Universität für Bodenkultur Wien
University of Natural Ressources and Life Sciences Vienna

Department für Wasser – Atmosphäre – Umwelt
Institut für Wasserwirtschaft, Hydrologie und konstruktiven Wasserbau

Entwicklung einer Software zur automatisierten Objekterkennung in videoüberwachten Fischaufstiegen

Masterarbeit
zur Erlangung des akademischen Grades
Diplomingenieur

Eingereicht von

Frederik Kratzert

Betreuer: Ao. Univ.Prof. Dipl.-Ing. Dr.nat.tech Helmut Mader

September 2016

Danksagung

An dieser Stelle möchte ich mich bei denjenigen bedanken, die mich während der Anfertigung dieser Masterarbeit unterstützt und motiviert haben und mir stets mit Rat zur Seite standen.

Zur erst möchte ich mich bei meinem Betreuer Helmut Mader bedanken, für die stets freundschaftliche Zusammenarbeit, sowie die vielen Freiräume die mir bei der Erstellung dieser Masterarbeit gelassen wurden.

Diese Masterarbeit ist aus einem Foschungsprojekt der VERBUND Hydropower AG entstanden. Ich möchte deshalb insbesondere Sabine Käfer vom VERBUND für die Finanzierung des Projektes, sowie für die gute Zusammenarbeit in dem laufenden und vergangenen Projekten danken.

Auch möchte ich meine Dankbarkeit gegenüber den Kollegen am IWHW ausdrücken. Die stets lockere Arbeitsatmosphäre hat sicherlich dazu beigetragen, dass ich gerne unzählige Stunden am Institut verbracht habe.

Ein ganz besonderer Dank gilt Claire für ihre andauernde Unterstützung und Motivation während der letzten Jahre. Ohne ihre Korrekturarbeit und Verbesserungsvorschläge wäre diese Masterarbeit nicht zu dem geworden was sie ist.
“ *We go together like peas and carrots” - Forest Gump*

Zuletzt möchte ich mich bei meiner Familie, vor allem meiner Mutter, bedanken für die liebevolle Unterstützung auf meinem Lebensweg und die finanzielle Unterstützung während meines Studiums.

Kurzfassung

Eines der großen Anliegen der europäischen Wasserrahmenrichtlinie ist es, das Kontinuum der europäischen Flüsse zu erhalten bzw. wieder herzustellen. Eine gängige Methode zur Wiederherstellung des Kontinuums ist der Bau von Fischaufstiegshilfen (FAH). Im Anschluss an den Bau wird dem Betreiber in der Regel ein Funktionsmonitoring vorgeschrieben. Dabei soll nachgewiesen werden, dass alle im Gewässer vorhandenen Fischarten die gebaute Anlage nutzen. Ein dafür gängiges Verfahren ist das Reusenmonitoring, das jedoch oft einen erheblichen zeitlichen, personellen und finanziellen Aufwand darstellt. Das neu entwickelte FishCam-System ist ein Versuch den Stress für die Fische bzw. die Auswirkungen des Monitorings auf die Fischwanderung zu reduzieren und gleichzeitig Kosten sowie Personalaufwand des Monitorings zu senken. Die Grundidee ist, die Wanderung der Fische in der FAH mit einem Unterwasser-Kamerasystem zu erfassen und die aufgezeichneten Videos mithilfe einer Software automatisch auszuwerten.

Diese Masterarbeit beschäftigt sich mit der Entwicklung der Software, die FishNet getauft wurde. Es werden im Rahmen dieser Masterarbeit die entworfene Programmstruktur von FishNet, sowie alle bisher implementierten Funktionen, vorgestellt. Zu den bereits implementierten Funktionen zählen unter anderem das Erkennen und Verfolgen von Objekten in Videos und das Klassifizieren der Objekte als *Fisch* oder *kein Fisch*.

Das FishCam-System wird, parallel zur Entwicklung von FishNet, bereits in einer Vielzahl von FAH in ganz Österreich für das Monitoring eingesetzt. Alleine die automatische Aussortierung der Videos, die keinen Fisch enthalten, sorgte für eine enorme Reduzierung des manuellen Arbeitsaufwands. Insgesamt wurden bisher über 1,3 Millionen Videos ausgewertet, wovon nur ca. 4.5 % der Videos tatsächlich zumindest einen Fisch enthielten.

Abstract

One of the main objectives of the European Water Framework Directive the preservation and restoration of the continuum of European river networks. Regarding vertebrate migration, fish passes are a measure which is widely used in order to overcome anthropogenic constructions.

In practice, the functionality of almost every newly built fish pass has to be assessed through monitoring. The monitoring aims to prove that the fish pass is used by all relevant fish species in the river system. Traditionally this is done by fish trap monitoring, which involves a lot of time and cost-intensive field work.

The newly developed FishCam-System tries to minimize the cost of monitoring, while at the same time reducing the stress induced on the fish. The main idea is to observe the fish migration using an underwater camera system and automatically analyze the recorded videos with the help of a special software.

This master thesis deals with the development of such a software, named FishNet. It provides an overview of the general structure of FishNet as well as a detailed description of the implemented functions. These include object detection and tracking as well as object classification (*fish* or *no fish*).

While FishNet is still under development, the entire FishCam-System is already in active use for monitoring in a large number of fish passes across Austria. Over 1.3 million videos have been analyzed by FishNet to date, 4.5 % of which contained at least one fish. Thus FishNet, in its current state, has already proven to be able to reduce the manual effort of fish monitoring considerably by sorting out all the videos not containing any fish.

Inhaltsverzeichnis

1 Einleitung

1.1 Projekthintergrund

In der europäischen Wasserrahmen Richtlinie ist festgelegt, dass alle durch Menschen verursachten Wanderhindernisse in Fließgewässern für Organismen wieder passierbar gemacht werden müssen (EU-Kommission, 2000). Eine gängige Methode dafür ist der Bau einer Fischaufstiegshilfen (FAH). Für diese wird in der Praxis häufig ein Funktionsmonitoring vorgeschrieben, wobei untersucht werden soll, ob *„alle relevanten Fischarten und Altersstadien tatsächlich in ausreichender Anzahl in die FAH einwandern und diese auch erfolgreich durchwandern können“* (BMLFUW, 2012). Ein dafür gängiges Verfahren ist das Reusenmonitoring (Woschitz et al., 2003), dass jedoch oft einen erheblichen personellen und finanziellen Aufwand darstellt. Je nach Landesfischereigesetzt muss bis zu zwei Mal am Tag muss Reuse von einem geschulten Fischökologen entleert und gereinigt werden, sowie alle gefangenen Fische bestimmt und vermessen werden. Dies bedeutet zum einen Stress für die Fische und zum anderen scheint es, als werden die Fische durch die Reuse in ihrer freien Wanderung behindert (Mader et al., 2016). Der finanzielle Aufwand kommt gerade bei großen Wasserkraftanlagen zum tragen, denen teilweise durchgehende Monitoringzeiträume von bis zu zwei Jahren vorgeschrieben werden. Aus diesem Grund hat die VERBUND Hydropower AG die Universität für Bodenkultur, BOKU, (Projektleitung Helmut Mader) mit der Entwicklung eines alternativen Monitoringverfahrens beauftragt. Dieses Projekt trägt den Namen FishCam und hat die Entwicklung eines Verfahrens zur Funktionsüberprüfung von FAH zum Ziel, das den Stress für die Fische bzw. die Auswirkungen auf die Fischwanderung reduziert und Kosten sowie Personalaufwand des Monitorings senkt.

Die Idee des Projektes ist ein Video-basiertes Funktionsmonitoring, das aus zwei

Bestandteilen besteht. Zum einen der Hardware, einem, mit einer Videokamera ausgestatteten Erkennungstunnel der problemlos in bestehende FAH eingesetzt werden kann. Zum anderen eine Software, welche die im Erkennungstunnel aufgenommenen Videos analysiert und die Monitoringaufgaben so weit wie möglich automatisiert. Die Entwicklung dieser Software ist das Ziel dieser Masterarbeit.

1.2 Aufbau und Funktionsweise des FishCam-Systems

Die Entwicklung der Videomonitoring-Anlage wurde weitestgehend vor Beginn der Masterarbeit abgeschlossen (Mader and Kratzert, 2016). Das entwickelte System FishCam besteht aus einer Kameraeinheit, einer Reinwasserkammer und einem Erfassungstunnel (siehe Abbildung 1.1).

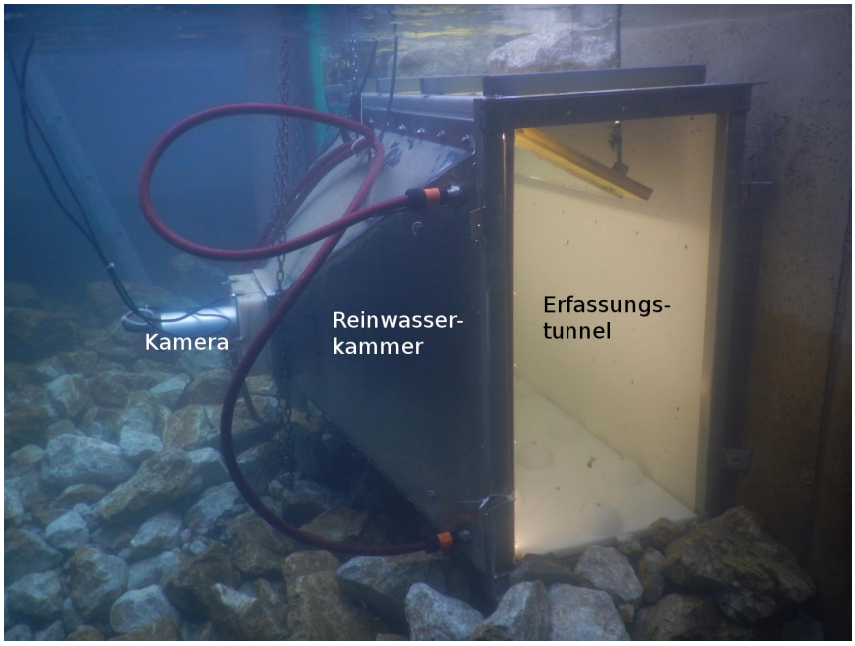

Abbildung 1.1: Unterwasseransicht des FishCam-Systems

Die Kamera ist eine adaptierte Überwachungskamera des Types Axis P1357 mit 5 Megapixel Auflösung. Bei dieser Auflösung können Videos mit bis zu 12 fps aufgezeichnet werden. Die Videospeicherung startet automatisch, sobald sich rund 3 % der Pixel im Kamerasichtfeld verändern. In das gespeicherte Video wird die letzte

Sekunde vor dem Bewegungsbeginn hinzugefügt und die Aufnahme stoppt, nachdem sich für fünf Sekunden nichts mehr maßgeblich im Bild verändert hat. Daraus ergibt sich eine Mindestlänge von 6 Sekunden pro Video. Zusätzlich wurde eine Maximallänge von 5 Minuten festgelegt.Abschnitten

Um die Bildqualität bis zur Rückwandwand des Erfassungstunnels möglichst hoch zu halten, wurde unmittelbar vor die Kamera ein Pyramidenstumpf, der mit Reinwasser gefüllt ist, eingebaut. So kann sichergestellt werden, dass selbst bei trübem Wasser die Sicht bis zur Rückwand unbeeinflusst von der Gewässertrübe ist. Das Sichtfeld der Kamera an der Vorderwand beträgt rd. 1300 x 1000 mm.

Der restliche Aufbau wurde mehrmals leicht verändert mit dem Hintergedanken die automatische Videoauswertung durch bauliche Maßnahmen zu erleichtern. Abbildung 1.2 zeigt die unterschiedlichen Versionen, die in der Praxis zur Anwendung kamen. Variiert wurden die Art und Anordnung der Beleuchtung, Materialart und -farbe und Struktur der Bodenplatten. Des Weiteren wurde ab der zweiten Version (b) ein Spiegel im oberen Bereich des Erfassungstunnels montiert. Dieser ermöglicht die Positionierung der Fische im Raum und dadurch die Berechnung der Fischlänge. In der aktuell genutzten (Stand September 2016) Version (d) aus Abbildung 1.2 besteht der Boden des durchschwimmbaren Bereiches der FishCam aus einem weißen Boden mit 14 darauf montierten Halbkugeln DN 100 mm. Diese dienen der Schaffung von strömungsberuhigten Bereichen in Bodennähe um schwimmschwachen Fischen Ruhezonen zu bieten. Die Rückwand besteht aus einer matten weißen Kunststoffplatte ohne Hintergrundbeleuchtung und Musterung.

Abbildung 1.2: Die unterschiedlichen Versionen des FishCam-Systems von der ersten Version (a) bis zur aktuellen Version (d) (Stand September 2016).

Um konstante Lichtverhältnisse in den Videos zu garantieren, wurden externe Lichtquellen im FishCam-System montiert. In der letzten Version besteht die Beleuchtung aus einem Mono Flex LED Stripe mit 630 Lumen/m der rundläufig um die vordere Plexiglasscheibe montiert ist (siehe Abbildung 1.3). Verbaut sind rund 5 m LED Streifen mit weißem Licht und einer Farbtemperatur von 6000 K. Des Weiteren wird das Becken, in dem das FishCam-System eingebaut ist, so gut wie möglich abgedeckt damit kein Sonnenlicht ins Wasser einfällt.

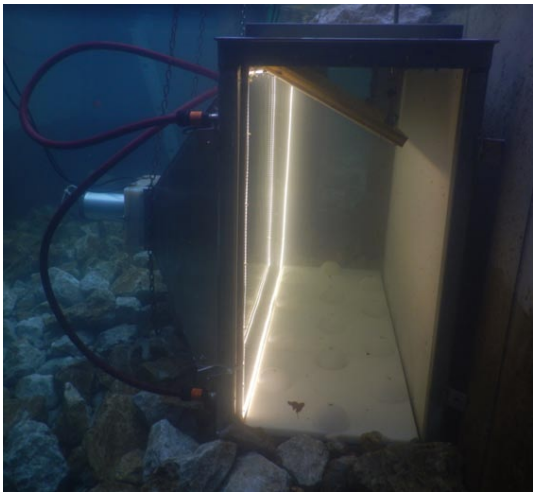

Abbildung 1.3: Seitenansicht des FishCam-Systems.

Damit alle Fische, welche die überwachte FAH durchwandern, durch den Erfassungstunnel schwimmen, wird der restliche Teil des Erfassungstunnels mit Hilfe von Gitterresten bzw. Lochblechen abgesperrt (siehe Abbildung 1.4).

Abbildung 1.4: Veranschaulichung der Absperrung rund um den Erfassungstunnel.

1.3 Idee der Automatisierung

Einhergehend mit der Idee ein auf Videoüberwachung basierendes Monitoringsystem für FAH zu entwickeln, war die Überlegung möglichst viele der anfallenden Aufgaben des Monitorings zu automatisieren. Zu diesen zählt klassischerweise die Bestimmung der Fischart, Fischlänge und der Wanderrichtung.
Relativ schnell zu Beginn des Projektes zeigte sich allerdings, dass eine der wichtigsten Funktionen die automatische Aussortierung von Videos ist, welche keine Fische enthalten. Bis zum jetzigen Zeitpunkt (Stand Juli 2016) wurden rund 1.3 Millionen Videos von den FishCam-Systemen aufgezeichnet wovon lediglich rd. 4.5 % tatsächlich zumindest einen Fisch enthalten. Der Rest der Videoaufzeichnungen wurde durch andere im Wasser treibenden Gegenstände (Laub, Holz, Müll, ...), Luftblasen bei turbulentem Wasser, Algen, welche im Erfassungstunnel wachsen und sich in der Strömung bewegen oder restliches Sonnenlicht, welches trotz abgedecktem Erfassungstunnel in das Wasser einfällt, ausgelöst.

1.4 Zielsetzung der Masterarbeit

Ziel dieser Masterarbeit ist die Entwicklung einer Software zur automatischen Objekterkennung in Videos des FishCam-Monitoringsystems. Die Software soll die Auswertung des Videomaterials verschiedener Fischaufstiege ermöglichen und Videos, die Fische enthalten von jenen trennen die keine Fische enthalten. Bei der anschließenden Fischarten- und Längenbestimmung werden nur diejenigen Videos von Experten gesichtet für die mithilfe der entwickelten Software zumindest ein Objekt als Fisch klassifiziert wurde. Mithilfe dieser Vorsortierung kann der manuelle Aufwand des Monitorings erheblich reduziert werden.

2 Beschreibung der Monitoringsoftware FishNet

Die für die automatisierte Objekterkennung entworfene Software trägt den Namen FishNet. Dieser Name steht zum einen sinnbildlich für das Fischnetz, dass aus der großen Anzahl der Videos nur jene „herausfischt“ in welchen zumindest ein Fisch vorhanden ist. Zum anderen ist die Namensgebung ein Hinweis auf den Teil des Programms, welcher für die Objektklassifizierung zuständig ist – ein Convolutional Neural Network (siehe Abschnitt 4.1.3). Die Programmstruktur von FishNet ist in Abbildung 2.1 dargestellt und lässt sich in zwei Blöcke unterteilen.

