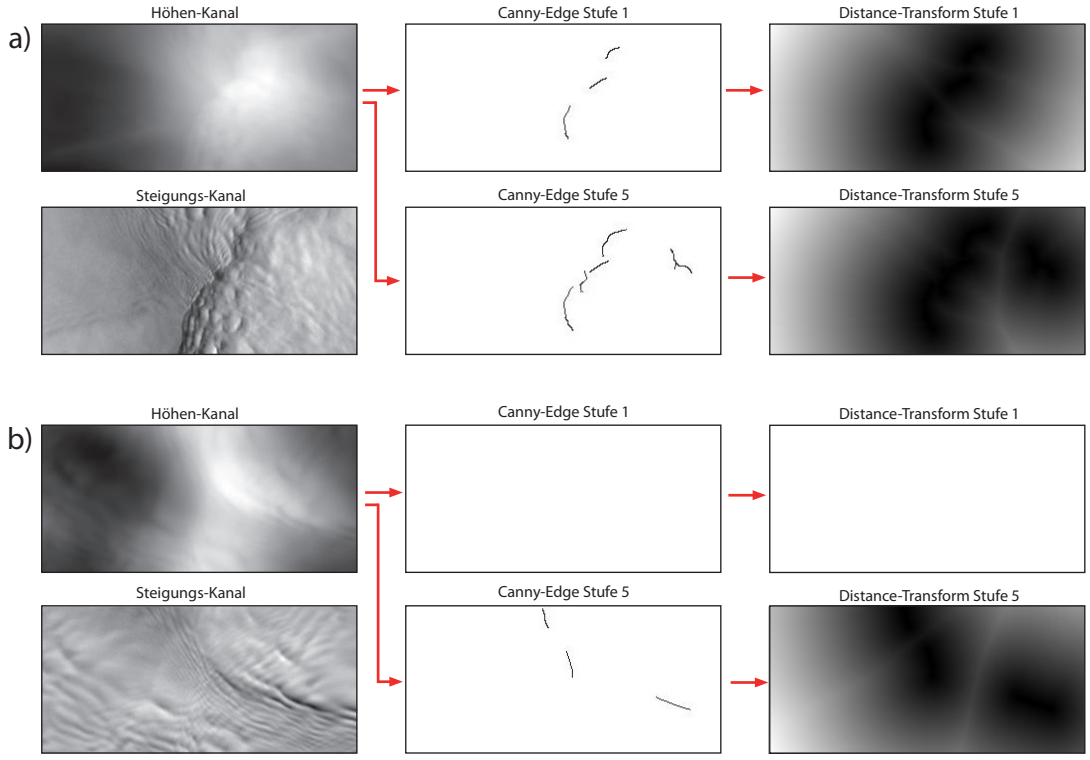

Abbildung 4.19: Anwendung des Wellenfront-Features auf den Höhenkanal einer brechenden Welle in a) und den Höhenkanal eines Beispiels ohne brechende Welle in b) (Die Abbildung des Steigungskanal dient hier nur zur besseren Unterscheidung der beiden Beispiele). Dargestellt sind jeweils die auf Grauwertänderungen unempfindlichste Stufe 1 und die empfindlichste Stufe 5. In den *Distance-Transform*-Abbildungen ist der Abstand von den gefundenen Kanten über einen zunehmenden Grauwert codiert.

Aus Abbildung 4.19 wird ersichtlich, dass dieses zusätzliche Feature nur Fehldetektionen verhindern kann, wenn keine Wellenfronten beziehungsweise keine zu starke Kanten in der Nähe kritischer Bereiche im Bild gefunden werden. In dem in b) dargestellten Beispiel werden auf mindestens der höchsten Empfindlichkeitsstufe trotzdem Kanten gefunden, obwohl keine brechende Welle vorhanden ist. Zwischen ausgeprä-

ten Wellenfronten von brechenden und nicht-brechenden Wellen kann dieses *Feature* daher nicht unterscheiden. Dies führt aber trotzdem schon zu einer signifikanten Verbesserung der Segmentierung.

Probleme mit der Detektion von Microbreaking basierend auf Einzelbildern

Das bisher beschriebene Verfahren zur Detektion von *Microscale Breaking* beruht darauf, dass in Ilastik ein *Random-Forest* mit einem Satz aus Einzelbildern trainiert wird. Es werden also ausgewählte einzelne Zeitschritte als Trainingssatz verwendet. Dies bringt einige Probleme mit sich.

Fehlender Zeitbezug

Der *Random-Forest* soll darauf trainiert werden Regionen hoher Variabilität in zwei der drei zur Verfügung gestellten Kanälen zu finden. Ohne Kontext ist diese Herangehensweise allerdings nicht sehr erfolgversprechend, da solche, den gewünschten Regionen sehr ähnliche Strukturen, häufig auftreten und es selbst dem Mensch schwerfällt, ohne Zeitbezug die Strukturen sicher auszumachen. Daher wurde versucht, zumindest den Kontext zur Wellenfront herzustellen, indem deren Umgebung eine Bedeutung gegeben wird. Da aber zunächst auf einzelnen Zeitschritten sowohl trainiert als auch klassifiziert wird, geht dieser Kontext sofort verloren, sobald die Wellenfront den Bildbereich verlässt.

Abhängigkeit von der Höhe der Wellenfront

Das Feature zur Wellenfrontanalyse arbeitet wie beschrieben mit fünf verschieden sensiblen Canny-Edge-Detektoren. Das Problem dabei ist, dass die Stärke der Wellenfront und das Auftreten hoher Verwirbelungen in keinem klaren Zusammenhang stehen. Deutlich wird dies in den folgenden Abbildungen 4.20 und 4.21.

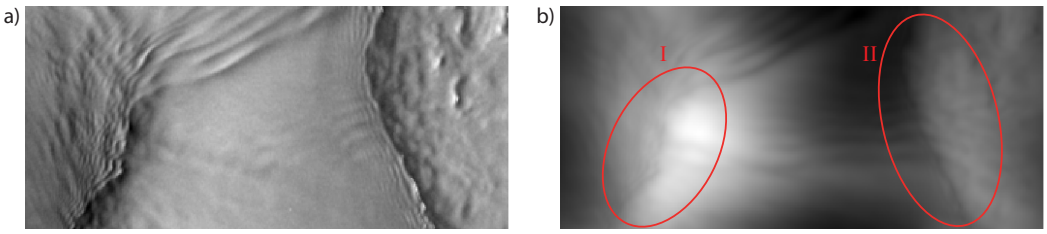

Abbildung 4.20: Probleme der Wellenfrontdetektion, a) Situation im Steigungskanal, b) im Höhenkanal

In Abbildung 4.20 ist in a) ein Beispiel mit zwei Wellenfronten in der selben Aufnahme zu sehen. Man erkennt bei der vorderen Welle, wenn überhaupt, nur marginal ausgeprägt die gesuchte Struktur, bei der hinteren allerdings sehr deutlich. Aus b) lässt sich erkennen, dass die mit I markierte Wellenfront im Höhenkanal aber eine deutlich stärkere Kante aufweist als die zweite, wodurch das Wellenfront-Feature dem Fall I eine hohe Bedeutung beimessen wird. Die mit II markierte Welle hingegen wird in ihrer Bedeutung nicht erkannt, wie aus nachfolgender Abbildung 4.21 zu sehen.

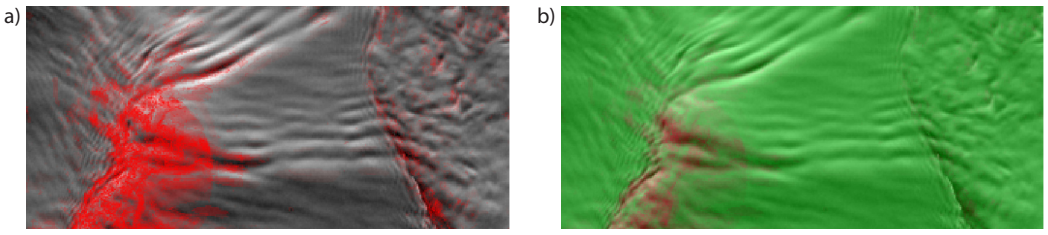

Abbildung 4.21: a) Unsicherheit des *Random-Forest* bei Klassifikation mit Feature zur Wellenfrontanalyse, b) Vorhersage

Die Thresholds des Canny-Edge-Filters so sensibel einzustellen, dass diese auch in Fällen wie oben gezeigt anschlagen, führt allerdings wieder zu einer erhöhten Fehldetektion an unrelevanten Stellen, da dann unkontrolliert Kanten in den Bildern gefunden werden.

Canny-Detektor auf Neigungskanälen

Ein weiteres Problem dieser Herangehensweise ist, dass Ilastik (siehe Kapitel 2.3) die gewählten Features auf allen vorhandenen Kanälen anwendet und dem *Random-Forest* (Kapitel 2.1.4) als Trainingsset übergibt. Der Wellenfrontdetektor ist aber eigentlich nur auf dem Höhenkanal sinnvoll, da er auf den Neigungskanälen eher unkontrolliert Kanten findet. In späteren Versionen von Ilastik wird aber eine Auswahl, welche Features auf welche Kanäle wirken sollen, möglich sein.

Fehler

Um ein Maß für die Güte dieses Verfahrens zu finden, wurden 40 Bilder (20 positive und 20 negative Beispiele für *Microscale-Breaking*) zufällig ausgewählt und mit dem Trainierten *Random-Forest* (siehe Abschnitt 2.1.4) segmentiert. Im Anschluss

wurde das Ergebnis pixelweise mit einer Segmentierung verglichen, die nach menschlicher Einschätzung von Hand erfolgte. Vier Beispiele aus dieser Auswertung sind in nachfolgender Abbildung zu sehen.