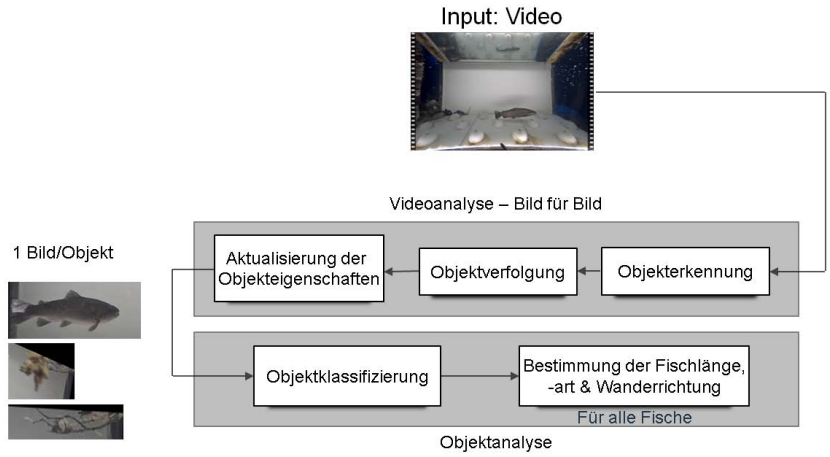

Abbildung 2.1: Entworfene Programmstruktur von FishNet.

Der erste Block steht für die Analyse der aufgenommenen Videos (siehe Kapitel 3). Hier wird das Video Bild für Bild bearbeitet, wobei sich bewegende Objekte erkannt werden und durch das gesamte Video verfolgt werden. Für jedes Objekt wird eine Datenstruktur angelegt in welcher eine Vielzahl an Informationen (z.B. Position im

Bild, Größe des Objektes, etc.) aus jedem Videobild abgespeichert werden. Zusätzlich wird für jedes Objekt ein repräsentatives Bild gespeichert.
Im Anschluss an die Videoanalyse kommt der zweite Block, in welchem die gefundenen Objekte des Videos ausgewertet werden (siehe Kapitel 4). Dazu wird jedes Objekt als erstes anhand des repräsentativen Bildes als *„Fisch“* oder *„kein Fisch“* klassifiziert. Danach werden dann nur noch die als *„Fisch“* klassifizierten Objekte betrachtet und für diese die Wanderrichtung, Länge und Fischart bestimmt.
Dass die Klassifizierung anhand eines einzigen repräsentativen Bildes pro Objekt durchgeführt wird, hat folgende Begründungen:

- Objektklassifikatoren - Algorithmen mithilfe derer die Objektklassifizierung durchgeführt wird - müssen mit einem Trainingsdatensatz an Beispielbildern der beiden Klassen *„Fisch“* und *„kein Fisch“* trainiert werden. Diese Beispielbilder sollten denen im operationellen Einsatz analysierten Bildern möglichst ähnlich sein. Werden alle Bilder eines Videos klassifiziert, kann jedes einzelne Objekt in einer Vielzahl von Positionen und Ausrichtungen zur Kamera auftauchen; all diese müssten im Trainingsdatensatz enthalten sein. Damit ein Klassifikator all diese Fälle abfangen könnte, müsste der Trainingsdatensatz schier unendlich sein.

- Der Rechenaufwand wäre enorm, wenn jedes Objekt in jedem Bild erneut klassifiziert werden müsste. Außerdem stellt sich die Frage, welchen Mehrwert an Informationen man durch die vielfache Bestimmung jedes Objekts gewinnen würde, da die Objekte sowieso über das gesamte Video verfolgt werden.

Stattdessen wurde mit der Überlegung, dass sich Fische am besten in ausgestreckter Form von der Seite bestimmen lassen, ein Maß entwickelt, welches dasjenige Bild eines jeden Objektes auswählt, welches mit dieser ausgestreckten Form am besten übereinstimmt (siehe Abschnitt 3.3).
In den nun folgenden Kapiteln wird genauer auf den Aufbau der beiden Programmblöcke eingegangen und die im Rahmen dieser Masterarbeit implementierten Bestandteile näher erläutert.

3 Block I - Videoanalyse

Dieser Teil der Software beschäftigt sich mit der Analyse der Videos, die von der Kamera aufgenommen werden. Ziel ist es, alle Objekte des Videos zu erfassen, von Bild zu Bild zu verfolgen und dasjenige Einzelbild pro Objekt auszuwählen, das im Anschluss an die Videoanalyse für die Klassifizierung genutzt wird. Am Ende dieses Blockes der Software besitzt man eine Liste von im Video erkannten Objekten mit einer Reihe von abgespeicherten Eigenschaften (siehe Abbildung 3.1).

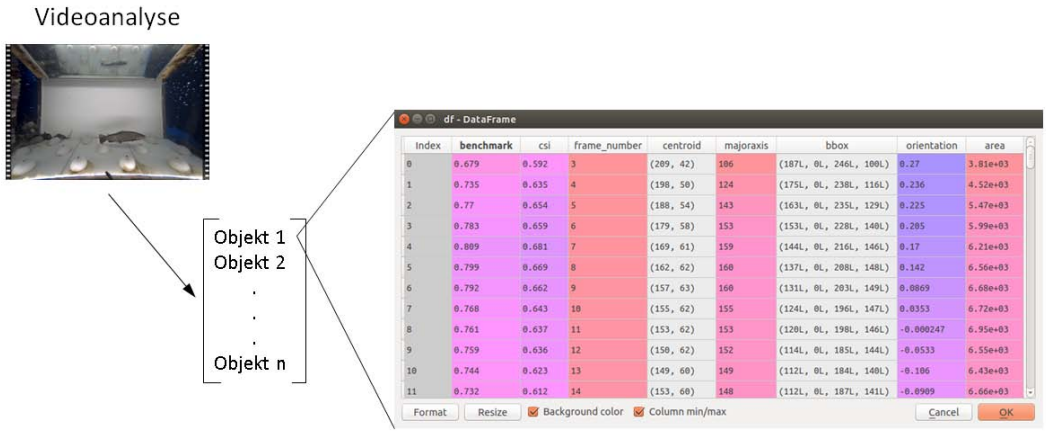

Abbildung 3.1: Im Anschluss an die Videoanalyse besitzt man zu jedem erkannten und verfolgten Objekt eine Tabelle. Diese beinhaltet unterschiedliche Eigenschaften wie Größe (area), Position (*centroid* und *bbox*), Übereinstimmung mit dem Fischmodell (*benchmark* und *csi*, siehe Abschnitt 3.3), uvm. für jedes Videobild (*frame_number*) in dem das Objekt erkannt wurde.

Um die Auswertung der Videos zu beschleunigen, wird nur ein Teilbereich des gesamten Videos betrachtet (siehe Abbildung 3.2). Manuelle Videosichtungen haben gezeigt, dass fast alle Fische im unteren Bereich den Erfassungstunnel durchschwimmen, wodurch es kaum bis keinen Verlust durch die Nicht-Betrachtung des Spiegels

bei der Objekterkennung gibt. Da die Randbereiche nicht beleuchtet sind, bzw. keine Rückwand aufweisen, und jeder Fisch für einen Auf- oder Abstieg durch die Bildmitte schwimmen muss, können die Randbereiche des Videobildes ebenfalls vernachlässigt werden. Hierdurch lässt sich die Fläche des auszuwertenden Bildes circa halbieren und die Auswertung, die zu einem Großteil pixelbasiert ist, ungefähr halbieren.

Abbildung 3.2: Das rote Viereck markiert den Bereich, der für die Videoauswertung betrachtet wird. Der Rest des Kamerabildes wird für die Videoanalyse vernachlässigt.

3.1 Erkennung von Objekten im Video

Der erste Schritt der Videobearbeitung befasst sich mit dem Erkennen von sogenannten Vordergrundobjekten in dem Videobild. Vordergrundobjekt sind jene Bereiche im Bild, die nicht zum statischen Hintergrund des aufgezeichneten Kamerabildes gehören. Dies ist kein neues Problem im Bereich der Video- und Bildbearbeitung und es existieren eine Vielzahl von unterschiedlichen Ansätzen. Viele dieser Verfahren berechnen auf verschiedenste Weise eine statische Repräsentation des Bildhintergrundes und bestimmen den Vordergrund dann über Abweichungen zu dieser Repräsentation. Andere Verfahren hingegen betrachten die dynamische Veränderung von Bild zu Bild des Videos. Allgemein lassen sich die verschiedenen Verfahren grob in die folgenden Gruppen unterteilen (Jabbar et al., 2014).

- **Einfache Hintergrundmodelle**: Bei diesen Verfahren wird für jedes Pixel ein Farbwert bestimmt, welcher den statischen Hintergrund repräsentiert. In einem beliebigem Bild werden die Pixel dann als Vordergrund klassifiziert, sobald sich ihr aktueller Farbwert wesentlich von dem des Hintergrundes unterscheidet.
- **Statistische Hintergrundmodelle**: Hierbei wird der Hintergrund aus einfachen Gauß-Verteilungen oder sogenannten Gaussian-Mixture-Models modelliert. Die Klassifizierung einzelner Pixel in Vorder- oder Hintergrund erfolgt dann über statistische Berechnungen.
- **Filter-Methoden**: Der statische Hintergrund in diesen Verfahren wird mittels unterschiedlicher Filterfunktionen geschätzt. Pixel werden als Vordergrund klassifiziert, falls ihr aktueller Farbwert von diesen Schätzungen zu weit abweicht.

Die Qualität der korrekten Klassifizierung in Vorder- oder Hintergrund unterscheidet sich bei diesen Verfahren zum Teil gravierend, jedoch kommt höhere Genauigkeit meist auch Hand in Hand mit größerem Rechenaufwand. Welches Verfahren schlussendlich gewählt wird, hängt deshalb von mehreren Faktoren ab.

- Ist die Kamera beweglich oder fixiert?
- Wie ist der Sichtbereich der Kamera beleuchtet? Nur durch natürliches Licht, also veränderlich durch Wolken und Tageszeit oder über künstliche Lichtquellen konstant?
- Ist der Hintergrund starr oder auch beweglich (z.B. sich im Wind bewegende Bäume)?
- Wird ein fortlaufendes Video betrachtet bei dem es zu signifikanten Veränderungen des Hintergrundes kommen kann oder werden eher kurze Videoclips bearbeitet?
- Wie wichtig ist die Qualität des Ergebnisses im Vergleich zur Schnelligkeit der Berechnung?

3.1.1 Verwendetes Hintergrundmodell

Wie in Abschnitt 1.2 beschrieben, gilt für den im Rahmen des Projektes entwickelten Erfassungstunnel, dass die Kamera fixiert ist, der Sichtbereich künstlich und konstant ausgeleuchtet und der Hintergrund unbeweglich ist. Des Weiteren wird nicht ein unbegrenzt fortlaufendes Video bearbeitet sondern nur kurze Videoclips. Für diese kann angenommen werden, dass die Verhältnisse im Erfassungstunnel weitestgehend unveränderlich sind und es innerhalb eines Videos zu keinen großen Veränderungen, wie Veralgung der Rückwand/des Bodens kommt.
Dies ermöglicht die Verwendung weniger komplexer Algorithmen aus der Gruppe der einfachen Hintergrundmodelle. Bei diesen wird einmalig pro Videoclip ein Hintergrundmodell berechnet und dies dann mit jedem einzelnen Bild des Videos verglichen.
Für das entwickelte Programm hat sich gezeigt, dass eine vereinfachte Version des in (Jabbar et al., 2014) vorgestellten Verfahrens ausreicht. Das eigentliche Hintergrundmodell nach Jabbar et al. (2014) setzt sich zusammen aus verschiedenen Farbbändern unterschiedlicher Farbräume, einer Repräsentation der Textur des Hintergrundes, sowie einer Repräsentation aller im Hintergrund vorkommenden Kanten. Verwendet wurden nur Repräsentationen verschiedener Farbbänder aus unterschiedlichen Farbräumen, da sich dies als ausreichend für die Erstellung des Hintergrundmodells erwies. Bei den verwendeten Farbbändern handelt es sich um:

- **Sättigungsband (S)** aus dem **HSV-Farbraum**: Dieses Farbband enthält Informationen über die Farbsättigung von Neutralgrau über weniger gesättigte Farbtöne zu gesättigten, reinen Farben.
- **Helligkeit (Y)** und **Blau-Gelb-Chrominanz (Cb)** aus dem **YCbCr-Farbraum**: Der Helligkeitskanal des YCbCr-Farbraumes ist ein Graustufenbild, wohingegen das andere Farbband die Abweichung der Farbigkeit der Pixel von Grau Richtung rot-grün darstellt.

Die einzelnen Farbbandrepräsentationen werden über Mittlung der letzten drei Bilder des Videos errechnet. Wie in Abschnitt 1.2 erläutert, ist die Videokamera so eingestellt, dass in den letzten 5 Sekunden (entspricht 60 Einzelbildern) des Videos

keine Veränderungen mehr auftreten sollten, sprich sich kein Objekt im Bild befindet. Somit können die letzten Bilder des Videos als Referenz genommen werden, wie der leere Erfassungstunnel aussieht. Algorithmus 3.1 zeigt den Pseudocode für die Berechnung des Hintergrundmodells.

Algorithmus 3.1 Bestimmung des Hintergrundmodells

Input: *Video* mit *n* Einzelbildern

1: Lade die letzten drei Bilder des Videos
2: Berechnung des mittleren Bildes
3: Konvertiere das gemittelte Bild in den HSV- und YCbCr-Farbraum

Output: *S-, Y-, Cb-Farbband*

Nachdem das Hintergrund-Modell bestimmt wurde, kann jedes Bild des Videos Pixel für Pixel mit dem Hintergrund-Modell verglichen werden und eine sogenannte „Similarity-Matrix“ errechnet werden. Dies ist ein Graustufenbild mit der gleichen Größe wie das betrachtete Bild, wobei der Grauwert von jedem Pixel anzeigt, wie sehr der aktuelle Wert dem des Hintergrund-Modells gleicht (siehe Abbildung 3.3).

Abbildung 3.3: Beispielhafte Visualisierung für die Kombination von Hintergrundmodell (links) einem beliebigem Bild des Videos (mittig) und daraus resultierender Similarity-Matrix (rechts).

Je dunkler ein Pixel, desto unterschiedlicher ist der aktuelle Farbwert im Bild im Vergleich zum Hintergrund-Modell. Entsprechend stehen helle Pixel für große Übereinstimmung zum Hintergrund-Modell. Algorithmus 3.2 zeigt den Pseudocode für die Berechnung der Similarity-Matrix. Die Faktoren die bei der Kombination der Übereinstimmungen der einzelnen Farbbänder im vierten Schritt verwendet werden, wurden experimentell ermittelt.

Algorithmus 3.2 Berechnung der Similarity-Matrix

Input: *Hintergrund-Modell* (S_H, Y_H, Cb_H) *, aktuelles Videobild*

1: Konvertiere Videobild in HSV und YCbCr-Farbraum (S_B, Y_B, Cb_B)

2: Für jedes Farbband *n*:

3: $$sim^n(x,y) = \begin{cases} \frac{n_H(x,y)}{n_B(x,y)} & \text{falls } n_H(x,y) < n_B(x,y) \\ 1 & \text{falls } n_H(x,y) = n_B(x,y) \\ \frac{n_B(x,y)}{n_H(x,y)} & \text{falls } n_H(x,y) > n_B(x,y) \end{cases}$$

4: $similarity = 0.3 * sim^S + 0.4 * sim^Y + 0.3 * sim^{Cb}$

Output: *similarity*

Aus dem Graustufenbild lässt sich mittels Schwellenwertverfahren eine binäre Vordergrundmaske erstellen, wobei schwarz (Null) für Hintergrundpixel steht und weiß (Eins) für Vordergrundpixel. Eines der hierfür bekanntesten Verfahren ist das Verfahren nach Otsu (1979). Dabei wird angenommen, dass die Verteilung aller Grauwerte einer bimodalen Verteilung folgt. Es wird dann der Grenzwert gesucht, der die Varianz zwischen den Klassen maximiert und innerhalb der Klasse minimiert. In Abbildung 3.4 ist für ein Videobild beispielhaft die Verteilung der Grauwerte der Similarity-Matrix aufgetragen. Es lässt sich die bimodale Verteilung erahnen, wobei die Anzahl der Pixel, die große Übereinstimmung mit dem Hintergrundmodell aufweisen, deutlich überwiegt. Die schwarze Linie markiert den Schwellenwert der mittels des Verfahrens nach Otsu (1979) berechnet wurde. Dieser Schwellenwert bedeutet in weiterer Folge, dass alle Pixel, die einen Grauwert größer dem Schwellenwert besitzen, als Hintergrund klassifiziert sind. Dementsprechend werden alle Pixel mit Grauwerten kleiner dem Schwellenwert als Vordergrund klassifiziert.