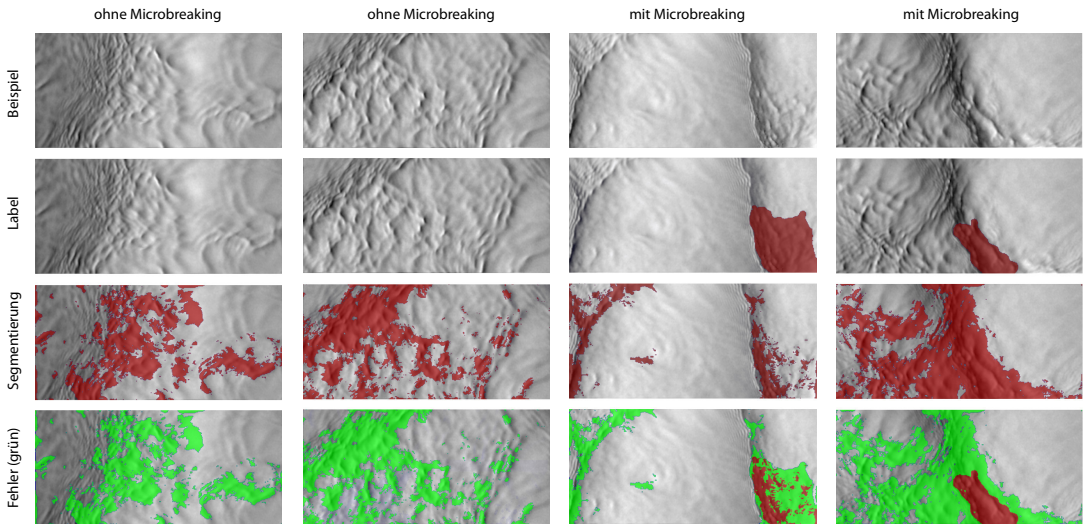

Abbildung 4.22: Fehler in der Segmentierung

In Abbildung 4.22 sind vier der Beispiele dargestellt, die zur Fehlerauswertung verwendet wurden. Die beiden Spalten links enthalten kein zu erkennendes Ereignis, die rechten beiden weisen *Microscale-Breaking* auf. In der zweiten Reihe sind jeweils die von Hand gesetzten Segmentierungen abgebildet, in der dritten die vom *Random-Forest* errechneten und in der letzten Reihe der Vergleich von beiden; wobei jedes in Reihe drei, aber nicht in Reihe zwei enthaltene Pixel auf Grün gesetzt und damit als Fehler markiert wurde.
Daraus wurden zur besseren Vergleichbarkeit mit nachfolgender Methode, zusätzlich zum *Out-Of-Bag-Error* F_{oob} des *Random-Forest* (siehe 2.1.4), zwei Fehler bestimmt. Zum Einen die fehlerhafte Vorhersage des *Random-Forest* in Pixeln ohne Ereignis, also ein falsch positiv Fehler F_+ und zum Anderen in Bildern mit *Microscale Breaking*, die prozentualen Abweichungen F_- von den „idealen“, von Hand gesetzten Segmentierungen:

$$F_{oob} = 11.2\%$$
$$F_+ = 32.76\%$$
$$F_- = 19.72\%$$

Fazit

Als Fazit muss bei dieser Art der Detektion von *Microscale-Breaking-Waves* gesagt werden, dass die Methode nicht ausreichend ist um gesicherte Aussagen über die Häufigkeit dieser Ereignisse innerhalb eines Datensatzes treffen zu können. Sehr ausgeprägte Ereignisse werden zwar erkannt, aber die Fehldetektionen sind doch sehr hoch. Mangelnder Arbeitsspeicher auf einem normalen Home-PC führte dazu, dass maximal auf 20 Beispielen trainiert werden konnte, jedoch auch Versuche auf einem High-End-Gerät mit 24GB RAM und der vierfachen Menge an Trainingsdaten brachte keine deutliche Verbesserung der Ergebnisse.
Was bei den sehr charakteristischen Ereignissen der Regentropfen-Segmentierung (siehe Kapitel 4.2) keine Rolle spielte, wird bei den Wellen zum Problem, die fehlende Zeitinformation. Auch das menschliche Auge, respektive das Gehirn, benötigt als wichtigstes Erkennungsmerkmal für die Vorgänge in einem Wellenfeld die Zeitkomponente.

4.3.3 Segmentierung mit Bildsequenzen

In den neueren Versionen von Ilastik wird auch die Segmentierung in Volumen unterstützt. Durch diese dritte Dimension ist es in den Wellendaten möglich den Zeitkontext herzustellen und somit einige der Probleme in der Segmentierung auf Einzelbilder (Siehe Abschnitt 4.3.2) zu umgehen. Allerdings wird hierfür, selbst bei der niedrigen Auflösung der Daten (3*149*320), sehr viel Arbeitsspeicher benötig. Für die folgenden Ergebnisse wurde auf einem Rechner mit 48GB Ram und 8 Cores gearbeitet, was auch in vollem Umfang ausgenutzt wurde.

Volumensegmentierung

Die Vorgehensweise bei der Arbeit mit Volumendaten[1] in Ilastik ist nahezu die Selbe wie bisher, außer dass ein Trainingssatz mit n Trainingsbeispielen nun statt aus $3*n$ Einzelbildern aus $3*n$ Einzelvolumen zusammenhängender Abschnitte der Datensätze besteht. Hierfür wurden pro Volumen 40 aufeinander folgende Bilder verwendet. Dies entspricht einer Zeitdauer von $0.128\,s$ je Volumen. Trainiert wurde auf 30 solcher Sequenzen, wobei zur Vermeidung von Randproblemen immer nur maximal in den drei mittleren Bildern des Volumens gelabelt wurde. Ein Beispiel ist in Abbildung 4.23 zu finden. Man erkennt die vier Darstellungsperspektiven. Rechts unten eine 3D-Ansicht und rechts oben die gewohnte Draufsicht auf die xy-Ebene. Links sind die Zeitachsen zu sehen, wobei oben der Schnitt parallel zur y-Richtung (in 3D Übersicht waagerechter Ausschnitt) und links unten parallel zur x-Richtung (in 3D-Ansicht senkrecht) dargestellt ist.

[1] Im Kontext dieser Arbeit sollte der Begriff Volumendaten allerdings nicht wörtlich genommen werden, da es sich hier bei der dritten Dimension um die Zeit handelt.

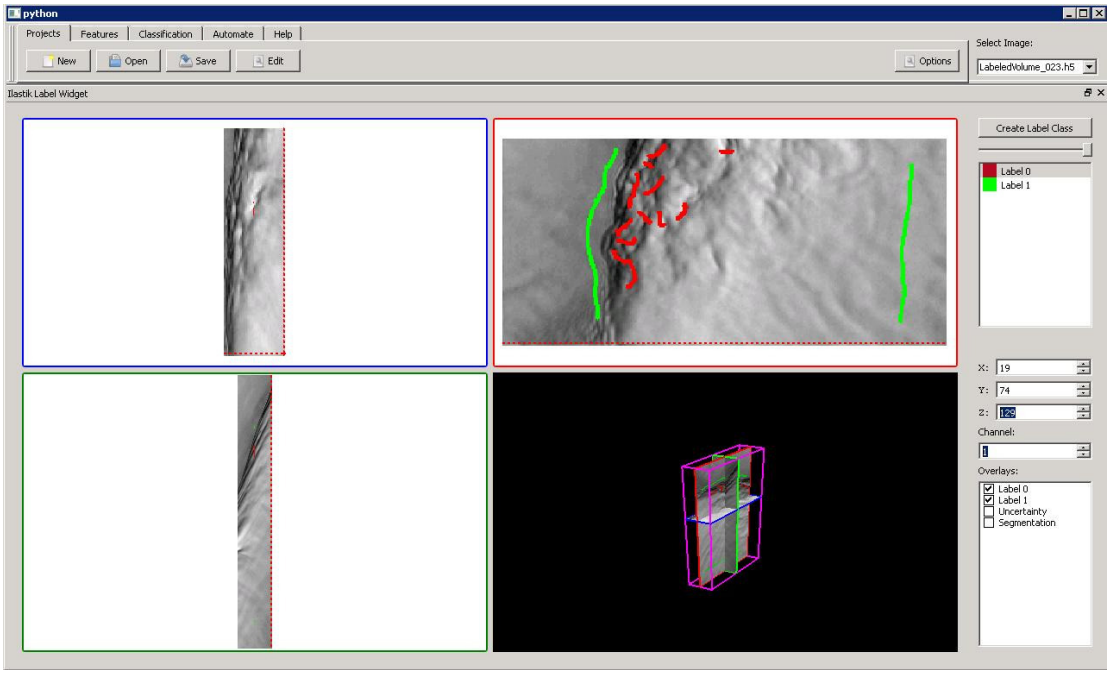

Abbildung 4.23: 3D Labeling in Ilastik

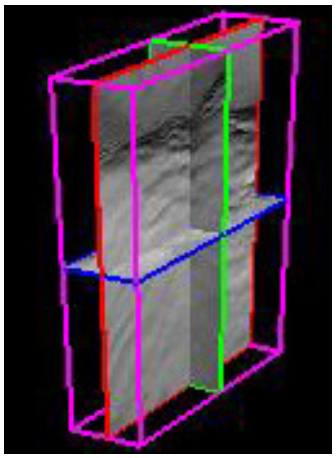

Abbildung 4.24: 3D Ansicht in Ilastik.

Der sonstige Ablauf ist analog zu allem bisher beschriebenen. Es wurden wieder die Features *Color* und *Texture* in *Large* und *Huge* gewählt und der *Random-Forest* auf den 30 gelabelten Volumen trainiert. Das Feature zur Wellenfrontdetektion hat im dreidimensionalen Fall keine Bedeutung mehr. Abbildung 4.25 zeigt eine Vorhersage aus dem Trainingsatz, wobei die Bilder b),c) und d) zu diesem Zeitpunkt noch ungelabelt waren und zur Kontrolle der Vorhersagekraft des *Random-Forest* dienten.

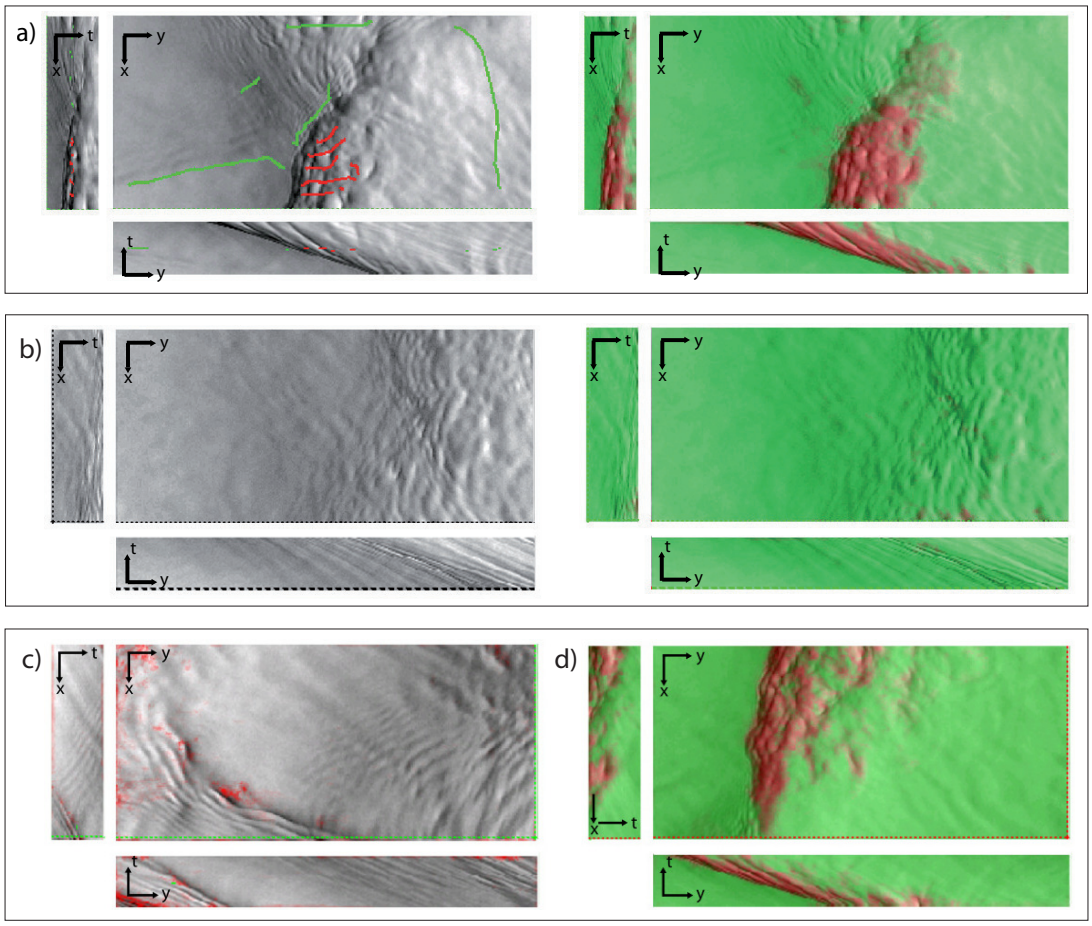

Abbildung 4.25: a) Gelabeltes Beispiel aus Volumen-Trainingsatz. b),c),d) Ungelabelte Beispiele aus diesem Trainingsatz zur Kontrolle des *Random-Forest*. Zu diesem Zeitpunkt war erst die Hälfte der Trainingsbeispiele gelabelt. Man erkennt in c) die geringen Unsicherheiten und in d) die gute Vorhersagefähigkeit.