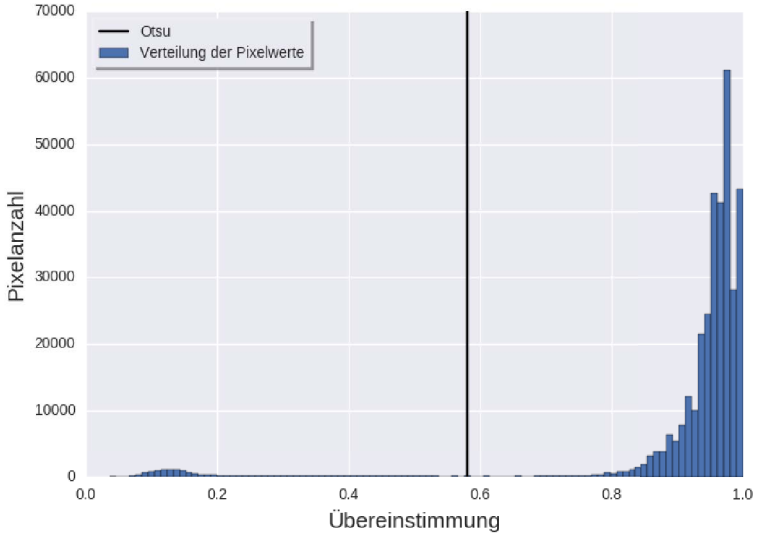

Abbildung 3.4: Histogramm aller Pixelwerte aus der Similarity-Matrix beispielhaft für ein Videobild. Niedrige Werte (dunkle Grautöne) stehen für geringe Übereinstimmung mit dem Hintergrundmodell. Werte nahe Eins symbolisieren dementsprechend eine große Übereinstimmung mit dem Hintergrundmodell. Die schwarze Linie kennzeichnet den Schwellenwert nach Otsu (1979).

Das Ergebnis des Schwellenwertverfahrens für die in Abbildung 3.3 rechts gezeigte Similarity-Matrix ist in Abbildung 3.5 zu sehen.

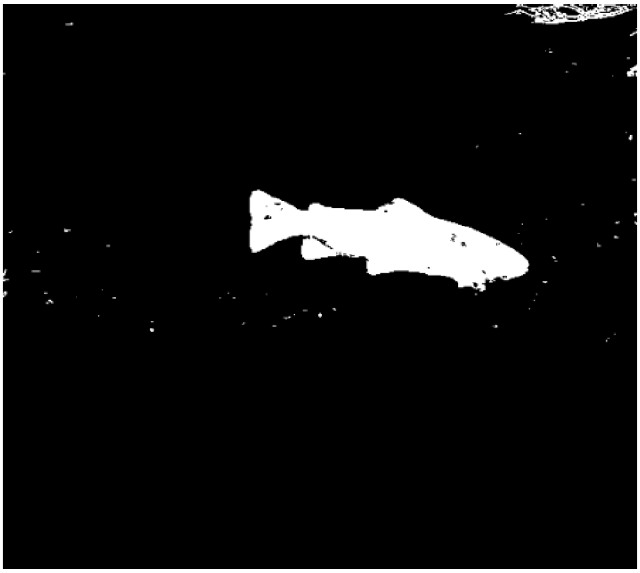

Abbildung 3.5: Ergebnis des Schwellenwertverfahrens für die in Abbildung 3.3 (rechts) dargestellte Similarity-Matrix

Zu beachten ist allerdings, dass das Otsu-Verfahren die Annahme trifft, dass sich in jedem Bild ein Vordergrundobjekt befindet und für jedes Graustufenbild einen Schwellenwert ermittelt, der die Verteilung der Grauwerte bestmöglich in Vordergrund- und Hintergrundpixel unterteilt. Dies geschieht unabhängig davon, ob sich in dem Bild tatsächlich ein Objekt befindet oder nicht. Dies führt dazu, dass auch in Bildern des Videos, in denen kein Objekt ist, Pixelregionen als Vordergrund klassifiziert werden (siehe Abbildung 3.6). Die Grauwerte der Similarity-Matrix für diese Bilder sind allerdings alle ziemlich hoch, da die Übereinstimmung mit dem Hintergrundmodell groß ist. Deswegen wurde manuell ein oberes Limit für den Schwellenwert aus dem Otsu-Verfahren festgelegt, welches bei ca. 0.9 (bzw. 229, da die Similarity Matrix im vorzeichenlosen 8bit-Format weiterbearbeitet wird) liegt. Dies kann so verstanden werden, dass der Unterschied zwischen dem aktuellen Pixel und dem Hintergrundmodell größer als 10 % sein muss, um überhaupt als Vordergrundobjekt wahrgenommen zu werden. Mit dem festgelegten oberen Grenzwert des Schwellenwertes ergibt sich für das Videobild aus Abbildung 3.6 korrekterweise eine leere (schwarze) Vordergrundmaske.

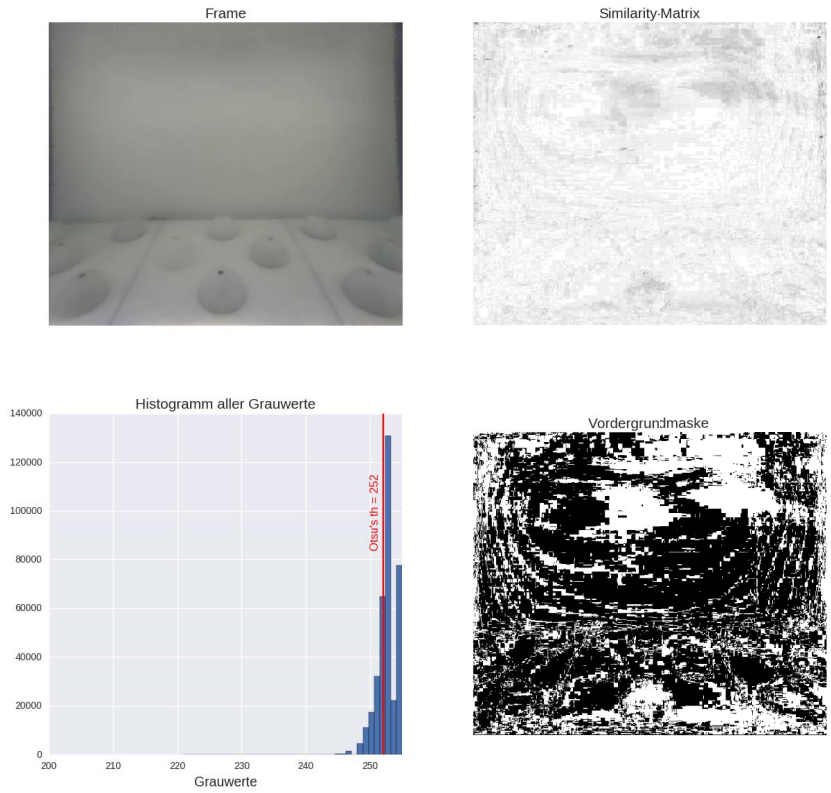

Abbildung 3.6: Veranschaulichung der Generierung der Vordergrundmaske über das Otsu-Verfahren für ein Videobild ohne Objekt und ohne oberes Limit des Schwellenwertes.

3.1.2 Binäre Bildnachbearbeitung

Da das verwendete Schwellenwertverfahren ein rein pixelbasiertes Verfahren ist, ist die resultierende Vordergrundmaske oft anfällig gegenüber Rauschen. Gemeint sind damit zumeist einzelne Pixel oder kleine Gruppen die fälschlicherweise als Vorder- oder Hintergrund klassifiziert werden. Um das Resultat des Schwellenwertverfahrens zu verbessern und das Rauschen zu reduzieren, wird die Vordergrundmaske anschließend einigen Verfahren der binären Bildbearbeitung unterzogen. Dazu gehört das Füllen von Löchern, das Entfernen von einzelnen weißen Pixel bzw. kleinen Gruppierungen bis zu einer bestimmten Größe und die beiden Verfahren Dilatation und Erosion aus dem Bereich der morphologischen Bildverarbeitung (Tcheslavski, 2009). Im Zuge der Dilatation werden Objekte gewissermaßen aufgebläht, die Erosion hingegen verkleinert Objekte. Wendet man beide Verfahren hintereinander an, kommt

es je nach Reihenfolge entweder zum sogenannten Schließen oder Öffnen von Objekten. Im Zuge des Schwellenwertverfahrens kommt es teilweise vor, dass zum Beispiel der Übergang zwischen Flosse und Fischkörper nicht als Vordergrund klassifiziert sind (ansatzweise in Abbildung 3.5 bei der Analis zu sehen) und es sich somit um zwei Objekte handeln würde. Deswegen ist die implementierte Reihenfolge Erosion gefolgt von Dilatation, also das sogenannte Schließen.
Beim Löschen kleiner Objekte muss zunächst ein Grenzwert, der festlegt bis zu welcher Größe Objekte als zu klein gelten und gelöscht werden, gewählt werden. Es gilt allerdings zu beachten: Wählt man den Wert zu hoch, werden eventuell kleine Fische nicht als Objekt erfasst. Wählt man im Gegensatz den Wert zu niedrig, müssen viel mehr Objekte verfolgt und klassifiziert werden. Das wiederum beeinflusst die Performance des Programms . Als ungefährer Richtwert kann folgender Wert gesehen werden:

- **minimale Objektgröße: 600 px**

Dies entspricht ungefähr der Größe die ein rd. 10 cm langer Fisch, der sich ganz an der Rückwand befindet.
Der genaue Ablauf der Nachbearbeitung der Vordergrundmaske ist in Algorithmus 3.3 zu sehen.

Algorithmus 3.3 Nachbearbeitung der Vordergrundmaske

Input: *Vordergrundmaske (unbearbeitet)*
MinObjektSize

1: Fülle Löcher
2: Dilatation (mit kreisförmigem strukturierendem Element)
3: Fülle Löcher
4: Erosion (mit kreisförmigem strukturierendem Element)
5: Entfernen aller Objekte mit: Fläche < minObjektSize

Output: *Vordergrundmaske (bearbeitet)*

Das Ergebnis der Nachbearbeitung für die Vordergrundmaske aus Abbildung 3.5 ist in Abbildung 3.7 zu sehen. Man sieht, dass das Rauschen vollständig entfernt wurde und auch die Maske des Fisches besser erfasst ist.

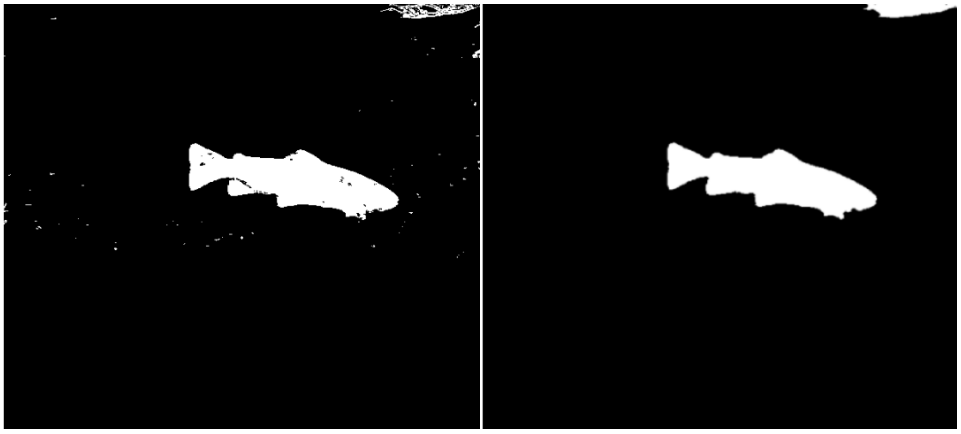

Abbildung 3.7: Vergleich zwischen der unbearbeiteten Vordergrundmaske (links) aus Abbildung 3.5 an dem Ergebnis der Nachbearbeitung (rechts).

Dies ist die finale Vordergrundmaske, die alle erkannten Objekte in weiß und Hintergrundbereiche in schwarz darstellt. Anhand dieser Maske lassen sich für die Objekte einige repräsentative Statistiken berechnen, die im Weiteren für die Objektverfolgung wichtig sind. Beispiele hierfür sind der Massenschwerpunkt oder die Größe.

3.2 Verfolgen von Objekten

Um Fische verlässlich zählen zu können, ist es wichtig, einzelne Individuen über das gesamte Video zu verfolgen. In Abschnitt 3.1 wurde erläutert wie Objekte in einem einzelnen Bild erkannt werden. Dieser Abschnitt beschäftigt sich damit, wie diese Objekte von Bild zu Bild des Videos verfolgt werden.
Für das Verfolgen von Objekten gibt es eine Vielzahl an unterschiedlichen Ansätzen. Die Verfahren lassen sich nach Yilmaz et al. (2006) in folgende Übergruppen unterteilen:

- Punktverfolgung
- Kernelverfolgung
- Silhouettenverfolgung

Allgemein gilt, dass man für das Verfolgen eine Repräsentation oder ein Modell der Objekte benötigt und das man eine Vorhersage trifft, wie dieses Modell im nächsten

Bild des Videos aussieht. In dem nächsten Bild werden dann alle erkannten Objekte mit den Vorhersagen verglichen und über ein bestimmtes Gütemaß bereits bekannten oder neuen Objekten zugeordnet. Es wird dabei oft angenommen, dass die Bewegung der Objekte gleichmäßig, ohne abrupte Veränderungen ist und die Geschwindigkeit relativ konstant bleibt. Des Weiteren vereinfachen Informationen über die Anzahl der Objekte, deren Größe oder Form maßgeblich die Komplexität der Objektverfolgung (Yilmaz et al., 2006). Diese Informationen sind allerdings im Rahmen dieses Projektes nicht gegeben, da sowohl die Anzahl von Objekten im Bild, wie auch deren Größe oder Erscheinungsform völlig variabel sind.
Nachdem das Ergebnis des vorherigen Schritts der Objekterkennung (Abschnitt 3.1) bereits eine Repräsentation der Objekte im Videobild ist (weiße Bereich der Vordergrundmaske), müssen für die Objektverfolgung noch die folgenden Probleme gelöst werden:

- Die Vorhersage für das nächste Bild sowie
- die Zuordnung von erkannten Objekten des nächsten Bildes zu Objekten aus dem vorherigen Bild

3.2.1 Positionsvorhersage

Das verwendete Verfahren für die Vorhersage von Objekten im nächsten Videobild gehört zur Gruppe der Punktverfolgung. Der Referenzpunk, anhand dem die Vorhersage und damit die Verfolgung geschieht, ist der geometrische Schwerpunkt der weißen Objekte der Vordergrundmaske. In der ersten Phase der Entwicklung wurde zur Positionsvorhersage der vielfach verwendete Kalman-Filter benutzt (Broida and Chellappa, 1986; Kalman, 1960). Letztendlich zeigte sich allerdings, dass ein wesentlich einfacheres Verfahren gleich gute Ergebnisse liefert, ohne das ähnliche Modellparameter geschätzt werden müssen. Das hier verwendete Verfahren der kleinen Geschwindigkeitsänderungen nimmt an, dass sich die Geschwindigkeit und Richtung von Bild zu Bild nicht maßgeblich verändert (Yilmaz et al., 2006). Danach kann die

Position $\vec{x}$ zum Zeitpunkt $t+1$ nach folgender Gleichung ermittelt werden:

$$\vec{x}_{t+1} = \vec{x}_t + (\vec{x}_t - \vec{x}_{t-1}) \tag{3.1}$$

Eine Ausnahme besteht für den Fall der ersten Vorhersage zum Zeitpunkt $t = 0$, sprich wenn das Objekt in dem aktuellen Bild das erste mal erkannt wurde. Hier wird angenommen, dass die Position für das Bild $t+1$ der Position aus dem vorherigen Bild entspricht. Da die Fortbewegungen der Fische und der sonstigen Objekte im Verhältnis zur Bildrate der Videos (12 fps) in der Regel gering ist, führt dies zu keinerlei Problemen. Auch für den Fall schnell schwimmender Fische (z.B. bei Jagd/Flucht-Situationen) zeigten sich keine Probleme. Zur Veranschaulichung ist in Abbildung 3.8 sowohl die vorhergesagte Schwimmstrecke (grün) als auch die tatsächlich zurückgelegte Schwimmstrecke (blau) anhand des geometrischen Schwerpunktes beispielhaft für ein Video dargestellt.

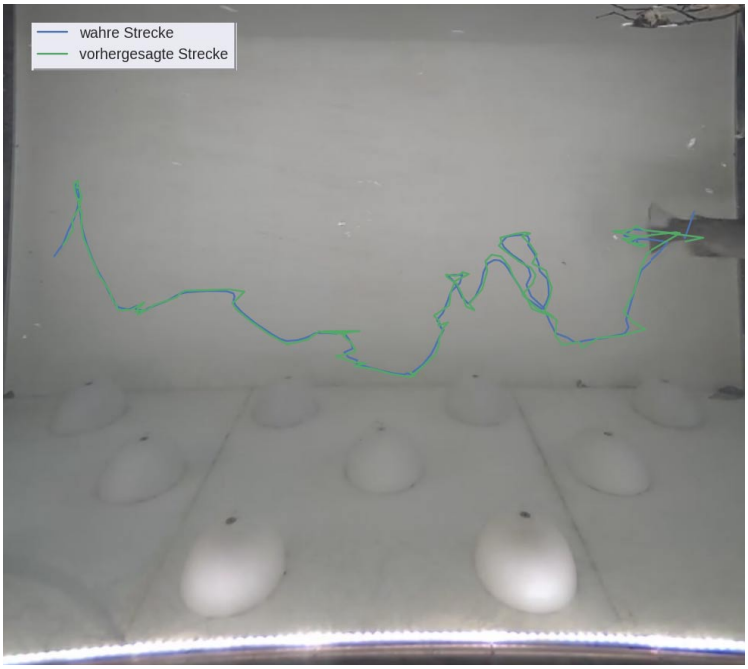

Abbildung 3.8: Veranschaulichung der vorhergesagten Schwimmstrecke (grün) gegenüber der tatsächlich zurückgelegten Strecke (blau).

3.2.2 Zuordnung von Vorhersagen und erkannten Objekten

Nachdem nun bekannt ist, wie die Position für Objekte (*tracks*) im nächsten Bild vorhergesagt wird, wird nun die Zuordnung von Vorhersagen und erkannte Objekten im nächsten Bild (*detections*) erläutert. Für diese Zuordnung wird zunächst ein Gütemaß, Kosten genannt, benötigt, dass für jede Kombination aus tracks und detections berechnet wird. Eine gängige Möglichkeit ist die Berechnung der euklidischen Distanz zwischen der vorhergesagten Position und dem geometrischen Massenschwerpunkt der erkannten Objekte:

$$c_{i,j} = \sqrt{track_i^2 - detection_j^2} \tag{3.2}$$

wobei $c_{i,j}$ für die Kosten der Zuordnung von $track_i$ zur $detection_j$ steht. Trägt man die Kosten aller Kombinationen in einer Matrix ein, erhält man die sogenannte Kostenmatrix (siehe Abbildung 3.9).

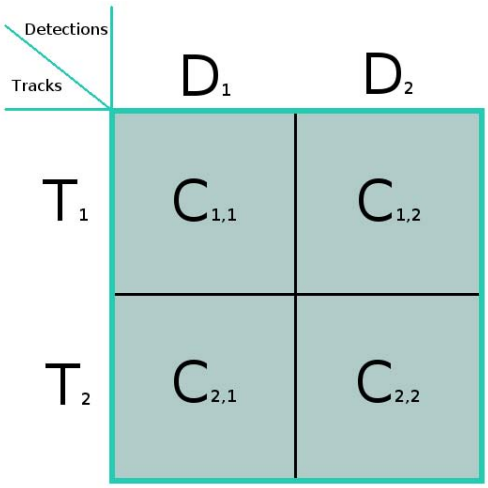

Abbildung 3.9: Darstellung der Kostenmatrix anhand eines Beispieles mit zwei tracks und zwei detections (Vgl. The MathWorks 2016).

Mit Hilfe des Kuhn-Munkres-Algorithmus (Kuhn, 1955; Munkres, 1957) lässt sich bestimmen, welches erkannte Objekt im aktuellen Bild (*detection*) einem verfolgten Objekt aus dem vorherigen Bild (*track*) zugeordnet wird. Der ungarische Algorithmus bestimmt diejenige Zuordnung, welche die Summe der Kosten minimiert.
Da dies allein nicht zu befriedigenden Ergebnissen führte, wurde das Verfahren um

zwei Punkte erweitert.

Während der Entwicklung dieses Programmteils und ersten Tests fiel auf, dass es teilweise durch fehlerhafte Erkennung der Vordergrundobjekte zu Fehlern in der Objektverfolgung kam. Zum Beispiel kann es vorkommen, dass nicht der ganze Fisch als ein Objekt erkannt wird, sondern durch Schwächen der Objekterkennung aus zwei geteilten Objekten in der Vordergrundmaske besteht (ein häufiger Fehler ist die Trennung von Fisch und der Bauch- oder Schwanzflosse). Dies passiert meist nur über ein oder zwei Videobilder, jedoch kann es mit der reinen Kostenberechnung über die euklidische Distanz dazu kommen, dass der ehemals verfolgte Fisch nicht dem eigentlichen Fisch sondern dem fälschlicherweise als eigenes Objekt erkanntem Körperteil zugeordnet wird. Der eigentliche Fisch würde nun einem neuen Objekt zugeordnet werden und es käme am Ende zu Fehlern bei der Fischzählung. Um dem entgegen zu wirken wurde als erstes die Berechnung des Kostenwertes erweitert. Die Idee ist, dass neben der euklidischen Distanz zwischen der vorhergesagten Position eines tracks und der Position einer detection auch das Größenverhältnis in die Berechnung der Kosten einfließt. Dabei sollen die Kosten für unterschiedlich große Objekte vergrößert werden und für annähernd gleich große Objekte ungefähr gleich bleiben. Es muss somit sichergestellt sein, dass der Wert des Größenverhältnisses größer als eins ist. Somit errechnen sich die Kosten für die Zuordnung von $track_i$ zu $detection_j$ nach folgender Formel:

$$c_{i,j} = \frac{\sqrt{track_i^2 - detection_j^2} * max(area_i, areaj)}{\min(area_i, areaj)} \tag{3.3}$$

wobei $area_i$ für die Größe des $track_i$ steht und entsprechend $area_j$ für die Größe der $detection_j$. Mit der Größe ist jeweils die Summe der Pixel eines Objektes aus der schwarz-weiß Maske gemeint.

Die zweite Änderung ist die Erweiterung der Kostenmatrix um einen festen Kostenwert für die Nicht-Zuordnung eines tracks zu einer detection beziehungsweise einer detection zu einem track (siehe Abbildung 3.10).

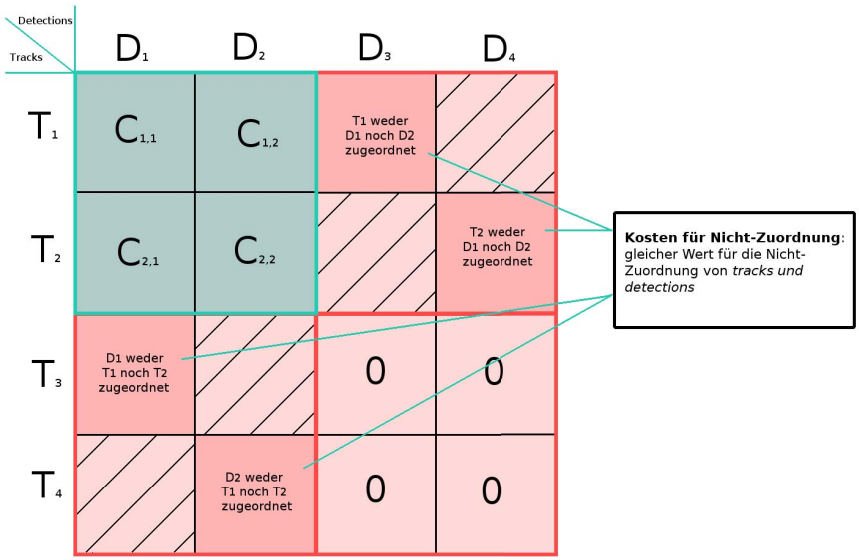

Abbildung 3.10: Erweiterte Kostenmatrix um einen fixen Kostenwert für die Nicht-Zuordnung von tracks zu detections und vice versa am Beispiel vom zwei tracks und detections (Vgl. The MathWorks 2016).

Dies ermöglicht die Initialisierung eines neuen tracks immer dann, wenn einer detection kein track zugeordnet wird und die Löschung eines tracks wann immer einem track keine detection zugeordnet wird (dazu genauer in Abschnitt 3.2.3). Die Größe dieses fixen Kostenwertes ist ein Modellparameter, der je nach gesetztem Wert maßgeblichen Einfluss auf die Performance der Verfolgung von Objekten hat. Ein Wert, der sich über den Lauf der Entwicklung bewehrt hat ist:

- **Kosten für die Nicht-Zuordnung: 50**

3.2.3 Weitere Parameter des Verfolgungsalgorithmus

Da es nicht sinnvoll ist, einen track sofort zu löschen, wenn er in einem Videobild keiner detection zugeordnet ist, wurden weitere Parameter eingebaut, welche die „Gesundheit“ eines tracks überprüfen.
Zum einen gibt es einen Parameter, welcher die Anzahl an Videobildern zählt, für welche ein track kontinuierlich keiner detection zugeordnet wurde. Überschreitet dieser Wert einen definierten Grenzwert, wird der track gelöscht, da er zu lange aus dem

Bild ist. Dieser Wert hängt maßgeblich von der Videofrequenz ab. Als ein ungefährer Richtwert hat sich ein Wert von ungefähr 4 Sekunden als sinnvoll erwiesen. Bei der Videoauflösung des FishCam-Systems entspricht dies einem Wert von:

- **Anzahl an Videobildern, die ein track nicht im Bild sein darf: 48**

Des Weiteren gibt es einen Mindestwert an Videobildern, die ein verfolgtes Objekt im Video auftreten muss, bevor es für den weiteren Verlauf des Programms als glaubhaftes Objekt erkannt wird. Dies hat den Hintergrund, dass durch die verwendete Methodik zur Berechnung der Vordergrundmaske in einzelnen Bildern Bereiche des Bildes als Vordergrund klassifiziert werden, die kein Vordergrundobjekt sind. Außerdem ist anzunehmen, dass ein passierender Fisch oder ein passierendes Objekt eine Mindestanzahl an Bildern im Video zu sehen sein muss, bevor er den Erfassungstunnel durchschwommen haben kann. Jedoch muss dieser Wert vorsichtig gewählt werden, da sonst schnell schwimmende Fische (zum Beispiel in Jagdszenen oder beim schnellen Abstieg) nicht erfasst werden. Als ein sinnvoller Wert hat sich erwiesen:

- **Anzahl an Bildern, die ein Objekt mindestens sichtbar sein muss: 3**

3.3 Auswahl des "besten" Bildes von jedem Objekt

Wie bereits in Kapitel 2 erläutert, klassifiziert das Programm jedes Objekt anhand eines einzigen Bildes. Dabei stellt sich die Frage mit welchem Maß sich bestimmen lässt, wie gut sich ein Bild im Vergleich zu einem Anderen für die Klassifizierung eignet. Die Grundidee der gewählten Vorgangsweise besteht darin, dass Fische zur Artenbestimmung zumeist von der Seite in ausgestreckter Form betrachtet werden (siehe Abbildung 3.11).

Abbildung 3.11: Bestimmungstafel heimischer Fischarten in Österreich (Landesfischereiverband-Oberösterreich, 2013).

Gleichzeitig sollte der Fisch so nahe wie möglich an der Kamera und somit so groß wie möglich im Bild sein. Hinzu kam zum Zeitpunkt der Entwicklung dieses Features (Objekteigenschaft), dass die Maske des Fisches so gut wie möglich den gesamten Fisch erfassen sollte (mehr dazu siehe Abschnitt 4.1.1). Unter all diesen Gesichtspunkten wurde das sogenannte *Fischmodel* entwickelt.

3.3.1 Herleitung des Fischmodels

Die Idee des *Fischmodels* ist es, eine Vergleichsform zu erzeugen, die der durchschnittlichen Körperform der Fische in den zu überwachenden Gewässern in der Seitenansicht entspricht (siehe z.B. Abbildung 3.11). Da die Objekte während der Videoanalyse in Form ihrer binären Maske erfasst und verfolgt werden, liegt auch das Fischmodel als binäre Maske vor.

Die binäre Maske des Fischmodels wurde erstellt, indem von 225 manuell ausgewählte Fischen aus unterschiedlichen Videos die Maske gewählt wurde, in der die Körperform am besten in ihrer Gesamtheit erfasst wurde und der Fisch sich seitlich ausgestreckt zur Kamera befindet. Danach wurden diese 225 Masken in ihrer Größe

(Anzahl der Pixel) normiert und anhand ihrer Flächenschwerpunkte überlagert. Die resultierende Matrix enthält in jeder Zelle einen Wert zwischen 0 und 225, welcher angibt, zu wie vielen der 225 Fischmasken diese Zelle gehörte.

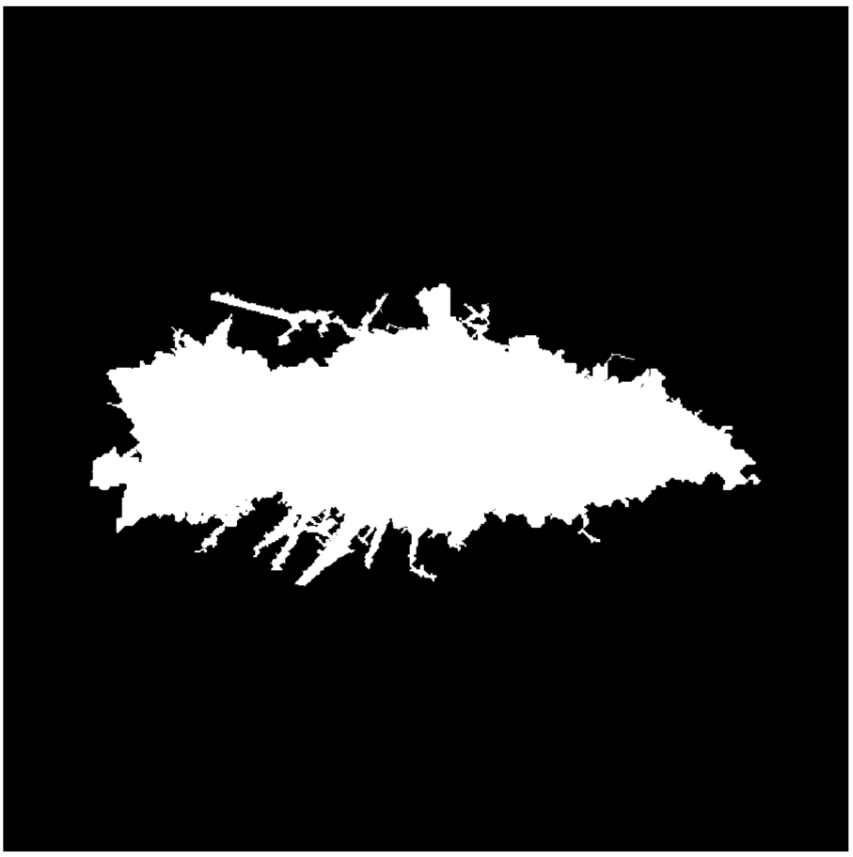

Abbildung 3.12: Überlagerung von 225 manuell ausgewählten Fischmasken als binäres Bild.

Betrachtet man lediglich welche Zellen überhaupt zu einer der 225 Fischmasken gehört, ergibt sich das Bild aus Abbildung 3.12. Zwar lässt sich ansatzweise die Form eines Fisches erkennen, dessen Kopf auf der rechten Seite und dessen Schwanzflosse auf der linken ist, jedoch ist die Silhouette weit entfernt von der des durchschnittlichen Fisches.

Berechnet man aus dieser Matrix jedoch den Mittelwert aller Zellen, die zumindest von einer der 225 Fischmasken belegt sind und betrachtet nur alle Zellen deren Wert über diesem Mittelwert liegen, erhält man als Ergebnis das Fischmodel aus Abbildung 3.13.

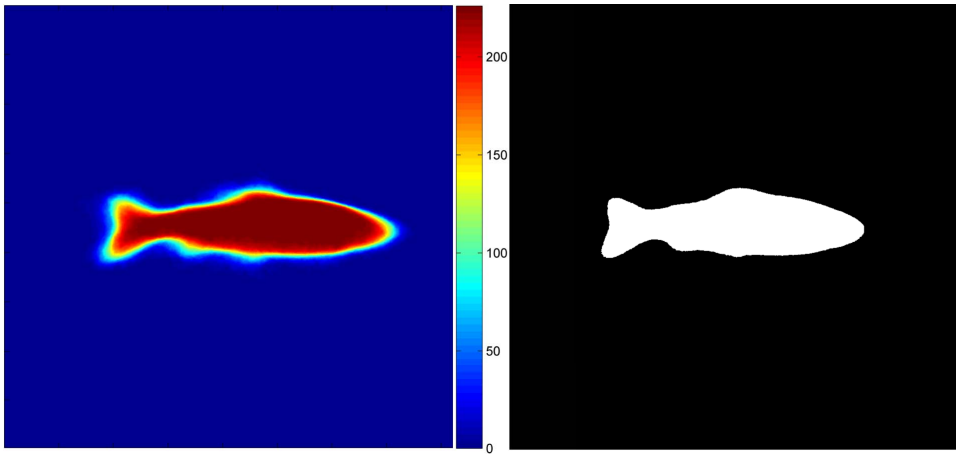

Abbildung 3.13: Links: Heatmap aus Überlagerung aller 225 manuell ausgewählten Fischmasken. Rechts: Resultierendes Fischmodel aus Grenzwertbildung mit dem Mittelwert aller belegten Zellen.

In dem binären Bild des Fischmodels aus Abbildung 3.13 erkennt man nun eindeutiger die typische Form eines stromlinienförmigen Fischkörpers.