In Abbildung 4.25 ist außerdem zu sehen, welche Bedeutung die Zeitkomponente spielt, wenn man in den Beispielen die Zeitprojektionen betrachtet. Man erkennt, dass sich auch in der Zeitdimension die interessanten Strukuren bei brechenden (Abbildung 4.25) a) und d) und nicht brechenden Ereignissen b) und c) signifikant unterscheiden, da die orangenhautartige Struktur in der Schleppe einer brechenden Welle genauso in der Zeitrichtung auftaucht. Das heißt, der *Random-Forest* muss nun nicht mehr zwischen sehr ähnlichen Strukuren auf einer Fläche entscheiden, sondern er „sieht“ ganze Teilvolumen bestimmter Charakteristiken, was die Inhomogenität der Information in den Daten erhöht. Außerdem zeigt Abbildung 4.25 c) den Vorteil gegenüber der zweidimensionalen Segmentierung. Man erkennt eine deutlich ausgeprägte Wellenfront auf die im 2D Fall der Wellenfrontdetektor reagiert hätte.

Da dieser bei Mitnahme der Zeitinformation nicht nötig ist existiert dieses Fehlerpotential überhaupt nicht, was sich in der dargestellten Unsicherheit des *Random-Forest* bemerkbar macht, die nur marginal auftritt.

Bei der Klassifikation unbekannter Daten müssen Ilastik ebenfalls Volumen übergeben werden. Wichtig hierbei ist, sich die mögliche Größe des Volumens zu überlegen. In diesem Fall standen nutzbare 32GB zur Verfügung, Ilastik bekommt das zu klassifizierende Volumen der Größe $V = 3\,\text{Kanäle} * 149 * 320\,\text{Pixel} * n_V\,\text{Frames}$. Die gewählten Features (*Color* und *Texture*) werden in zwei Größen angewandt (*Large* und *Huge*), wobei das Feature *Color* ein Volumen zurück gibt, das Feature *Texture* fünf. Das macht insgesamt 27 im Arbeitsspeicher verweilende Volumen der Größe V. Die Daten werden in *float* abgelegt, somit nimmt jede Zahl vier Byte in Anspruch:

$$n_V = \frac{32 * 10^9}{3 * 149 * 320 * 27 * 4} \approx 2071$$

Die 32GB könnten also theoretisch 2000 Frames große Volumen in Ilastik verarbeiten. Allerdings muss noch beachtet werden, dass der Trainingsdatensatz zur Berechnung des *Klassifikators* ebenfalls noch im Speicher liegt. Bei neueren Versionen von Ilastik lässt sich nach absolviertem Training der *Random-Forest* exportieren und eigenständig auf zu klassifizierende Daten anwenden, was unter Umständen viel Speicherpotential freigibt. Hier wurde entschieden, die 5000 Frames großen Datensätze in 5 Volumen aufzuteilen und einzeln zu segmentieren. Es wurde aber noch ein Überlapp von 50 Frames an die Volumen angehängt und nach der Segmentierung wieder entfernt um unschöne Randeffekte zu vermeiden.

Fehler

Analog zum zweidimensionalen Fall wurden aus der dreidimensionalen Segmentierung die Fehler F_{oob}, F_+ und F_- ermittelt. Dazu wurden wieder 20 positiv und 20 negativ Beispiele klassifiziert. Die zufällig ausgewählten Sequenzen bestanden aus 40 Frames und waren nicht Bestandteil des Trainingssatzes.

$$F_{oob} = 2.1\%$$
$$F_+ = 7.85\%$$
$$F_- = 4.69\%$$

Nachfolgende Abbildung 4.26 zeigt schon optisch die Verbesserung in der Genauigkeit. Etwa in gleichen Verhältnissen haben sich die drei Fehler im Mittel um einen Faktor 4 verbessert.

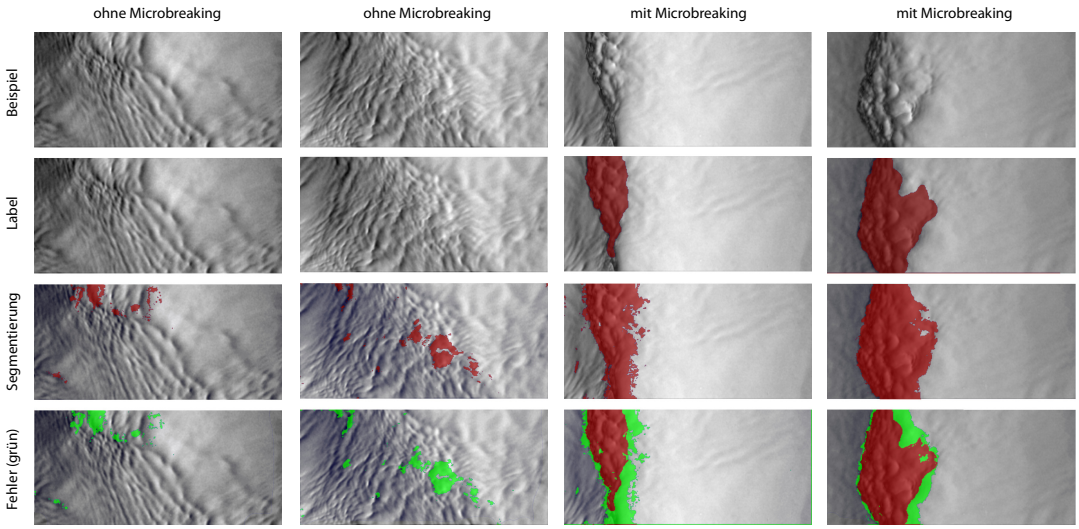

Abbildung 4.26: Fehler in der Segmentierung

Im direkten Vergleich (Abbildung 4.27), dargestellt in WaveVis, ist die deutlich bessere Segmentierung unter Mitnahme der Zeitinformation zu erkennen. Man sieht anhand des letzten Bildvergleichs, dass eindeutige *Microbreaking*-Ereignisse in beiden Varianten recht gut segmentiert werden, dass aber die Fehlinterpretationen des *Random-Forest* in der 3D Variante signifikant reduziert werden können.

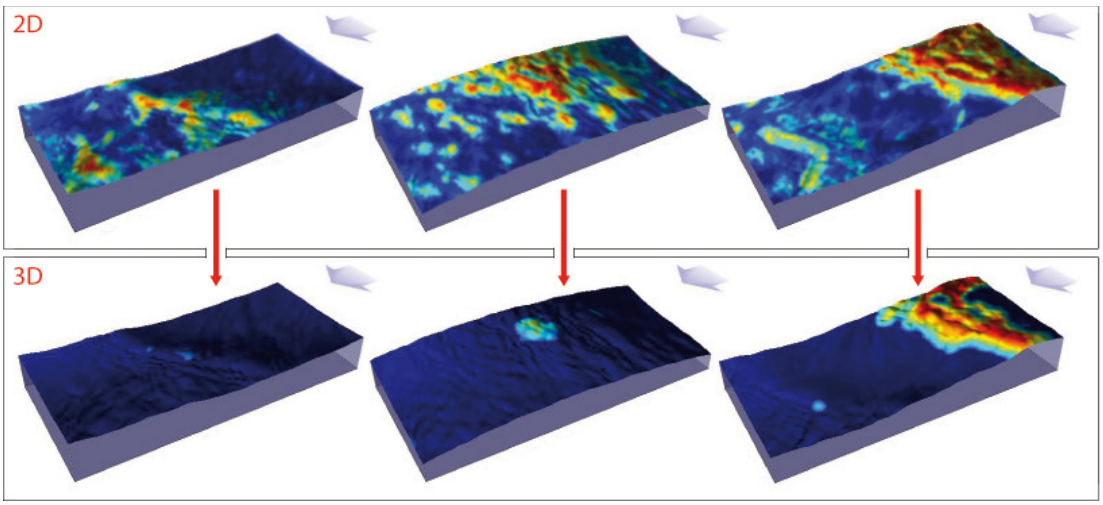

Abbildung 4.27: Vergleich Segmentierung in 2D (oben), in 3D (unten). Die ersten beiden Bildvergleiche weisen kein *Microbreaking* auf. Die Fehldetektionen im den oberen Bildern werden in 3D weitestgehend vermieden.

4.3.4 Ergebnisse

Die Ergebnisse aus der Segmentierung im Volumen, also unter Berücksichtigung der Zeitinformation, zeigen eine sehr sichere Erkennungsrate bei ausgeprägten brechenden Wellen (siehe Abbildung 4.29), wie sie bei Windgeschwindigkeiten ab circa 6 m/s häufig auftreten. Prozentuale Anteile pro Datensatz zeigt Abbildung 4.28.

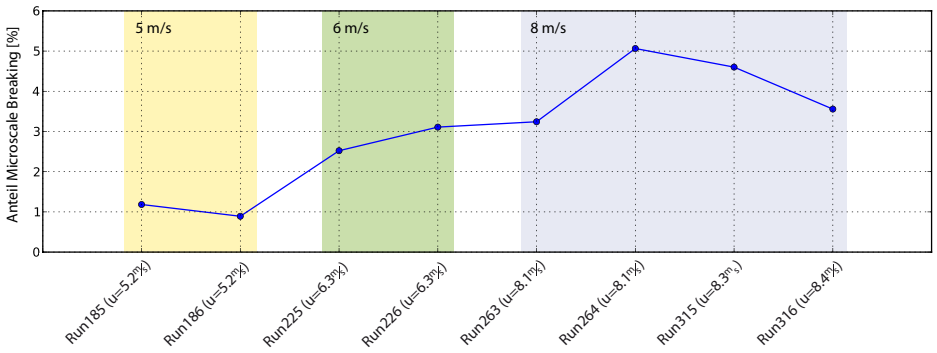

Abbildung 4.28: Ergebnisse der Segmentierung verschiedener Datensätze unterschiedlicher Windbedingungen, angegeben in prozentualem Flächenanteil.