3.3.2 Gütemaß für die Bestimmung des "besten" Bildes

Um zu Überprüfen, welches Bild, bzw. welche Maske eines Fisches am besten mit dem Fischmodel übereinstimmt, bedarf es eines geeigneten Gütemaßes. Darin soll neben der Übereinstimmung mit dem Fischmodel auch die Nähe zur Kamera einfließen, so dass sich das endgültige Gütemaß aus zwei Komponenten zusammensetzt:

$$\textit{Gütemaß} = \text{Gütemaß(Fishmodel)} + \text{Gütemaß(Kameranähe)} \tag{3.4}$$

3.3.2.1 Gütemaß aus der Übereinstimmung mit dem Fischmodel

Das für die Evaluierung der Übereinstimmung mit dem Fischmodel verwendete Gütemaß – der *critical sucess index* (csi) – kommt vielfach in der 2D-Modellierung im Bereich der Meteorologie und Hydrologie zum Einsatz (Donaldson et al., 1975; Schaefer, 1990).
Abbildung 3.14 zeigt beispielhaft die Anwendung aus dem Bereich der Hochwas-

sersimulation (Weichel, 2011), wobei sich die Ergebnisse der Gegenüberstellung von Simulation und Beobachtung auch als Tabelle darstellen lassen.

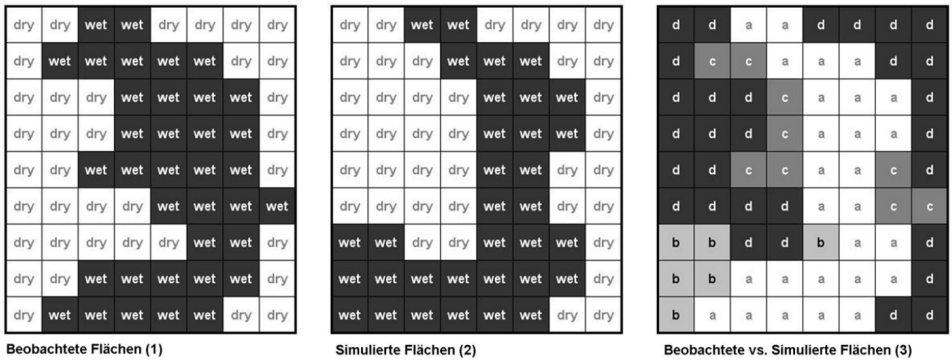

Abbildung 3.14: Gegenüberstellung (3) von überschwemmten Flächen aus Beobachtung (1) und Simulation (2) (Weichel, 2011).

		Beobachtung	
		überschwemmt	nicht überschwemmt
Simulation	überschwemmt	a	b
	nicht überschwemmt	c	d

Tabelle 3.1: Schematische Vergleichsmatrix der Gegenüberstellung von Abbildung 3.14(3).

Der *critical sucess index* (csi) errechnet sich für dieses Beispiel nach folgender Formel:

$$csi = \frac{\sum a}{\sum a + \sum b + \sum c} \tag{3.5}$$

Mit a sind alle Pixel oder Zellen gekennzeichnet, die sowohl in der Beobachtung als auch in der Simulation als überschwemmt gekennzeichnet sind, mit b entsprechend die Zellen, für die eine Überschwemmung simuliert jedoch nicht beobachtet wurde sowie vice versa, mit c all jene Zellen in denen eine Überschwemmung beobachtet wurden, diese jedoch in der Simulation nicht berechnet wurde. Der gesamte Bereich, der in beiden Fällen als nicht überschwemmt gekennzeichnet ist (d) geht nicht in das Gütemaß ein. Aus der Formel ist ersichtlich, dass der csi einen Wertebereich von [0,1] besitzt und einen desto größeren Wert annimmt, desto größer die Übereinstimmung zwischen Beobachtung und Simulation ist.
Dies lässt sich relativ einfach in ein Gütemaß für die Übereinstimmung mit dem

Fischmodel übertragen, wobei das binäre Bild des Fischmodels (siehe Abbildung 3.13) als Beobachtung gilt und jede Maske von einem Objekt, dass mit diesem verglichen wird, als Simulation. In Abbildung 3.15 ist beispielhaft eine Gegenüberstellung vom Fischmodel mit einer beispielhaften Fischmaske abgebildet.

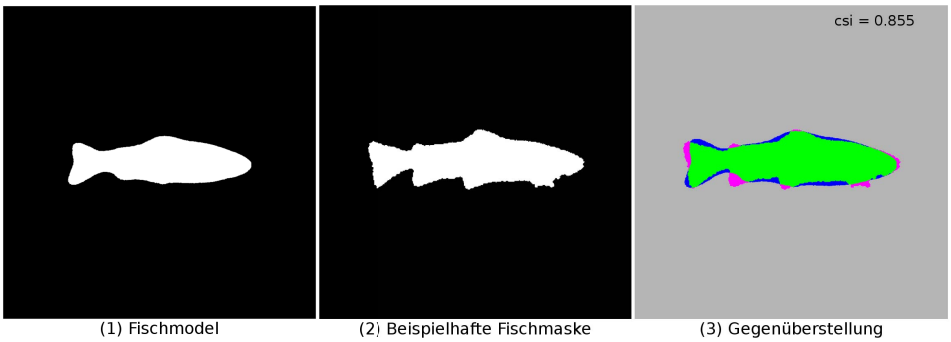

Abbildung 3.15: Gegenüberstellung (3) vom Fischmodel (1) und einer beispielhaften Fischmaske (2).

Die *grünen* Bereiche symbolisieren dabei die Flächen, die sowohl zum Fischmodel gehören, als auch zu der Fischmaske (oben mit a bezeichnet), die *blauen* Bereiche jene Fläche, die zum Model gehören, jedoch nicht von der Fischmaske abgedeckt sind (oben c). Die *pinken* Bereiche sind jenen Pixel zugeordnet, die zur Fischmaske gehören, jedoch nicht zum Fischmodel (oben als b bezeichnet). Der *graue* Bereich entspricht der Fläche die in beiden binären Masken schwarz ist und ist oben als d benannt. Bei dieser Gegenüberstellung ist es wichtig, dass die Maske des zu überprüfenden Objektes in die Horizontale rotiert wird und in ihrer Größe normiert. Des Weiteren muss darauf geachtet werden, dass die Ausrichtung der Fischmaske der des Fischmodels entspricht, sprich Schwanzflosse links, Kopf rechts.
Abbildung 3.16 zeigt die Gegenüberstellung vom Fischmodel mit der Maske des selben Fisches aus einem anderen Bild des Videos. In dem Bild des Videos ist der Fisch leicht gekrümmt, die Maske schlechter ausgeschnitten, was alles in allem zu einem schlechteren csi führt.

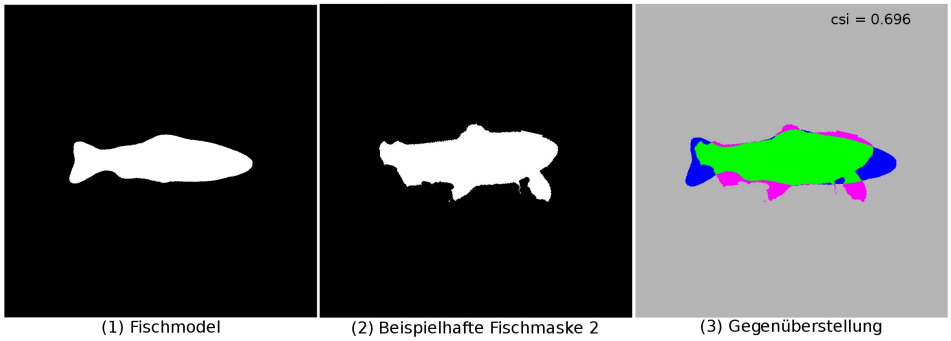

Abbildung 3.16: Gegenüberstellung (3) vom Fischmodel (1) mit der Maske des selben Fisches aus Abbildung 3.15 aus einem anderen Videobild (2).

Es ist anzumerken, dass der csi nicht Wert auf die Übereinstimmung der Silhouette legt und somit der Form, sondern auf die möglichst große Übereinstimmung der gesamten Fläche. Dies war zum Zeitpunkt des Entwurfes dieses Programmteiles ein entscheidendes Kriterium, da die Objektklassifizierung anhand von diversen Eigenschaften gemacht wurde, welche zum Teil aus der binären Maske errechnet wurden (siehe Abschnitt 4.1.1). Für diese Merkmale war es wichtig, dass die Fischmaske so genau wie möglich den gesamten Fisch abdeckt. Mit der aktuellen Art der Klassifizierung (siehe Abschnitt 4.1.3) spielt dies keine Rolle mehr. Allerdings hat sich gezeigt, dass der so verwendete csi als gutes Gütemaß für die Bestimmung des „besten" Bildes funktioniert und wurde deshalb nicht mehr verändert.

3.3.2.2 Gütemaß für die Bestimmung der Nähe zur Kamera

Neben der möglichst guten Ansicht von der Seite ist auch die Nähe zur Kamera ein wichtiger Faktor bei der Bestimmbarkeit von Objekten. Je näher ein Objekt bzw. Fisch an der Kamera ist, desto größer ist das Bild und damit auch seine Detailstärke. Deswegen fließt als weiteres Maß zur Bestimmung des „besten" Bildes eines Objektes die relative Länge im Verhältnis zur Bildbreite ein. Dadurch, dass die relative Länge und nicht die absolute Länge verwendet wird, ist sichergestellt, dass der Wertebereich dieses Gütemaßes ebenfalls im Bereich [0,1] liegt. Die Länge des Objektes, welche für die Berechnung verwendet wird, entspricht der Hauptachse einer Ellipse (*major axis length*) mit dem selben zweiten Zentralmoment wie das Objekt in der binären

Maske (scikit-image image development team, 2016).

$$lengthfactor = \frac{major\,axis\,length}{Bildbreite} \tag{3.6}$$

3.3.2.3 Verknüpfung der beiden Gütemaße

Während der Entwicklung hat sich gezeigt, dass die Schwankungsbreite des Längengütemaßes wesentlich größer ist, als die des Gütemaßes vom Fischmodel. Dies führte dazu, dass die Nähe zur Kamera zum entschiedenen Faktor wurde. Da jedoch vor allem wichtig ist, den Fisch gut von der Seite zu sehen und erst in weiterer Folge möglichst groß im Bild, wurde eine Skalierung des Längengütemaßes eingebaut. Statt dass der Wert zu 100 % in das endgültige Gütemaß einfließt, wird nur der halbe Wert genommen. Das finale Gütemaß - *benchmark* genannt -, über welches das „beste“ Bild ausgewählt wird, ergibt sich somit zu:

$$benchmark = csi + 0.5 * lengthfactor \tag{3.7}$$

wobei *csi* für den nach Formel 3.5 berechneten Wert steht und sich der *lengthfactor* nach Formel 3.6 berechnet.

4 Block II - Objektanalyse

Im Anschluss an den ersten Block, die Videoanalyse, folgt im Programmablauf der zweite funktionelle Teil der Software, die Objektanalyse. Hier werden alle Objekte, die im Video erkannt wurden, zunächst anhand eines repräsentativen Bildes als „Fisch“ oder „kein Fisch“ klassifiziert. In weiterer Folge werden für alle Fische die Wanderrichtung, Länge und Fischart bestimmt. Im Rahmen dieser Masterarbeit wurde die Automatisierung aller Arbeitsschritte bis einschließlich der Bestimmung der Wanderrichtung abgeschlossen. Für die automatisierte Längenbestimmung gibt es erste Ansätze; die automatische Fischartenbestimmung steht noch aus (siehe Kapitel 7).
Im Laufe der Programmentwicklung wurde der Teil der Objektklassifizierung am häufigsten überarbeitet, sodass sich dieser Teil am meisten, teils grundlegend, verändert hat. Die verschiedenen Ansätze werden in ihrer chronologischen Reihenfolge der Programmentwicklung in den Abschnitten 4.1.1 bis 4.1.3 erläutert. In Abschnitt 4.2 wird auf die Bestimmung der Wanderrichtung eingegangen.

4.1 Objektklassifizierung

Zu Beginn des zweiten Programmabschnittes werden alle erkannten Objekte eines Videos zunächst in die Klassen „Fisch“ oder „kein Fisch“ unterteilt. Wie anfangs in Abschnitt 1.3 erläutert, ist dies eine der Hauptfunktionen der entwickelten Software. Bis zum jetzigen Zeitpunkt werden einige Monitoringaufgaben (Fischarten- und Längenbestimmung) manuell von Fischexperten durch Sichtung des Videomaterials erledigt. Da von allen aufgezeichneten Videos nur ca. 3 % zumindest einen Fisch

enthalten, führt die automatisierte Vorsortierung der Videos und Klassifikation in „Fisch“ und „kein Fisch“ zu einer deutlichen Reduktion des Arbeitsaufwands. Die Klassifizierung der Objekte geschieht anhand des nach Abschnitt 3.3 bestimmten „besten“ Bildes eines Objektes. In Abbildung 4.1 sind einige Beispielbilder beider Klassen gezeigt, um die große Variabilität (Gewässertrübe, Bildqualität, Bildgrundton, Objektart, etc.) innerhalb der Klassen zu verdeutlichen.

In traditionellen Ansätzen der Objektklassifizierung werden zumeist speziell ausgewählte Objekteigenschaften (sogenannte Features) berechnet und ein Klassifikator trainiert, der mit Hilfe dieser Features Objekte in vordefinierte Klassen unterteilen soll (Alsmadi et al., 2009; Matai et al., 2012; Nery et al., 2005; Rodrigues, 2010; Rova et al., 2007).Die ersten Versionen des in FishNet verwendeten Objektklassifikators war an diese Methodik angelehnt und wird in Abschnitt 4.1.1 und 4.1.2 näher beschrieben.

Seit ca. 2012 dominiert im Bereich der visuellen Informatik allerdings eine andere Methodik zur Bildklassifizierung: sogenannte *(Deep) Convolutional Neural Networks (ConvNets)* (Russakovsky et al., 2015).Diese Art von Klassifikatoren arbeitet mit dem reinen RGB-Farbbild eines Objektes als Input. ConvNets lernen während der Trainingsphase selbstständig, welche Features sich am besten zur Differenzierung der einzelnen Klassen eignen. Die letzte und aktuelle Version von FishNet arbeitet mit solch einem ConvNet als Klassifikator und wird in Abschnitt 4.1.3 eingehender beschrieben.

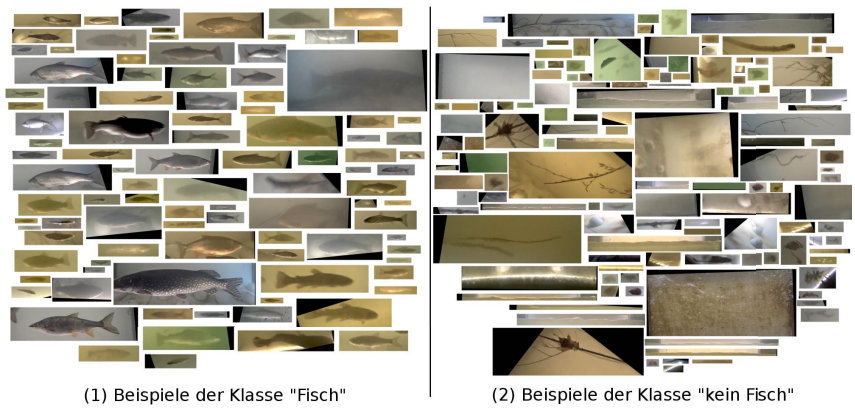

Abbildung 4.1: Beispielbilder der beiden zu unterscheidenden Klassen „Fisch“ und „kein Fisch“.

4.1.1 FishNet Version 1.0: Ein Klassifikator mit allen Features

Die ersten Ansätze zur Objektklassifizierung waren stark an bestehende Systeme zur Fischklassifizierung angelehnt. Auch wenn sich die Publikationen meist mit der Unterscheidung unterschiedlicher Fischarten beschäftigen (Alsmadi et al., 2009; Matai et al., 2012; Nery et al., 2005; Rodrigues, 2010; Rova et al., 2007),dienen sie doch als Anhaltspunkt bei der Wahl geeigneter Features für die Fischerkennung. Darauf beruhend wurde in einem ersten Schritt eine Vielzahl an Features implementiert (siehe Abschnitt 4.1.1.1) und ein erster Klassifikator zur Objektunterscheidung trainiert (siehe Abschnitt 4.1.1.2 und 4.1.1.3).

4.1.1.1 Liste aller Features

Die verwendeten Features beruhen hauptsächlich auf Nery et al. (2005) und lassen sich in die folgenden Gruppen unterteilen:

- **Formparameter**: Diese Parameter beschreiben die Form des Objektes und werden anhand der binären Maske (und nicht anhand des Farbbildes) des Objektes berechnet. Zu diesen Parametern gehören:
    - Circularity: Stellt ein Maß für die Komplexität der Objektkontur dar und wird mittels des quadrierten Umfangs und der Fläche des Objektes berechnet.
    - Aspect Ratio: Ist das Verhältnis der Längs- zu Querachse eines Objektes.
    - Rectangle Fit Factor: Gibt an, wie sehr das Objekt das kleinste umschließende Rechteckt ausfüllt. Je näher das Objekt einem Rechteck ähnelt, desto näher ist dieser Wert an 1.
    - Area Perimeter Ratio: Ein weiteres Maß für die Komplexität der Objektkontur, welches sich aus dem Verhältnis zwischen Fläche und Umfang eines Objektes bestimmt.