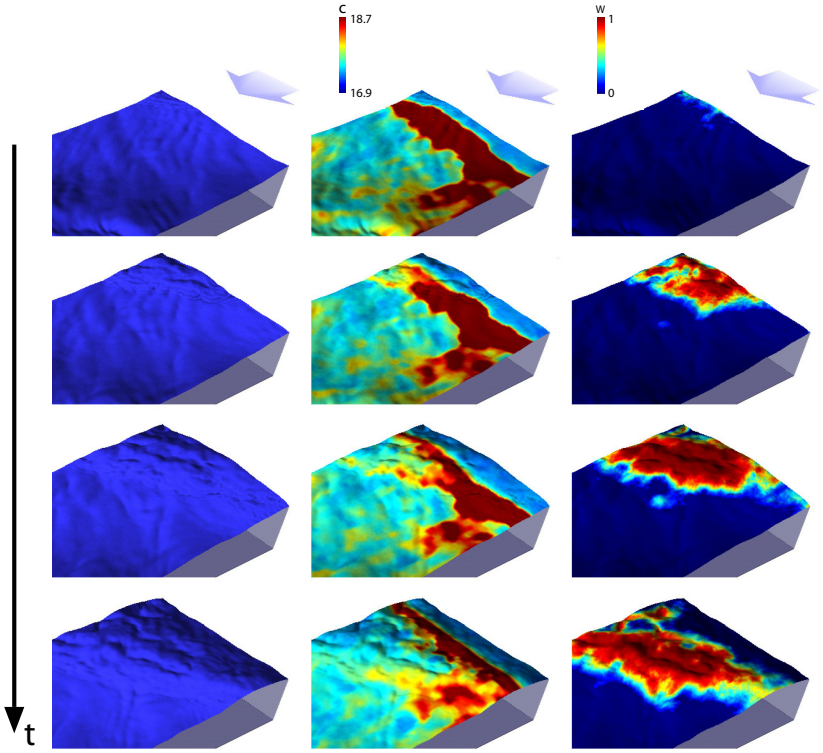

Abbildung 4.29: Ins Bild laufende brechende Welle. Zum Vergleich sind zusätzlich IR-Overlay und die Segmentierung dargestellt.

Die Fehldetektionen konnten bei der Segmentierung auf Bildsequenzen stark reduziert werden. Es muss allerdings beachtet werden, dass eine durch erhöhte Verwirbelung verursachte charakteristische Struktur in den Neigungskanälen detektiert wird, nicht die Turbulenzen selbst. Die durch diese Turbulenzen verursachten Prozesse der Oberflächenerneuerung können auf diese Art nicht detektiert werden. Deshalb werden auch Wellen im Grenzbereich[1] oft nur unzureichend erkannt. Hierfür müssen die in Kapitel 2.5.2 erläuterten Konzepte verwendet werden. Ein kurzer Vergleich der hier vorgestellten Ergebnisse der Detektion von *Microscale-Breaking-Waves* mit den Methoden der Thermographie soll im folgenden angestellt werden.

Vergleich Oberflächenerneuerung aus IR-Daten mit Microbreaking aus Segmentierung

Wie in Kapitel 2.5.2 beschrieben, wurden während der WiSSCy Messkampagne verschiedene Laser-Modi eingesetzt, um Wärme als Tracer auf die Wasseroberfläche aufzubringen. Zur optischen Identifizierung mittels WaveVis ist der Linien-Modus sehr gut geeignet. Möchte man aber Turbulenzen detektieren, die kühles Tiefenwasser zur Oberfläche transportieren, eignet sich der Scan-Modus (siehe Kapitel 2.5.2) allerdings besser. Geht man vom Ideal aus, dass mit dem Lasersheet so schnell über die Wasseroberfläche gescannt wird, dass eine Oberflächenschicht konstanter Temperatur entsteht, können regionale Abweichungen dieser Temperatur als Oberflächenerneuerung angesehen werden.

Im Experiment wurde der Aufbau aus Abbildung 2.23 verwendet. Der CO_2-Laser scannte mit einer Frequenz von 19.53 Hz über den abgebildeten Bereich und erhöhte dadurch die Wassertemperatur bis zu einer Tiefe von circa $z_0 = 11\,\mu\text{m}$[1]. Die aufgebrachte Energie benötigte im Mittel sieben Sekunden zur vollständigen Abstrahlung, was sehr viel länger ist als die hier betrachteten Zeitskalen [30]. Da in den IR-Daten das Lasersheet aufgrund der hohen Aufnahmefrequenz (312 Hz) des CISG (siehe Abschnitt 2.5.1) deutlich sichtbar ist, wurden die Sequenzen zur Vorbereitung zuerst mit einem Box-Filter in der Zeit geglättet. Danach Mittelwerte T_{mean} und Standardabweichung σ_T der vollständigen Datensätze (5000 Frames) Pixelweise ermittelt. Ein Oberflächenerneuerungsereignis liegt vor, wenn ein Threshold T_{Th} unterschritten wird.

$$T_{Th} = T_{mean} - \gamma * \sigma_T \tag{4.3}$$

[1] Wellen, die Wärmeumwälzung verursachen und daher als *Microscale-Breaking-Wave* bezeichnet werden können, die typische Orangenhautstruktur aber nur schwach bis noch gar nicht aufweisen.

[1] $T(z) = T_0 exp(-z/z_0) \quad | \; z = z_0 \rightarrow 1/e * T_0 \approx 37\%$

Gilt also $T < T_{Th}$, wird das Pixel als Ereignis gezählt und durch Summation der Flächenanteil A_n über den gesamten Datensatz ermittelt. Bezeichnet $g(t)$ das Bild zum Zeitpunkt $t \in \mathbb{N}$ und $g_{ij}(t)$ das Pixel ij zum Zeitschritt t, so wird A_n wie folgt ermittelt.

$$A_n = A^{-1} \cdot \sum_{t=0}^{5000} \sum_{g_{ij} < T_{Th}} 1 \qquad \text{mit} \quad A = \sum_{t=0}^{5000} \sum_{ij} 1 \tag{4.4}$$

Ähnliche Herangehensweisen finden sich in Zappa et al. [39, 40] und Loewen et al. [25]. Hier wurden ebenfalls mit aktiver Thermographie Oberflächenerneuerungsereignisse detektiert und Flächenanteile bei verschiedenen Windgeschwindigkeiten bestimmt. Allerdings ist die Aussagekraft der hier verwendeten Infrarotdaten schwer zu beurteilen. In Abbildung 4.30 werden die Standardabweichungen der verwendeten IR-Daten aller betrachteten Datensätze dargestellt.

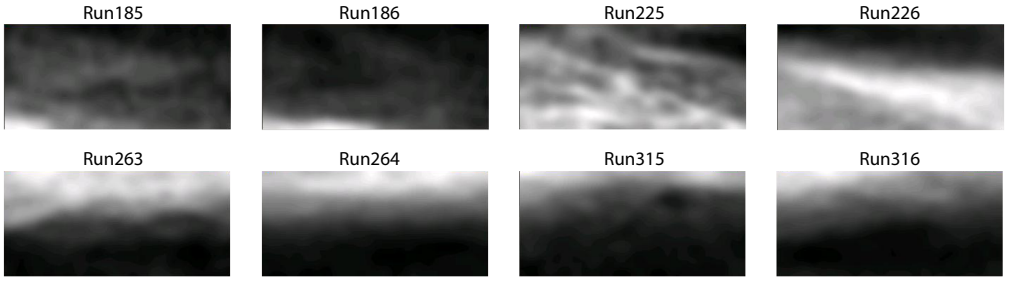

Abbildung 4.30: Standardabweichungen der verwendeten IR Datensätze. Die Inhomogenität beruht auf der kurzen Mittellungszeit (circa 16s) und der gaußförmigen Struktur des Lasersheets in senkrechter Richtung. Zudem lässt sich ein Wandern des Laserschwerpunktes im Verlauf der Kampagne erkennen.

Die Inhomogenität der Standardabweichungen aus Abbildung 4.30 hat im Wesentlichen zwei Ursachen. Der aufgefächerte Laser (siehe Abbildung 2.23) wies eine gaußförmige Intensität senkrecht zur Scanrichtung auf, deren Schwerpunkt aber nicht immer zentral über den Bildbereich lief. Zusätzlich verschob sich der Schwerpunkt im Laufe der Kampagne, was aus Abbildung 4.30 ersichtlich ist. Dadurch waren weder im einzelnen Datensatz noch im Kampagnenverlauf konstante Bedingungen gegeben. Zum Anderen wurde die Statistik für jede Auswertung der Oberflächenerneuerung nur auf dem jeweiligen Datensatz gemacht. Da jeder Datensatz aus 5000 Bildern besteht, was nur 16s Aufnahme entspricht, ist die statistische Aussagekraft der Mittelwerte und Standardabweichungen in Frage zu stellen.

In Abbildung 4.31 sind die nach Gleichung 4.3 und 4.4 ermittelten Oberflächenerneuerungsanteile der verschiedenen Datensätze in Abhängigkeit von γ dargestellt.

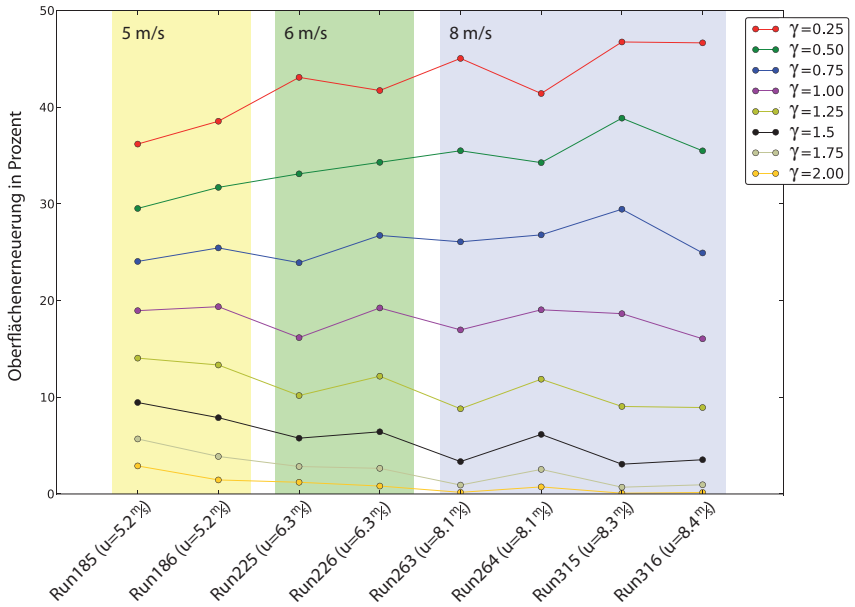

Abbildung 4.31: Vergleich Oberflächenerneuerung bei verschiedenen γ und Windgeschwindigkeiten zwischen 5 und 8 m/s.