- BENCHMARK und CSI: Gütemaß für die Übereinstimmung der Objektform mit dem Fischmodell nach Abschnitt 3.3.
- BILDMOMENTE: Beschreiben die Verteilung aller zu einem Objekt gehörenden Pixel im Bezug auf die Hauptachse des Objektes. Nach Cadieux et al. (2000) wurden die ersten sieben HU-MOMENTE (Hu, 1962) verwendet.

- **FARBPARAMETER**: Diese Parameter beschreiben die farblichen Eigenschaften eines Objektes und werden für bestimmte Regionen des Fischkörpers und ausgewählte Farbbänder berechnet. Nach Nery et al. (2005) werden mittlere Farbwerte für den Rücken (ventral) und Bauchbereich (dorsal) des Fisches aus Farbbändern des YcbCr- und des HSV-Farbraumes (insgesamt 8 Feature) bestimmt.
- **TEXTURPARAMTER**: Diese Parameter beschreiben die Muster innerhalb der Objektfläche und werden mithilfe eines Graubildes, das aus einem konvertierten Farbbild erzeugt wird, bestimmt. Die zwei verwendeten Features (INERTIA und ENERGY) geben an, wie wahrscheinlich bestimmte Abfolgen von Grauwertkombinationen benachbarter Pixel sind.

Die Liste umfasst somit insgesamt 23 Features die sowohl aus dem Farbbild, der binären Maske und auch dem Graubild berechnet werden.

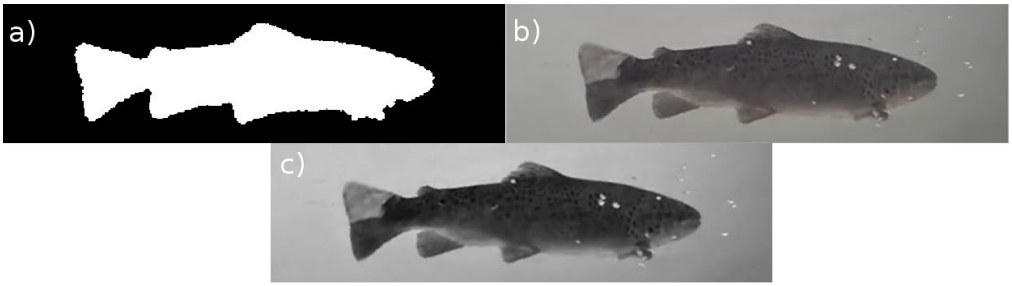

Abbildung 4.2: a) Binäre Maske zum dazugehörigen b) Farbbild und c) das entsprechende Graubild. Anhand dieser drei Bilder werden alle Features für ein Objekt bestimmt.

4.1.1.2 Klassifikator: Random Forest

Nach der Entwicklung und Implementierung der Features wurde als nächstes ein Klassifikator ausgewählt. Dieser übernimmt - nach einer Trainingsphase - im späteren Betrieb die Unterscheidung der Objekte in „Fisch“ und „kein Fisch“. Nach Caruana et al. (2008) gehört der von Leo Breiman und Adele Cutler entwickelte Random Forests (Breiman, 2001) zu den problemübergreifend besten Klassifikatoren. Diese Einschätzung beruht auf einer Reihe von Vorteilen des Random Forests, wie zum Beispiel:

- Benötigt keine Skalierung der Features
- Relativ schnell zu trainieren
- Relativ unsensibel gegenüber unterschiedlicher Modellparameter-Einstellungen und benötigen somit keine große Anpassung an die Problemstellung

Aus diesem Grund wurde der Random Forest als Klassifikator gewählt.
Random Forests sind aus vielen unabhängigen Entscheidungsbäumen aufgebaut und gehören deshalb zur Klasse der Ensemble Modelle. Ein Entscheidungsbaum wiederum setzt sich aus vielen einzelnen Entscheidungsknoten (siehe Abbildung 4.3) zusammen. In der Verbindung lassen sich so komplexe Klassengrenzen im Feature-Raum bilden.

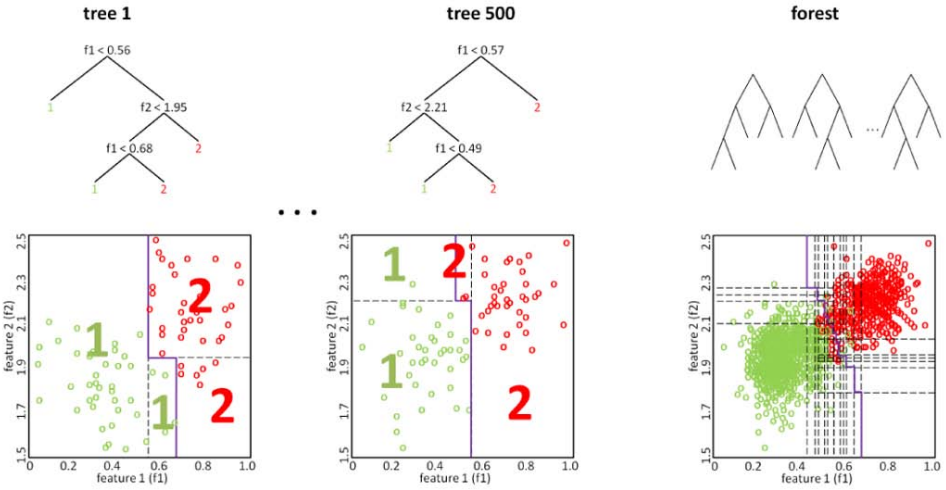

Abbildung 4.3: Schematische Veranschaulichung eines Random Forests. Durch die Verknüpfung mehrerer einzelner Entscheidungsbäume können komplexe Klassengrenzen gebildet werden. Abbildung aus Hanselmann et al. (2009).

Die Entscheidungsbäume werden während der Trainingsphase anhand von Beispieldaten trainiert und auf folgende Art und Weise gebildet (Breiman, 2001):

- Aus dem aus N Beispielen bestehendem Trainingsdatensatz werden n Beispiele per Zufall ausgewählt (mit Wiederholungen). Diese Daten bilden die Trainingsgrundlagen für diesen einen Baum.
- An jedem Entscheidungsknoten werden aus den M vorhandenen Features eine Teilmenge m $<<$ M zufällig ausgewählt und anhand dieser m Features die bestmögliche Teilung der Daten in die beiden Klassen bestimmt. Der Wert m ist konstant für alle Entscheidungsbäume im Random Forest.
- Jeder Baum wird bis zu einer maximalen Größe ausgebildet, die manuell festgelegt wird.

In der Testphase treffen dann alle Entscheidungsbäume einzeln anhand der übergebenen Input-Daten eines Objektes eine Klassenentscheidung und als finales Ergebnis des Random Forests wird diejenige Klasse gewählt, für die am häufigsten von allen Entscheidungsbäumen gestimmt wurde.

4.1.1.3 Erzielte Genauigkeit

Der zum Training des Random Forest zu diesem Zeitpunkt zur Verfügung stehende Datensatz bestand aus einer relativ geringen Anzahl an Beispielen: rund 4400 Beispiele, wobei nur knapp ein Drittel zu der Klasse „Fisch“ gehörte und der Rest zur Klasse „kein Fisch“. Außerdem stammten die Daten fast ausschließlich aus Videos mit klarem Wasser, da zu Beginn der Programmentwicklung im Frühjahr/Sommer 2014 diese Trübungsverhältnisse deutlich überwogen. Für das Training des Random Forest wurden 70 % der Daten verwendet. Die verbliebenen 30 % wurden für den Test des Klassifikators genutzt. Wie in Tabelle 4.1 zu sehen ist, ist das Ergebnis mit einer Genauigkeit von rund 96 % richtig klassifizierter Objekte relativ zufriedenstellend. In Abschnitt 4.1.2 wird jedoch erläutert, warum dieses Ergebnis leicht trügerisch ist und die Objektklassifizierung erweitert wurde/werden musste.

		Vorhersage	
		Fisch	Kein Fisch
Wahre Klasse	Fisch	372	24
	Kein Fisch	24	882

Tabelle 4.1: Konfusionsmatrix der Klassifizierungsergebnisse des Testdatensatzes für die erste Version des Random Forest Klassifikators

4.1.2 FishNet Version 1.1: Erweiterung um klar/trüb Klassifizierung

Wie bereits erwähnt, stammten die am Anfang zur Verfügung stehenden Datensätze hauptsächlich aus Wintermonaten bzw. aus der Möll bei dem Kraftwerk Rottau mit nahezu kristallklarem Wasser über das gesamte Jahr. Im weiteren Projektverlauf kamen jedoch mit den Fischaufstiegen an der Drau, Fischa und Salzach Flüsse hinzu, bei denen die Trübe des Flusswassers im Verlauf des Jahres zwischen kristallklar und absoluter Trübe schwankt (siehe Abbildung 4.4). Des Weiteren gab es Probleme mit dem automatischen Weißabgleich der Kamera und der damals verbauten LED Beleuchtung (3000 K Farbtemperatur und 900 Lumen pro Meter bei 4,5 verbauten Metern LED-Streifen), was dazu führte, dass die Videos große Schwankungen im Grundton hatten (vgl. die ersten zwei Reihen mit den letzten zwei Reihen in Abbildung 4.4). Da der Random Forest anfangs jedoch lediglich mit Objekten aus klarem Wasser trainiert wurde, stellten die aus diesen Videos extrahierten Objekten den Klassifikator vor große Probleme. Zum einen, da die Farbbilder eine ganz andere Qualität aufwiesen und sich somit auch die daraus berechneten Farbfeatures stark von denen des in Abschnitt 4.1.1 verwendeten Trainingsdatensatzes unterschieden.

Abbildung 4.4: Beispiele für Unterschiede in der Gewässertrübe, sowie der Beleuchtung bzw. dem automatischen Weißabgleich der Kamera.

Zum anderen ist die Qualität des verwendeten pixelbasierten Objekterkennungsalgorithmus aus Abschnitt 3.1 bei trübem Wasser wesentlich geringer, was zu zwei Problemen führte: Erstens, dass Objekte teilweise nicht erkannt wurden und zweitens, dass ein erkanntes Objekt nicht genau entlang seiner Körperkontur vom Hintergrund ausgeschnitten wurde (siehe Abbildung 4.5). Dies führte wiederum dazu, dass die Features, die anhand der binären Maske berechnet werden, nicht mit den bekannten Trainingsdaten übereinstimmten.

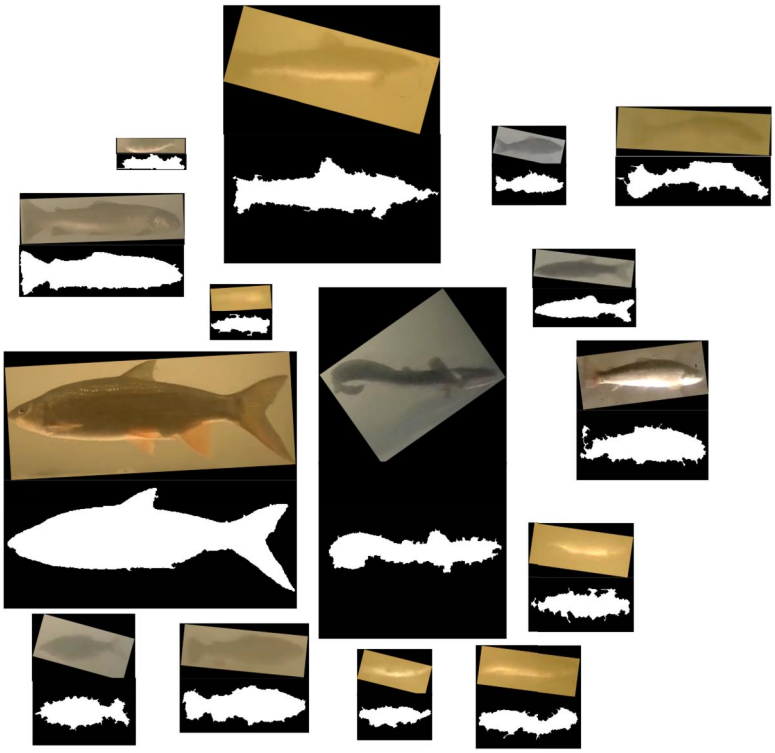

Abbildung 4.5: Unterschiede in der Qualität der Objekterkennung je nach Trübe des Flusswassers.

All diese Punkte führten dazu, dass die Genauigkeit der Objektklassifizierung in der Anwendung wesentlich geringer war, als die des Testdatensatzes.

Eine der offensichtlichsten Möglichkeiten zur Erhöhung der Genauigkeit ist die Vergrößerung des Trainingsdatensatzes und speziell die Hinzunahme von Beispielen aus trübem Wasser. Dies allein führte aber nicht zu zufriedenstellenden Ergebnissen und löste nicht das Problem, dass Objekte im trüben Wasser zum Teil gar nicht als Vordergrundobjekte erkannt wurden.

Aus diesem Grund wurden eine Reihe von Änderungen an der Methodik aus Abschnitt 4.1.1 vorgenommen. Zum einen wurde die Objekterkennung leicht abgeändert, so dass Objekte auch in kontrastarmen Videos wieder zuverlässig erkannt wurden. Des Weiteren wurden neue Features für die Objektklassifizierung implementiert. Um die Genauigkeit der Klassifizierung weiter zu erhöhen, wurde ein Maß zur Unterscheidung von klarem und trübem Wasser entwickelt und der Trainingsdatensatz in Beispiele aus klarem bzw. trübem Wasser aufgeteilt. Für diese Teildatensätze wur-

den dann separate Klassifikatoren trainiert. Eine weitere Neuerung war die Idee ein Ensemble zu trainieren, in dem die separat generierten Ergebnisse zum Schluss gemittelt werden.

4.1.2.1 Anpassung der Objekterkennung an trübes Wasser

Das Problem, dass Objekte zum Teil nicht erkannt wurden, konnte ziemlich schnell auf den mangelnden Kontrast in den Videos mit trübem Wasser zurück geführt werden (siehe Abbildung 4.6). Ein weiterer hiermit verbundener Punkt ist das manuell festgelegte obere Limit für den Grenzwert nach Otsu (1979)(siehe Abschnitt 3.1.1), der für die Abtrennung von Vorder- und Hintergrund verwendet wird. Dieser ist zwar nötig, um in Videobildern ohne Objekt keine künstlich erzeugten Objekte zu generieren, sorgt allerdings im Fall von trüben, kontrastarmen Videos dafür, dass Fische nicht als Vordergrund erkannt werden.

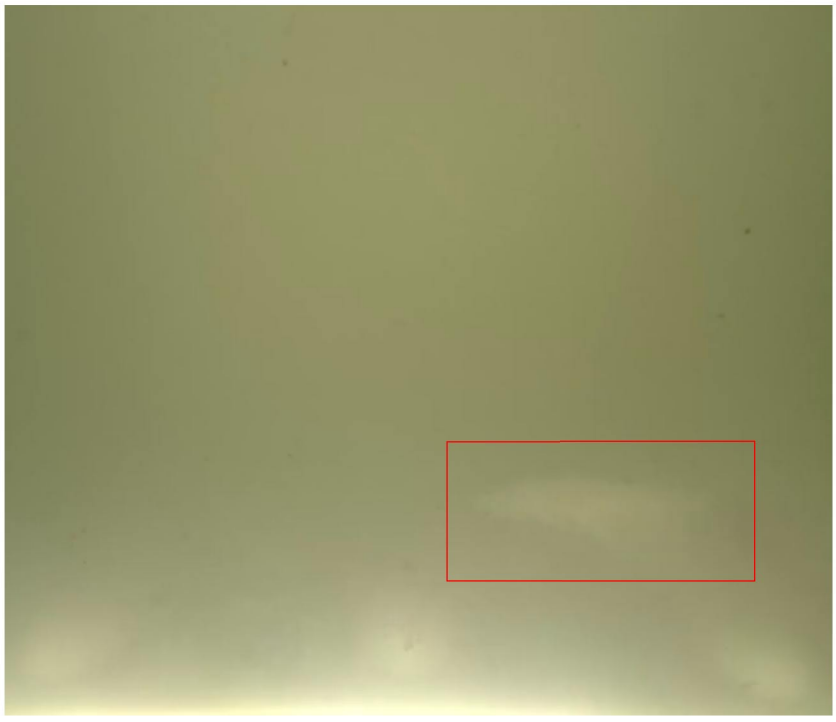

Abbildung 4.6: Beispiel für einen Fisch in trübem Wasser. Der Bildbereich des Fisches ist durch das rote Rechteck markiert.