Das die Anteile der Oberflächenerneuerung in Abbildung 4.31 mit steigender Windgeschwindigkeit bei hohen Werten von γ (ab $\gamma = 1.0$) abnehmen, lässt sich mit den Standardabweichungen der verwendeten IR-Sequenzen erklären (Abbildung 4.32).

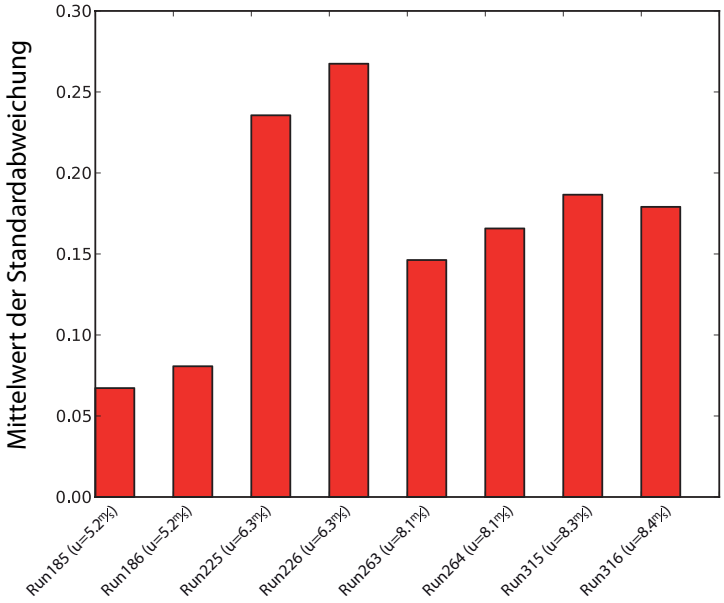

Abbildung 4.32: Mittelwerte der Standardabweichungen aller verwendeter Datensätze der Messkampagne aus Abbildung 4.30.

Da diese mit zunehmender Windgeschwindigkeit größer werden, wird auch der Beitrag $\gamma \cdot \sigma_T$ aus Gleichung 4.3 größer als bei niedrigen Windgeschwindigkeiten trotz gleichem γ. Dies führt dazu, dass die Anzahl der gezählten Pixel unterhalb des *Thresholds* bei höheren Windgeschwindigkeiten kleiner werden kann als bei niedrigen (Vergleiche Abbildung 4.33).

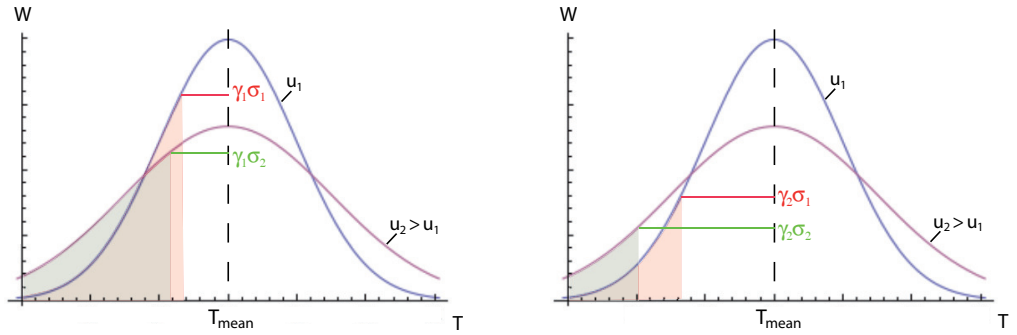

Abbildung 4.33: Illustration gaußförmig verteilter Standardabweichungen bei unterschiedlichen Windgeschwindigkeiten (niedrige Standardabweichung schmaler Gauß, höhere breiter Gauß). Rechte Abbildung veranschaulicht das bei gleichem γ aber unterschiedlichem σ_T die registrierten Pixelanzahlen trotz höherer Windgeschwindigkeit kleiner ausfallen können.

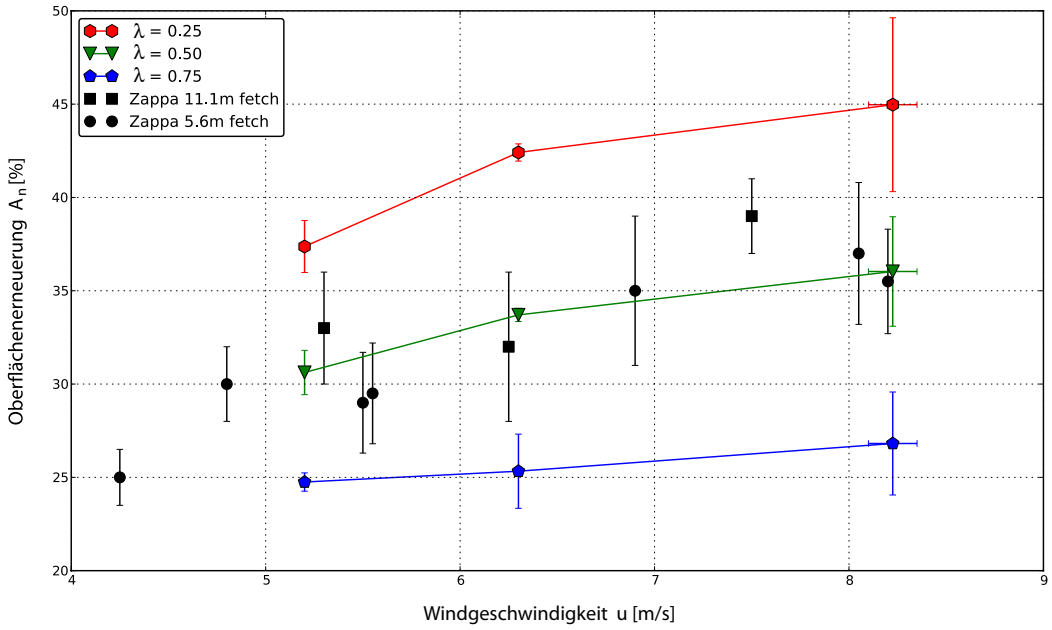

Abbildung 4.34: Vergleich des Oberflächenerneuerungsanteils aus den WiSSCy-IR Daten mit früheren Ergebnissen aus Zappa et al. [39]. In den Fehlern der WiSSCy-Datenpunkten sind keine systematischen Fehler, sondern nur die Abweichungen aus der Mittelung aller verwendeten Datensätze (Siehe Anhang A) dargestellt.

Abbildung 4.34 zeigt, dass für einen Threshold mit $\sigma = 0.5$ die Ergebnisse aus den WiSSCy-IR Daten recht gut zu den Daten von Zappa et al. [39] passen. Hierfür wurden die in Abbildung 4.31 dargestellten Oberflächenerneuerungsanteile aus der Segmentierung nach Windgeschwindigkeiten gemittelt und für $\sigma = [0.25, 0.5, 0.75]$ in Bezug zu den Ergebnissen aus Zappa et al. [39] dargestellt. Die IR-Daten der WiSSCy-Messkampagne wurden bei einem *fetch*[1] von 15 m aufgenommen.

Zum Vergleich von Oberflächenerneuerung und Segmentierung wird in folgender Abbildung 4.35 die Korrelation beider Ergebnisse dargestellt.

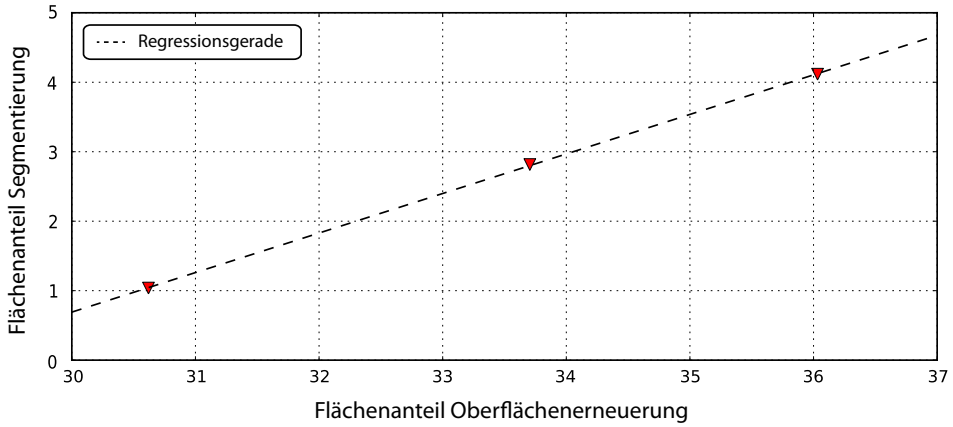

Abbildung 4.35: Korrelation zwischen Oberflächenerneuerung und *Microbreaking*-Segmentierung. Für die Datenpunkte der Segmentierung wurden, analog zu den Werten der Oberflächenerneuerung, die bestimmten Werte aus Abbildung 4.28 beziehungsweise Anhang A den Windgeschwindigkeiten nach gemittelt.

Man erkennt, dass beide Resultate, bis auf den Offset, der etwa 30% Flächenanteil entspricht, gut korrelieren. Der Grund für den Offsets ist dass hier zwei verschiedene Vorgänge detektiert werden. Im Fall der Oberflächenerneuerung werden durch Turbulenzen verursachte Temperaturumwälzungen registriert, deren Ursachen nicht ausschließlich in *Microscale-Breaking-Waves* zu suchen sind (Vergleiche Abbildung 4.37). Die Segmentierung dagegen lässt nur Aussagen über das Vorhandensein von Strukturen auf der Wasseroberfläche zu die durch brechenden Wellen verursacht werden. Vergleicht man beide Vorgänge miteinander stellt man fest, dass die mittels Segmentierung gefundenen Flächen im Allgemeinen Teilmengen der Gebiete der Oberflächenerneuerung sind, das letztere aber auch unabhängig von den brechenden Wellen auftreten (Vergleiche nachfolgende Abbildungen 4.36, 4.37).
Abbildung 4.36 zeigt einen direkten Vergleich zweier ins Bild laufender *Microbreaking-Waves* mit der dort registrierten Oberflächenerneuerung. Es lassen sich neben den Übereinstimmungen auch vor der Wellenfront Bereiche erkennen in denen der *Threshold* unterschritten wird.

[1] Distanz über die der Wind auf die Wasseroberfläche wirken kann.

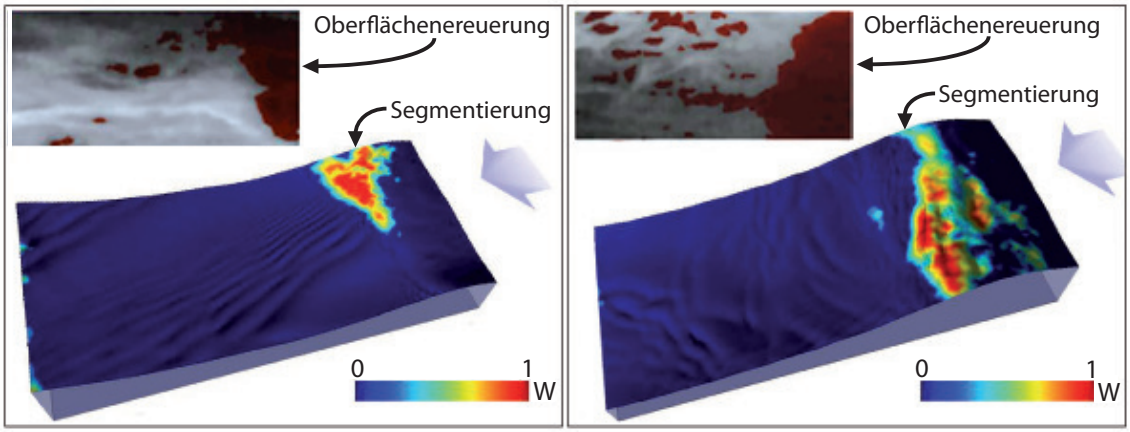

Abbildung 4.36: Vergleich Segmentierung von Microbreaking (Wahrscheinlichkeit dargestellt in WaveVis) und Oberflächenerneuerung (jeweils oben links). Die roten Regionen in den Teilbildern oben links sind dabei die, über einen Threshold von $\gamma = 0.5$, detektierten Regionen der Oberflächenerneuerung.

Noch deutlicher zeigt dies Abbildung 4.37. Hier wird ein Ausschnitt einer Bildsequenz gewählt in dem kein *Microscale-Breaking* zu sehen ist, was an der Segmentierung deutlich wird. Trotzdem wird in den IR-Sequenzen auf einer sehr großen Fläche der *Threshold* unterschritten.

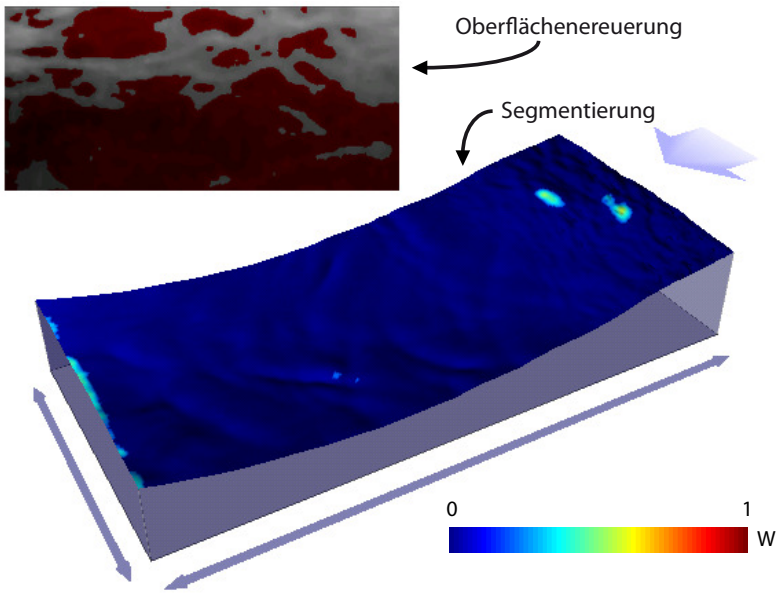

Abbildung 4.37: Vergleich Segmentierung, dargestellt in WaveVis und Oberflächenerneuerung (oben links) zu einem Zeitpunkt ohne brechende Welle. Man erkennt ausgeprägte Oberflächenerneuerung über einen Großteil des Aufnahmebereichs, allerdings keinen Hinweis auf *Microbreaking Waves* durch die Segmentierung.

Hieraus wird klar, dass Temperaturumwälzung verursachende Vorgänge nicht direkt mit dem Auftreten der optisch identifizierbaren brechenden Wellen in Zusammenhang gebracht werden können. Quantitative Aussagen über die Flächenanteile oberflächenerneuernder Wellen innerhalb eines Datensatzes lassen sich mit der Technik der Segmentierung also nicht treffen. Allerdings können über die Klassifizierung auf Bildsequenzen, die deutlich sichtbaren *Miscroscale-Breaking Waves* ohne die Verwendung von Infrarotdaten recht sicher identifiziert und von anderen Wellen unterschieden werden.

5 Zusammenfassung und Ausblick

Zusammenfassung

Es wurde zur Visualisierung großer Datensätze aus Wind-Wellen-Kanälen die Software WaveVis (siehe Kapitel 3) zum interaktiven Rendern weiterentwickelt. Die Software ist in der Lage, aus Neigungs- und Höheninformationen die Wasseroberfläche im Zeitverlauf dreidimensional darzustellen und ermöglicht das *Mapping* skalarer und vektorieller Daten als *Overlays* auf die Wasseroberfläche. Die Benutzerfreundlichkeit wurde wesentlich ausgebaut. Colormaps können zur Laufzeit angepasst, Lineale und Zusatzfunktionen eingeblendet, Lichtverhältnisse verändert und Screenshots gemacht werden. Es wurde eine Schnittstelle zur interaktiven Segmentierungssoftware Ilastik (Kapitel 2.3) geschaffen, über die direkt in der Visualisierung Bereiche gelabelt werden können. Zusätzlich wurde die Visualisierung skalarer und vektorieller Volumendaten implementiert, so dass 3D-Daten in Schnittflächen des Wasserkörpers darstellbar sind. Auf der Basis von Daten einer Messkampagne zur Messung von Gas- und Wärmeaustauschraten unter Wind- und Regenbedingungen wurden Methoden entwickelt, Ereignisse in diesen Daten automatisch zu detektieren. Dafür wurde mittels Techniken des maschinellen Lernens ein *Random-Forest* in Ilastik darauf trainiert, Einschläge von Regentropfen sowie Charakteristiken von *Microscale-Breaking-Waves* zu erkennen. Dabei zeigte sich das Einschläge von Regentropfen auf die Wasseroberfläche sehr gut segmentierbar sind. Im Falle brechender Wellen, mussten bei der Klassifizierung auf Einzelbildern zusätzliche Feature gefunden werden um den Kontext der Wellenfronten einzubinden. Gute Ergebnisse konnten aber erst unter Einbeziehung der Zeitinformation durch das Training auf Bildsequenzen erreicht werden (siehe Kapitel 4). Die Ergebnisse fanden Anwendung in der Analyse von Datensätzen der Messkampagne um detailierte Informationen über die Regenraten zu erhalten, sowie um die Flächenanteile brechender Wellen bei unterschiedlichen Windbedingungen zu bestimmen. Ein Vergleich mit anderen Detektionsmethoden durch aktive Thermografie (Kapitel 2.5.2) wurde angestellt.

Ausblick

Im Weiteren soll die Visualisierungssoftware WaveVis ausgebaut werden. Dabei wäre eine flexiblere Datenverwaltung in Hinblick auf die Fähigkeit Daten in unterschiedlichen Koordinatensystemen und verschiedener Datentypen darstellen zu können eine wichtige Erweiterung. Die Darstellung von Volumendatensätzen sollte verallgemeinert werden, um den Zugang zur Definition der zugrundeliegenden Koordinaten der Datenpunkte frei wählen zu können. Dadurch könnte erreicht werden beliebige Daten aus Experimenten an Windkanälen zu visualisieren. Außerdem soll WaveVis auch zur Visualisierung simulierter Daten [38] optimiert werden.

6 Literaturverzeichnis

[1] E. Alpaydin: *Maschinelles Lernen*. Oldenburg Verlag, 2008.

[2] M. Atmane, W.E. Asher, A. T. Jessup, : *On the use of the active infrared technique to infer heat and gas transfer velocities at the air-water free surface*. J. Geophys. Res., 109, C08S12, 2004.

[3] G. Balschbach: *Untersuchungen statistischer und geometrischer Eigenschaften von Windwellen und ihrer Wechselwirkung mit der wasserseitigen Grenzschicht*. PhD thesis, Institut für Umweltphysik, Universität Heidelberg, February 2000.

[4] M. L. Banner, O. M. Phillips: *On the incipient breaking of small scale waves*. J. Fluid Mech., 65, 647-656, 1974.

[5] C. M. Bishop: *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] L. Breiman, J.H. Freidmen, R.A. Olsen, C.J. Stone: *Classification and Regression Trees*. Republished by CRC Press, 1984.

[7] L. Breiman: *Random forests*. Machine Learning, 45:5-32, 2001.

[8] L. Breiman: *Bagging Predictors*. Machine Learning, 24, S. 123-140, 1996,

[9] L. Breiman, A. Cutler: *Random Forests*. http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#intro.

[10] K.-A. Chang, P. L.-F. Liu: *Pseudo turbulence in PIV breaking-wave measurements*. Experiments in Fluids 2000, 29, 331-338.

[11] A. M. Fincham, G. R. Spedding: *Low cost, high resolution DPIV for measurement of turbulent fluid flow* Experiments in Fluids, 1997, 23, 449-462.

[12] J. Han, M. Kamber: *Data Mining: Concepts and Techniques, 2nd ed.* The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor Morgan Kaufmann Publishers, March 2006.

[13] T. Hastie, R. Tibshirani, J, Friedman: *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Science+Business Media, 2001.

[14] H. Hausecker, S.Reinelt, B. Jähne: *Heat as a proxy tracer for gas exchange measurements in the field: Principles and technical realization*. Air-Water Gas Transfer, edited by B. Jähne and E. C. Monahan, pp. 405-413, AEON, Hanau, Germany, 1995.

[15] H. Haussecker, U. Schimpf, B. Jahne: *Measurements of the air-sea gas transfer and its mechanisms by active and passive thermography*. In U. Schimpf, editor, Proc. IEEE International Geoscience and Remote Sensing Symposium IGARSS 1998, volume 1, pages 484-486 vol.1, 1998

[16] A. Herzog, F. Friedl, B. Jähne: *Simultaneous high-resolution LIF measurements of dissolved gas concentration fields and measurements of wave slope at a wavy free water surface with wind-induced turbulence*. 15h Int. Symp on Appl. Laser Techniques to Fluid Mechanics, Lisabon, Portugal, July 05–08, 2010.

[17] D. T. Ho, E. Asher, L. F. Bliven, P. Schlosser, and E. L. Gordan: *On mechanisms of rain-induced air-water gas exchange*. Journal of Geophysical Research, 105(C10):24,045-24,057,October 2000.

[18] F. Huhn: *A simple instrument for the measurement of the slope and height distributions of small scale wind-driven water waves*. Master's thesis, Institute for Environmental Physics, University of Heidelberg, 2008.