Dies liegt daran, dass sich die Bereiche des Fisches im Bild aufgrund der starken Trübung nur unwesentlich von den Pixelwerten des Hintergrundmodells unterscheiden. Die Werte in der Similarity-Matrix sind alle relativ hoch und der errechnete Grenzwert aus dem Otsu-Verfahren liegt über dem oberen, manuell festgelegten Limit (siehe Abschnitt 3.1.1). Eine Möglichkeit die Verteilung der Grauwerte zu entzerren ist die Multiplikation der Similarity Matrix mit sich selbst. Normiert man die Grauwerte zuvor auf den Wertebereich zwischen 0 und 1, führt die Quadrierung dazu, dass sich kleine Werte mehr reduzieren als Werte nahe 1 und sich die Verteilung dadurch entzerrt (siehe Abbildung 4.7). Der aus dieser entzerrten Verteilung errechnete Schwellenwert liegt niedriger als der ursprüngliche und eventuell sogar unter dem manuell fixierten Limit. Es hat sich gezeigt, dass durch dieses Vorgehen auch in kontrastarmen Bildern Objekte erkannt werden und gleichzeitig der Vorzug eines nach oben limitierten Grenzwertes, der sicherstellt, dass in Bildern ohne Objekt auch keines erkannt wird, gewahrt werden kann.

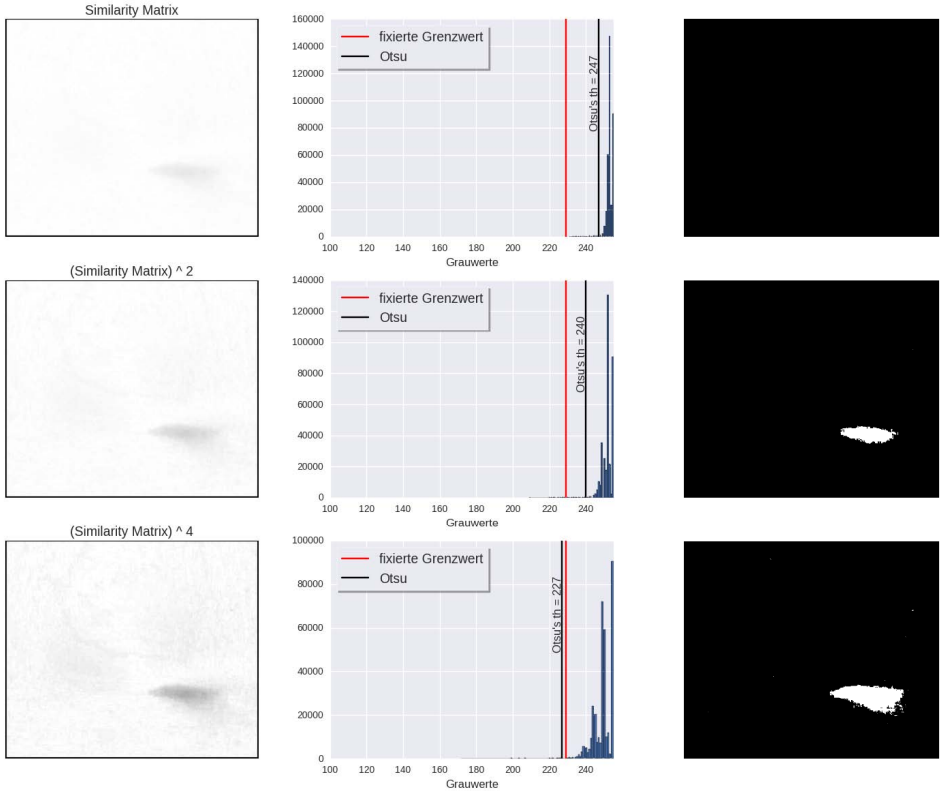

Abbildung 4.7: Vergleich der Auswirkung der Quadrierung der Similarity Matrix auf das Ergebnis der Vordergrundmaske. Liegt der vom Otsu-Verfahren ermittelte Grenzwert (schwarze Linie) über der des manuell gesetzten Grenzwertes, so wird die Maske über das Schwellenwertverfahren mit dem manuell Grenzwert ermittelt.

4.1.2.2 Unterscheidung zwischen klarem und trübem Wasser

Wie bereits angedeutet wurde als nächstes eine Möglichkeit gesucht, automatisch zu ermitteln, ob das Wasser im aktuellen Video trüb oder klar ist. Die Idee ist, dass für jeden dieser beiden Fälle separate Trainingsdatensätze erstellt werden und separate Klassifikatoren trainiert werden. Dadurch soll die Streuung der Features innerhalb einer Klasse reduziert werden, wodurch Objektklassen für den Klassifikator leichter zu unterscheiden sind.

Nach Versuchen mit unterschiedlichen Merkmalen, wurden am Ende zwei Merkmale gewählt, die sich aus dem Bildbereich des Überganges vom Spiegel zur Rückwand berechnen. Die Überlegung war, dass der Spiegel und vor allem die hintere Kante im trüben Wasser nicht sichtbar sind, im klaren Wasser natürlicherweise schon (siehe

Abbildung 4.8).

Abbildung 4.8: Vergleich der Sichtbarkeit des Spiegels in klarem bzw. trüben Wasser.

Um die beiden Merkmale zu berechnen wird der rot-markierte Bildausschnitt in ein Graustufenbild umgewandelt und das Mittel der Farbwerte über die Bildbreite gebildet. Trägt man den resultierenden Vektor an Graustufenwerten in ein Diagramm auf, kann man deutliche Unterschiede zwischen den beiden Kurven erkennen (siehe Abbildung 4.9). Die in grün dargestellte Kurve zeigt den Verlauf der gemittelten Grauwerte für ein Video mit trübem Wasser. Im Vergleich zur blauen Kurve, die aus einem Video mit klarem Wasser resultiert, ist der Verlauf nahezu homogen. Die blaue Kurve hingegen zeigt deutliche Sprünge im Randbereich des Spiegels auf.

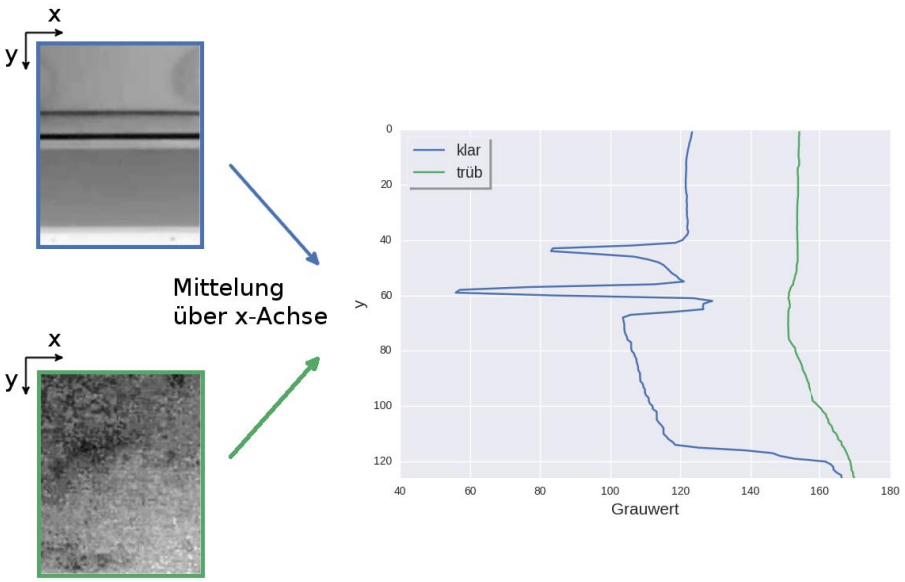

Abbildung 4.9: Veranschaulichung der Unterschiede der Grauwerte an der Spiegelkante bei trübem und klarem Wasser.

Aus den in Abbildung 4.9 rechts gezeigten Kurven werden die folgenden Merkmale zur Unterscheidung der Gewässertrübe berechnet.

- MAX_RANGE: Der Abstand zwischen dem größten und kleinsten Grauwert
- DELTA_CUM: Die Summe der absoluten Steigungen der Geraden

Für 125 Videos wurden diese Merkmale berechnet und manuell bestimmt, ob es sich um klares oder trübes Wasser handelt. Die Punktwolke der beiden Merkmale für diese Beispielvideos ist in Abbildung 4.10 dargestellt. Bei der Auswahl der Beispielvideos wurde darauf geachtet, dass möglichst alle Abstufungen von Gewässertrübe vorhanden sind.

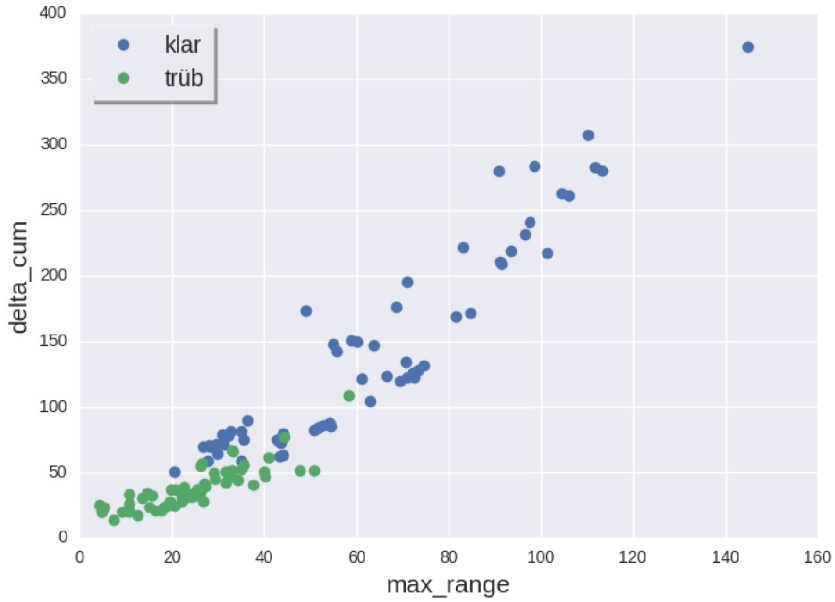

Abbildung 4.10: Darstellung der 125 Beispielvideos anhand der beiden Merkmale zur Bestimmung der Gewässertrübe.

Zu sehen ist, dass sich die beiden Klassen, „klar“ und „trüb“, relativ gut anhand der beiden Merkmale unterscheiden lassen. Um in der Videoauswertung automatisch entscheiden zu können, ob es sich im aktuellen Video um klares oder trübes Wasser handelt, wurde anhand dieser Beispieldaten ein binäres Logit-Modell trainiert (Cox, 1958), wobei 1 für klares Wasser steht und 0 für trübes Wasser. Gefüttert mit den beiden Merkmalen gibt dieses Modell einen Wert zwischen 0 und 1 zurück. Dies kann in einen Prozentsatz übersetzt werden, welcher angibt, wie klar das Wasser ist.

Für die Kombination der unterschiedlichen Klassifikatoren (siehe Abschnitt 4.1.2.5) werden daraus zwei Gewichte berechnet:

- W_KLAR: Ergebnis des Logit-Modells für die beiden berechneten Merkmale.
- W_TRÜB: entspricht (1 – W_KLAR) und steht somit sinnbildlich für den Grad der Gewässertrübe.

Mit Hilfe dieser beiden Gewichte ergibt sich die finale Klassifizierung nach folgender Formel:

$$P = w_{klar} * P_{klar} + w_{trüb} * P_{trüb} \tag{4.1}$$

wobei mit P_{klar} beziehungsweise $P_{trüb}$ das Ergebnis des Ensembles für klares bzw. trübes Wasser gemeint ist (siehe Abschnitt 4.1.2.5) und P die Wahrscheinlichkeit angibt, dass das Objekt zur Klasse „Fisch“ gehört. Ist dieser Wert größer als 0,5 wird das Objekt als Fisch klassifiziert, im anderen Fall als „kein Fisch“.

4.1.2.3 Neue Features

Auch die Liste der Features wurde erweitert, um die Genauigkeit der Klassifizierung zu erhöhen. Hinzugefügt wurden:

- Generic Fourier Descriptors (GFD): Werden mittels Spektralanalyse der gesamten Objektform für verschiedene Radial- und Winkelfrequenzen berechnet. Nach (Zhang and Lu, 2002) wurden die GFD bis zu einer radialen Auflösung von 3 und einer Winkelauflösung von 12 bestimmt (Gesamtanzahl von 52 GFD).
- Cityblock Distance: Abstandsmaß zwischen dem GFD Vektor des Fischmodels und dem betrachteten Objekt. Dient als weiteres Maß für die Ähnlichkeit eines Objektes zu dem Fischmodel.

Mit diesen 53 neuen Features bestand die Liste aller verfügbaren Features nun aus insgesamt 76 unterschiedlichen Features.

4.1.2.4 Reduzierung der Anzahl an Features

Da die Berechnung aller 76 Features relativ rechenaufwendig ist und die Bedeutung der einzelnen Feature auf das Klassifizierungsergebnis unbekannt war, wurde in einem nächsten Schritt, ähnlich zu Nery et al. (2005), die Wichtigkeit der einzelnen Features bestimmt. Ziel dieser Evaluierung ist es, schlussendlich nur eine repräsentative Teilmenge aller Features zu verwenden. Zur Überprüfung der Wichtigkeit der einzelnen Features für die korrekte Klassifizierung von Objekten, wurde das in MATLAB implementierte Verfahren der sequentiellen Featureselektion verwendet. Das Verfahren funktioniert wie folgt:

- Gewählt wird jenes Feature, welches die größte Genauigkeit im Testdatensatz erzielt. Dieses Feature gilt nun als gesetzt.
- Nun wird für dieses Feature und jede Kombination an den übrig gebliebenen Features ein neuer Klassifikator trainiert und erneut anhand der Testdaten ausgewertet.
- Nun wird dasjenige Feature als zweites gewählt, welches in Kombination mit dem ersten die größte Genauigkeit bei der Auswertung des Testdatensatzes erzielte.
- Diesem Schema folgend wird nun das dritte, vierte usw. Feature ausgewählt, bis zu dem Zeitpunkt, wo sich die Genauigkeit des Klassifikators nicht mehr verbessert.

Diese Auswertung wurde separat für zwei unterschiedliche Datensätze durchgeführt. Einmal für einen Datensatz der nur mit Beispielen aus klarem Wasser (nach Abschnitt 4.1.2.2) bestand und einmal für einen Datensatz nur mit Beispielen aus trübem Wasser.

Da der gewählte Klassifikator, der Random Forest, eine große Zufallskomponente enthält (siehe Abschnitt 4.1.1.2) gleicht keine Auswertung der Anderen. Aus diesem Grund wurden rund 400 Durchgänge für die sequentielle Featureselektion durchgeführt. Bemerkenswert ist, dass in allen 400 Durchläufen, sowohl für den Klarwasser-Datensatz als auch für den Trübwasser-Datensatz, das Feature benchmark (nach Abschnitt 3.3) als erstes Feature ausgewählt wurde und somit das stärkste Feature

für die Unterscheidung zwischen den Klassen „Fisch“ und „kein Fisch“ ist. Des Weiteren ist zu sehen, dass in trübem Wasser die Farbfeatures eine stärkere Bedeutung haben und im klaren Wasser vermehrt auch Formfeatures gewählt wurden (siehe Anhang für genaue Auflistung der gewählten Feature-Subsets). Dies kann daran liegen, dass im trüben Wasser die Segmentierung des Fisches vom Hintergrund schlechter funktioniert und somit die Aussagekraft der Formfeatures stark vermindert ist.
Anhand der Auswertung der sequentiellen Featureselektion wurden für beide Fälle, klares bzw. trübes Wasser, sechs unterschiedliche Feature-Subsets ausgewählt. Zum einen die drei Sets, welche am öftesten gewählt wurden und zum anderen die drei, die die besten Genauigkeiten erzielten.

4.1.2.5 Ensemble an Random Forest Klassifikatoren

Eine weitere Möglichkeit die Genauigkeit von Klassifizierungen zu erhöhen ist, sich nicht auf einen einzigen Klassifikator zu verlassen, sondern ein Ensemble von Klassifikatoren zu verwenden (Breiman, 1996; Opitz and Maclin, 1999). In Abbildung 4.11 ist die Funktionsweise von einem Ensemble an Klassifikatoren schematische dargestellt. Ein Objekt wird von voneinander unabhängigen Klassifikatoren klassifiziert und die Ergebnisse anschließend zu einem finalen Resultat kombiniert. Sowohl für den Datensatz aus Klarwasser-Videos, als auch für den aus Trübwasser-Videos, wurden jeweils sechs Random Forests trainiert. Dabei wurden die nach Abschnitt section 4.1.2.4 verwendeten Feature-Subsets benutzt. Sprich, jeder der insgesamt 12 Random Forests wurde mit unterschiedlichen Features trainiert.