[19] Bernd Jähne: *Digital Image Processing*. Springer-Verlag, Berlin, 6 edition, 2005.

[20] Bernd Jähne: *On the transfer processes at a free air-water interface*. Habilitation, Institut für Umweltphysik, Fakultät für Physik und Astronomie, Univ. Heidelberg, 1985. IUP D-200.

[21] Bernd Jähne, H. Haussecker: *Air-water gas exchange*. Annual Reviews of Fluid Mechanics, 30:443-468, 1998.

[22] H. Jirka, G. Jirka, C. S. Garbe; R. A. Handler und B. Jähne (ed.): *Turbulent gas flux measurements near the air-water interface in a grid-stirred tank Transport at the Air Sea Interface*. Springer-Verlag, 2007.

[23] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, A. Y. Wu: *An efficient k-means clustering algorithm: Analysis and implementation*. IEEE Trans. Pattern Analysis and Machine Intelligence. 24, 2002, S. 881-829.

[24] M. J. Kilgard: *The OpenGL® Utility Toolkit (GLUT) Programming Interface API Version 3*. http://www.opengl.org/documentation/specs/glut/spec3/spec3.html.

[25] M. R. Loewen, M. H. Siddiqui: *Detecting Microscale Breaking Waves.* Measurement Science and Technology. (2006), 17, 771-780.

[26] W. K. Melville: *The role of surface-wave breaking in air-sea interaction.* Ann. Rev. Fluid Mech. 28 279-321, 1996.

[27] D. Michie, D. J. Spiegelhalter: *Machine Learning, Neural and Statistical Classification.* Ellis Horwood Series in Artificial Intelligence. E. Horwood Verlag, New York 1994.

[28] A. Nischwitz, M. W. Fischer, P. Haberäcker: *Computergrafik und Bildverarbeitung.* Vieweg und Teubner; 2. Auflage, 2007.

[29] R. Rocholz: *Spatio-Temporal Measurement of Short Wind-Driven Water Waves.* PhD thesis, University of Heidelberg, Heidelberg, Germany, 2008.

[30] R. Rocholz, S. Wanner, U. Schimpf, B. Jähne: *Combined visualization of wind waves and water surface temperature.* 6th Int. Symp. Gas Transfer at Water Surfaces, Kyoto, May 17–21, 2010.

[31] P. M. Saunders: *The temperature at the ocean-air interface.* J. Atmos. Sci., 24(3), 269-273, 1967.

[32] U. Schimpf: *Untersuchung des Gasaustausches und der Mikroturbulenz an der Meeresoberfläche mittels Thermographie.* PhD thesis, University of Heidelberg, Heidelberg, Germany, 2000.

[33] Shih Yu-Shan: *Families of splitting criteria for classification trees.* Statistics and Computing, 1999, Vol 9, pp. 309-315.

[34] D. Shreiner: *OpenGL®Reference Manual: The Official Reference Document to OpenGL®, Version 1.4 4th Edition.* Addison-Wesley, 2004.

[35] D. Shreiner, M. Woo, J. Neider: *OpenGL®Programming Guide: The Official Guide to Learning OpenGL®, Version 2.1 6th Edition.* Addison-Wesley, 2007.

[36] C. Sommer: *Interactive Learning for the Analysis of Biomedical and Industrial Imagery.* Dissertation, HCI Heidelberg, Univerity of Heidelberg, 2010.

[37] R. Timofeev: *Classification and Regression Trees (CART) Theory and Applications.* Master Thesis, CASE - Center of Applied Statistics and Economics, Humbold University Berlin, 2004.

[38] Wu-ting Tsai, Li-Ping Hung: *Three-dimensional modeling of small-scale processes in the upper boundary layer bounded by a dynamic ocean surface.* Journal of Geophyssical Research, Vol. 112, C02019, 2007.

[39] C. J. Zappa, W.E. Asher, A. T. Jessup, J. Klinke, S.R.Long: *Microbreaking and the enhancement of air-water transfer velocity*. J. Geophys. Res., 109, C08S16, 2004.

[40] C. J. Zappa, W.E. Asher, A. T. Jessup: *Microscale wave breaking and air-water gas transfer velocities*. J. Geophys. Res. 106 9385-91, 2001.

[41] C. J. Zappa: *Microscale wave breaking and its effect on air-water gas transfer using infrared imagery*. Ph.D. disertation, Univ. of Wash., Seattle, 1999.

A Anhang

Regenraten Analyse der WiSSCy-Kampagne			
Datensatz	*Datum*	*u [m/s]*	*Regenrate [mm/h]*
Run052	10.05.07	8.0	90.0
Run053	10.05.07	8.0	114.95
Run134	15.06.07	4.0	100.13
Run135	15.06.07	4.0	99.27
Run232	20.06.07	6.1	41.24
Run233	20.06.07	5.9	44.20
Run234	20.06.07	6.0	50.50
Run235	20.06.07	6.0	67.17
Run270	26.06.07	8.1	35.31
Run271	26.06.07	8.1	34.08
Run272	26.06.07	8.0	33.46
Run323	28.06.07	8.3	69.63
Run324	28.06.07	8.4	58.40
Run325	28.06.07	8.2	56.30
Run326	28.06.07	8.4	60.01
Run348	29.06.07	10.0	54.57
Run349	29.06.07	10.3	62.72
Run350	29.06.07	10.1	61.11
Run431	04.07.07	4.1	62.60
Run432	04.07.07	4.0	65.19
Run433	04.07.07	4.0	62.35

Tropfenanzahl pro 200 Frames aller Datensätze

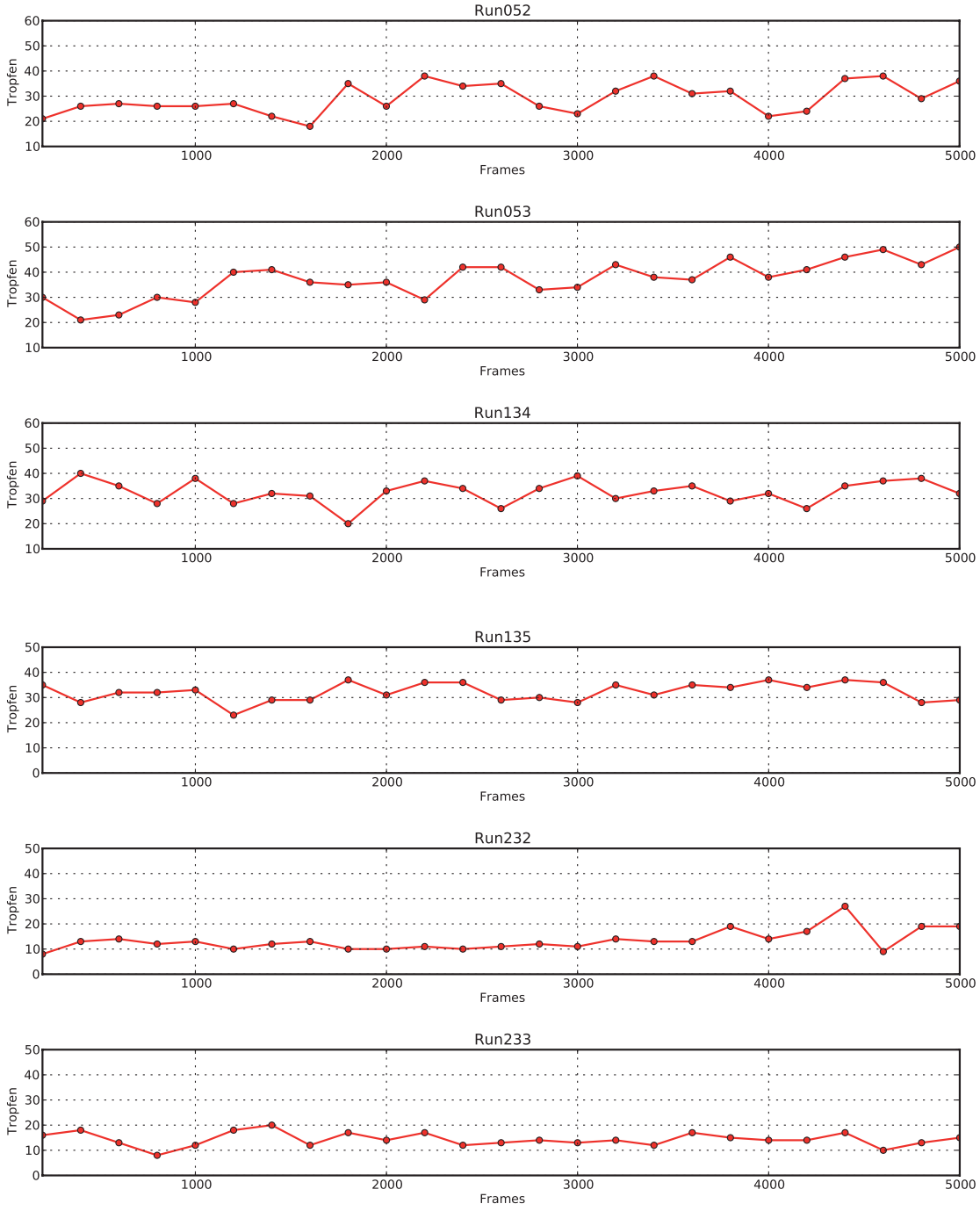

Abbildung A.1:

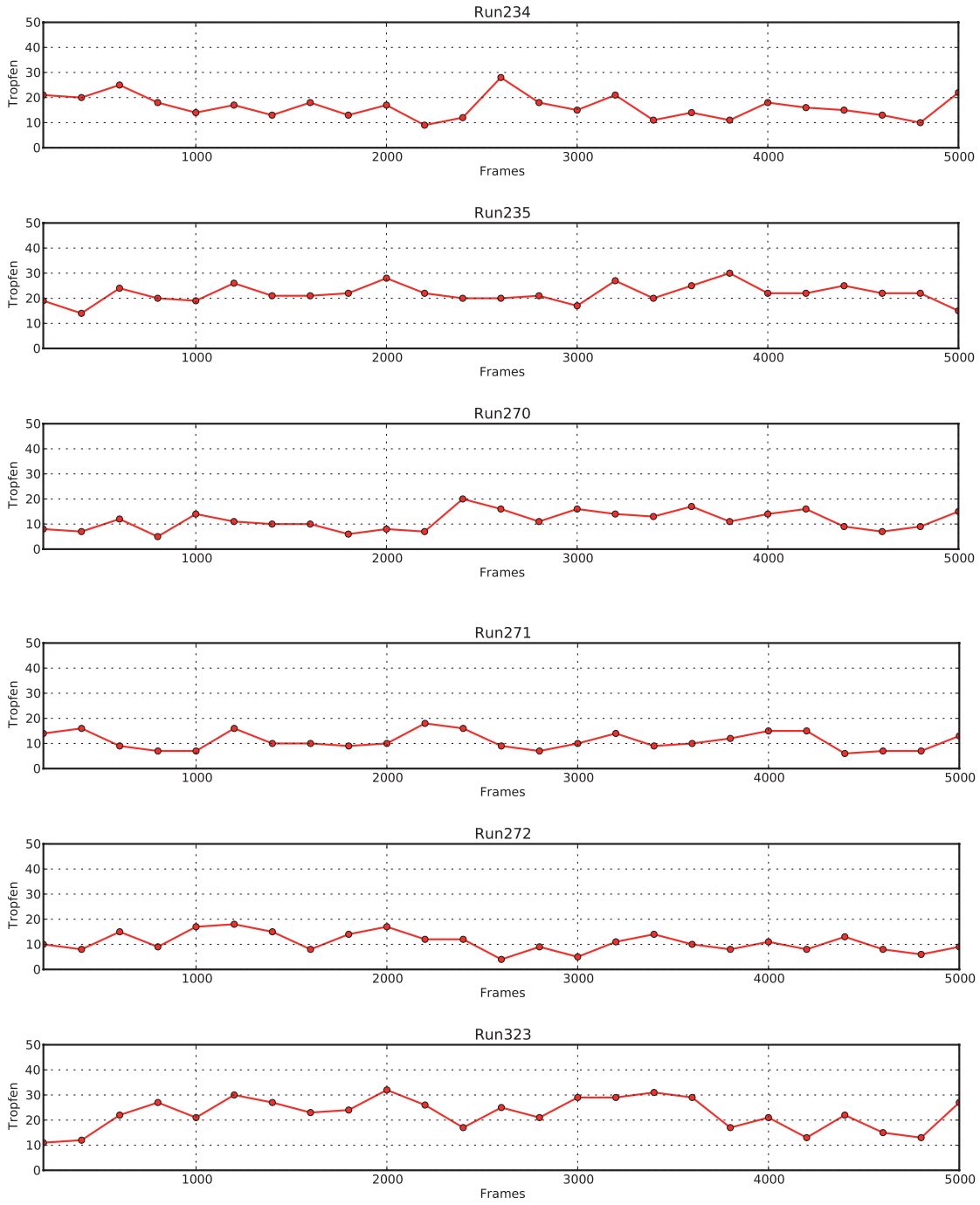

Abbildung A.2:

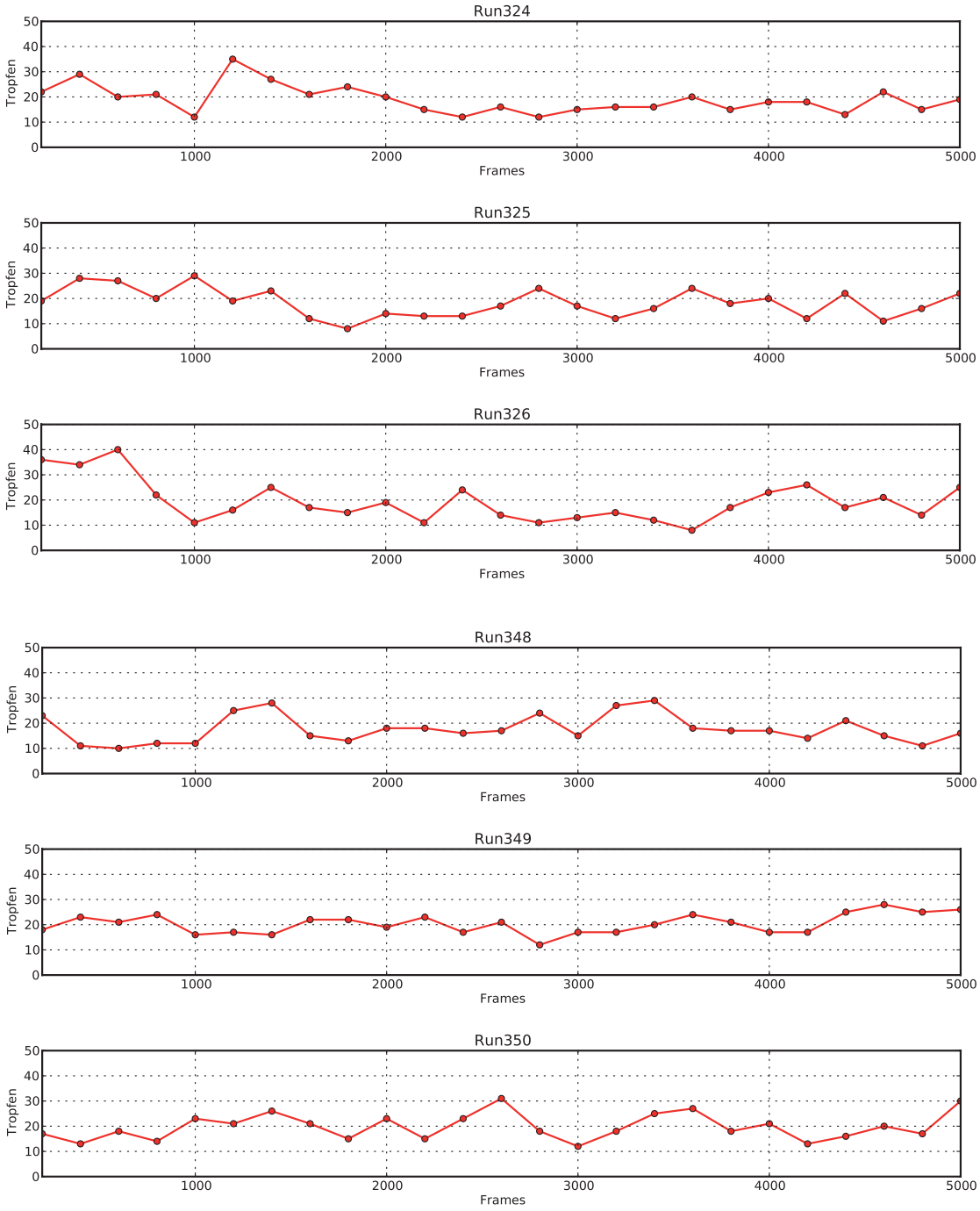

Abbildung A.3:

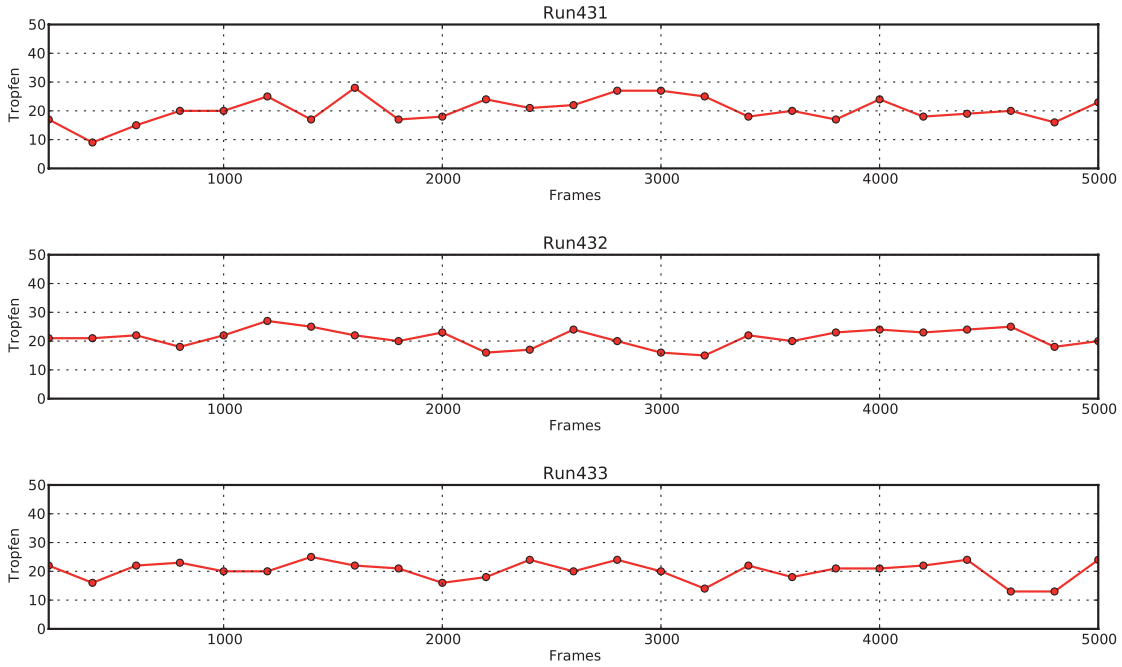

Abbildung A.4:

Segmentierung Microbreaking		
Datensatz	*u [m/s]*	*Anteil*
Run185	5.2	0.01184
Run186	5.2	0.00891
Run225	6.3	0.02522
Run226	6.3	0.03109
Run263	8.1	0.03242
Run264	8.1	0.05064
Run315	8.3	0.04601
Run316	8.4	0.03557

Oberflächenerneuerung				
Datensatz	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1.0$
Run185	0.3619	0.2953	0.2405	0.1896
Run186	0.3855	0.3171	0.2545	0.1937
Run225	0.4309	0.3311	0.2392	0.1617
Run226	0.4173	0.3430	0.2674	0.1924
Run263	0.4505	0.3550	0.2608	0.1697
Run264	0.4142	0.3427	0.2680	0.1905
Run315	0.4675	0.3887	0.2945	0.1865
Run316	0.4666	0.3549	0.2493	0.1605

Oberflächenerneuerung				
Datensatz	$\gamma = 1.25$	$\gamma = 1.5$	$\gamma = 1.75$	$\gamma = 2.0$
Run185	0.1405	0.0946	0.0569	0.0290
Run186	0.1334	0.0789	0.0388	0.0145
Run225	0.1018	0.0577	0.0284	0.0121
Run226	0.1218	0.0643	0.0265	0.0082
Run263	0.0881	0.0335	0.0092	0.0017
Run264	0.1187	0.0616	0.0254	0.0073
Run315	0.0905	0.0308	0.0070	0.0008
Run316	0.0894	0.0354	0.0095	0.0015

Danksagung

Für die Unterstützung durch Rat und Tat während meiner Diplomarbeit möchte ich mich herzlich bei Herrn Professor Bernd Jähne und meinem Betreuer Dr. Roland Rocholz bedanken. Genauso für die schöne Zeit bei der gesamten Arbeitsgruppe am Aeolotron. Ferner möchte ich meinen Dank an das Heidelberg Collaboratory for Imageprocessing und Dr. Christoph Sommer sowie Dr. Ulrich Köthe für Ihre Hilfe und Geduld richten. Besonderen Dank möchte ich auch Christoph Strähle und Wolfgang Mischler aussprechen die mir nicht selten eine große Hilfe waren.

Dank ganz anderer Art gebührt meinen Eltern sowie meiner Oma, ohne Euch hätte ich diese Arbeit nie schreiben können!

Außerdem danke ich Peter Herm, Lars Kaeding und Christoph Mang für ihre Zeit die sie mir zur Korrektur meiner Arbeit schenkten.
Selina, danke für alles!

Erklärung:

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 30. Oktober 2010 .....