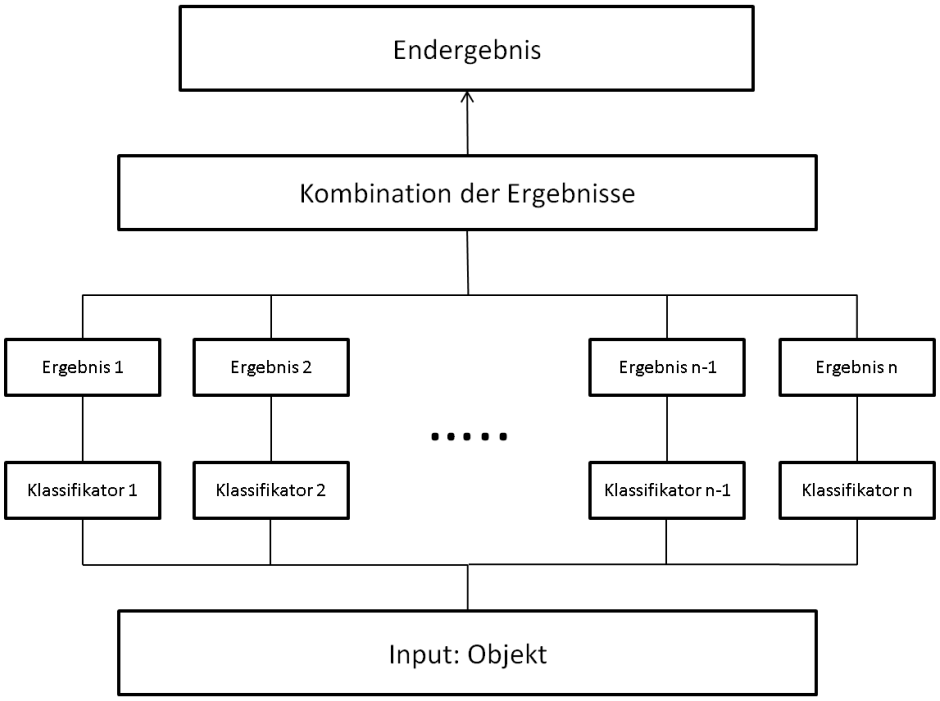

Abbildung 4.11: Schematische Darstellung der Funktionsweise von einem Ensemble an Klassifikatoren.

Die Ergebnisse der einzelnen Random Forests werden über Mittlung kombiniert. Das finale Klassifizierungsergebnis ergibt sich somit nach folgender Formel:

$$P_{klar/trüb} = \frac{1}{6} * \sum_{i=1}^{6} P_{klar/trüb,i} \tag{4.2}$$

wobei mit $P_{klar/trüb,i}$ die Wahrscheinlichkeit gemeint ist, mit welcher der Random Forest i das Objekt der Klasse „Fisch“ zuordnet.

4.1.2.6 Erzielte Genauigkeit

Die Datensätze die zum Training der Ensembles für klares und trübes Wasser verwendet wurden, unterschieden sich stark in der Größe. Da der Klassifikator größere Probleme mit der Unterscheidung von Objekten in trübem Wasser hatte, wurde hierfür ein deutlich größerer Trainingsdatensatz generiert.
Für trübes Wasser standen rund 12.000 Beispiele zur Verfügung, wovon wesentlich

mehr zur Klasse „kein Fisch“ gehörten. Das Ensemble wurde anhand von 85 % der Daten trainiert und die übrigen 15 % zur Auswertung verwendet. Das Ergebnis der Auswertung ist in Tabelle 4.2 zu sehen. Die Genauigkeit der Klassifizierung liegt bei 90,11 %.

		Vorhersage	
		Fisch	Kein Fisch
Wahre Klasse	Fisch	242	162
	Kein Fisch	45	1645

Tabelle 4.2: Konfusionsmatrix der Testdaten-Auswertung für den Trübwasser-Datensatz.

Für klares Wasser bestand der Datensatz aus rund 3200 Beispielen. Davon gehört ca. ein Drittel zur Klasse „Fisch“ und der Rest zur Klasse „kein Fisch“. Das Ensemble an Random Forests wurde ebenfalls anhand von 85 % der Daten trainiert und die übrigen 15 % wurden ebenfalls zu Auswertung genutzt (siehe Tabelle 4.3). Die erzielte Genauigkeit von 91,04 % liegt leicht über der des Trübwasser-Ensembles.

		Vorhersage	
		Fisch	Kein Fisch
Wahre Klasse	Fisch	149	35
	Kein Fisch	15	359

Tabelle 4.3: Konfusionsmatrix der Testdaten-Auswertung für den Klarwasser-Datensatz.

Die Werte liegen relativ weit unter den erreichten 96 % aus Abschnitt 4.1.1.3, sie spiegeln jedoch eher die Genauigkeit wider, mit der bei der Auswertung weiterer Videodatensätze gerechnet werden konnte. Das System der Objektklassifizierung wurde in dieser Form für ca. 1 Jahr und zur Auswertung mehrerer 100.000 Videos verwendet.

4.1.3 FishNet Version 2.0: Klassifizierung mittels Convolutional Neural Network

Auch wenn sich die Zuverlässigkeit der Objektklassifizierung in trübem Wasser durch die Berücksichtigung der Gewässertrübe, wie in Abschnitt 4.1.2 dargestellt, verbes-

serte, gab es in der Anwendung weiterhin Probleme. Aufgrund von teilweise starken Wasserstandsschwankungen (z.B. während Probebetrieb oder Optimierung der FAH) ergab die entwickelte klar/trüb-Erkennung trügerische Ergebnisse: das linke Bild in Abbildung 4.12 zeigt ein Beispiel für trübes Wasser, in dem sich allerdings der für die klar/trüb-Erkennung verwendete Übergang von Spiegel und Rückwand außerhalb des Wassers befindet. Im rechten Bild ist dieser Übergang trotz klarem Wassers nicht zu erkennen, da er durch Reflexionen an der Wasseroberfläche verdeckt wird. Da der Logit-Klassifikator jeweils der falschen Gewässertrübe-Klasse die größere Wahrscheinlichkeit zuweist, entscheidet in diesen Fällen schlussendlich das Random Forest Ensemble über die Objektklasse, welches für die jeweils falschen Sichtverhältnisse trainiert wurde.

Abbildung 4.12: Zwei Beispiele an denen die entwickelte klar/trüb-Erkennung scheitert. Im linken Bild sorgt der weit abgefallende Wasserstand dafür, dass der Spiegel komplett sichtbar ist, das trainierte Logit-Model also „klares Wasser“ ergab, wohingegen im rechten Bild der abgefallende Wasserspiegel im Pyramidenstumpf dafür sorgt, dass die hintere Spiegelkante gar nicht zu sehen ist, obwohl das Wasser klar ist. Das Resultat: „trübes Wasser“ laut dem Model aus Abschnitt 4.1.2.2.

Aufgrund der damit verbundenen Anzahl an Fehlklassifizierungen, wurde ein neuer Klassifikator, der ohne Unterscheidung aufgrund der Gewässertrübe funktioniert, gesucht. Die Recherche im Bereich der Bildklassifizierung führte schlussendlich zu *(Deep) Convolutional Neural Networks (ConvNets)*. Im Jahr 2012 gewann ein Deep Convolutional Neural Network, das AlexNet (Krizhevsky et al., 2012), erstmals den ImageNet Wettbewerb (Russakovsky et al., 2015). ImageNet ist ein Wettbewerb zur Objektklassifizierung und -lokalisierung mit einem riesigen Datensatz an

Bildern aus hunderten unterschiedlicher Klassen von Tieren, Pflanzen, Fahrzeugen, Essensgerichten und Alltagsgegenständen. Darunter befinden sich teils schwierig zu unterscheidende Klassen wie 120 verschiedene Hundearten (siehe Abbildung 4.13).

Abbildung 4.13: Beispiel unterschiedlicher Hundeklasse aus dem ImageNet-Datensatz. Abbildung von Karpathy (2014)

Das Maß der Modellgüte überprüft, wie häufig sich die wahre Klasse eines Objekts unter den fünf Klassen, denen der Klassifikator die höchsten Wahrscheinlichkeiten zuweist, befindet. Dieser Wert konnte von 74,3 % von 2011 ohne ConvNet (Sánchez and Perronnin, 2011) auf 96,43 % im letzten Jahr (He et al., 2015) mit einem ConvNet verbessert werden. Dies liegt ca. 1,5 % über der Genauigkeit, die in einer Vergleichsstudien von Menschen erreicht wurde(Russakovsky et al., 2015). Seit dem fulminanten Gewinn des AlexNets im Jahr 2012 dominiert diese Art von Klassifikatoren daher viele Bereiche der Bild- und Objektklassifizierung.

ConvNets waren keine Neuentwicklung im Jahr 2012, sondern existierten schon lange davor(LeCun et al., 1998). Allerdings ermöglichte erst die weiterentwickelte Hardware-Technologie (vor allem Grafikkarten mit ihren vielen kleinen Prozessoren) das Trainieren komplexer und vielschichtiger Netzwerke, wodurch ConvNets ihre Stärke bekommen.

Auch wenn ConvNets in vielen Bereichen der Computer Vision ihre Vorzüge gegenüber anderen Verfahren bewiesen haben, war zunächst nicht klar, wie gut sie mit den detail- und kontrastarmen Bildern - wie sie bei trübem Wasser im FishCam-System aufgenommen werden - zurechtkommen.

4.1.3.1 Funktionsweise von Convolutional Neural Networks

Convolutional Neural Networks sind sehr ähnlich zu gewöhnlichen Neuronalen Netzen. Neuronale Netze bestehen aus vielen Neuronen, die in (mehreren) aufeinanderfolgenden Schichten (Layern) angeordnet sind (siehe Abbildung 4.14). In der Regel sind alle Neuronen eines Layers mit allen des vorherigen und des nachfolgenden Layers verbunden. Innerhalb eines Layers besteht jedoch keine Verbindung zwischen den einzelnen Neuronen. Jede Verbindung zwischen zwei Neuronen besitzt ein Gewicht, das während der Trainingsphase des Netzwerkes optimiert wird. Als Eingangsdaten erhält das Netzwerk einen Vektor an Eingangsgrößen. Aus dem Vektor des letzten Layers, dem sogenannte Output-Layer, lässt sich im Falle der Klassifikation die Wahrscheinlichkeit pro Klasse ableiten. Die Aktivierung eines Neurons ergibt sich aus dem Skalarprodukt aller Aktivierungen des vorherigen Layers mit ihren spezifischen Gewichten (plus eines Bias-Wertes) und der Anwendung einer Aktivierungsfunktion.

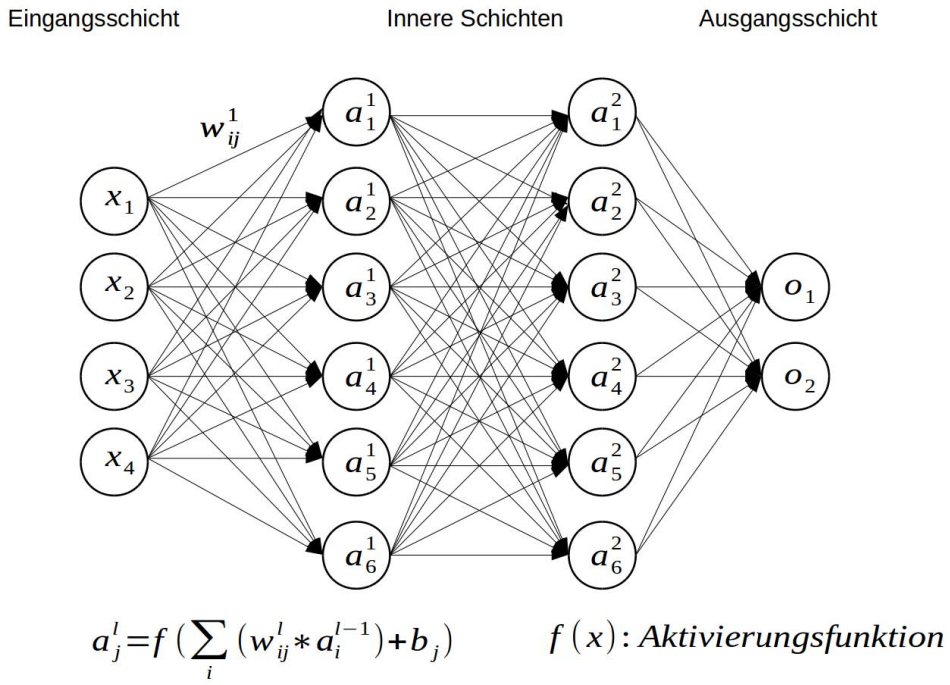

Abbildung 4.14: Schematische Darstellung des Aufbaues eines mehrlagigen (hier 3-lagig) Neuronalen Netzwerk – eines sogenannten fully-connected-networks – bei dem alle Neuronen benachbarter Schichten miteinander verbunden sind.

Transformiert man alle Pixel eines Bildes zu einem langen Vektor, könnte man auch ein solches normales Neuronales Netz für die Bildklassifizierung nutzen. Allerdings ergeben sich hierbei zwei Probleme:
Zum einen skaliert ein normales Neuronales Netzwerk schlecht. Nimmt man zum Beispiel ein Bild mit den Dimensionen 32 x 32 x 3 (Höhe, Breite und die drei Farbkanäle rot, grün und blau) und transformiert dieses in einen Vektor, hätte dieser die Länge 3072. Ein Neuron des ersten inneren Layers hätte somit 3072 zu optimierende Gewichte (eins für jede Verbindung zu einem der 3072 Input Neuronen). Die inneren Layer bestehen jedoch zumeist nicht nur aus einem Neuron, sondern aus vielen, genauso wie ein Neuronales Netzwerk meist nicht nur aus einem dieser Layer besteht, sondern aus vielen hintereinander gereihten Layern. Dadurch steigt die Anzahl der zu optimierenden Parametern schnell in riesige Dimensionen.

Zwar mag dies für ein so kleines Bild (32 x 32 x 3) noch möglich sein. Sind die Input-Bilder allerdings größer, zum Beispiel 256 x 256 x 3 (ungefähre Größenordnung der ImageNet Bilder), so hätte jedes Neuron des ersten inneren Layers bereits 196.608 Gewichte. Multipliziert man dies mit der Anzahl an Neuronen im ersten Layer und bedenkt, dass ein Neuronales Netzwerk aus vielen solcher Layer aufgebaut ist, erkennt man schnell das Skalierungsproblem normaler Neuronaler Netzwerke.
Das zweite Problem ist die Sensitivität gegenüber Verschiebungen im Bild. Normale Neuronale Netze machen keinen Unterschied zwischen Pixeln im Bild die weit voneinander entfernt sind und solchen in unmittelbarer Nähe zueinander. Jedes Pixel ist eine Zahl im Input-Vektor, die nicht in Beziehung zu ihren benachbarten Elementen steht. Aus diesem Grund gleicht jede auch noch so kleine Verschiebung im Bild einem ganz neuen Input-Bild. Für ein binäres Bild der Dimension 28 x 28 Pixel gibt es alleine 2^(28*28 = 2^784 unterschiedliche Möglichkeiten an Bildern. Das sind mehr unterschiedliche Bilder als die geschätzte Anzahl an Atomen im Universum (~ 10^80).
ConvNets hingegen sind speziell auf Bilder als Netzwerk-Input angepasst und sorgen durch ihre Architektur dafür, dass die Anzahl an zu optimierenden Parametern drastisch reduziert werden kann. ConvNets sind außerdem invariant gegenüber Verschiebungen von Objekten innerhalb des Bildes. Das Bild bleibt in seiner Struktur

unverändert, ist also sozusagen eine dreidimensionale Matrix (Bildhöhe x Bildbreite x 3 für die Farbkanäle rot, grün und blau).
Die zu optimierenden/lernenden (aus dem Englischen für „learnable parameters“) Parameter eines Convolutional-Layers werden Filter genannt.
Filter sind, ebenso wie die Bilder, dreidimensionale Matrizen, deren Höhe und Breite wählbare Parameter sind. Die dritte Dimension, die Tiefe, ist nicht wählbar sondern richtet sich nach der Anzahl der Kanäle des Input-Bildes (3 für ein normales Farbbild) oder, in tieferen Layern des ConvNets, nach der Anzahl der Aktivierungs-Maps des vorherigen Layers. Aktivierungs-Maps entsprechen den Neuronen eines Neuronalen Netzwerkes und sind ebenso wie die Filter Matrizen.
Für die Höhe und Breite werden zumeist kleine Werte, typischerweise um die 5 Pixel, gewählt. Bei einer Vorwärtsrechnung wird jeder Filter über alle möglichen Positionen des vorherigen Layers geschoben und an jeder Position wird zwischen dem kleinen durch den Filter überlagerten Bildausschnitt und dem Filter selbst das Skalarprodukt berechnet. Das Ergebnis des Verschiebens (genauer gesagt der Faltung = convolution) eines Filters über den gesamten Bereich des vorherigen Layers erzeugt dabei eine zweidimensionale Matrix, die Feature Map genannt wird. Zur Veranschaulichung zeigt Abbildung 4.15 diesen Vorgang für ein zweidimensionales Bild und zwei unterschiedliche Filter. Das Input-Bild auf der linken Seite ist 7 x 7 Pixel groß, die Filter besitzen 3 x 3 Pixel. Die Bildung des Skalarproduktes zwischen Bildausschnitt und Filter über alle möglichen Positionierungen des Filters im Bild ergibt als Resultat eine 5 x 5 Pixel große Feature Map (rechts im Bild). Beispielhaft sind zwei Zahlen in den Feature Maps umrahmt. Die Linien verdeutlichen aus welchem Bildbereich sich diese durch Kombination mit dem Filter errechnen.

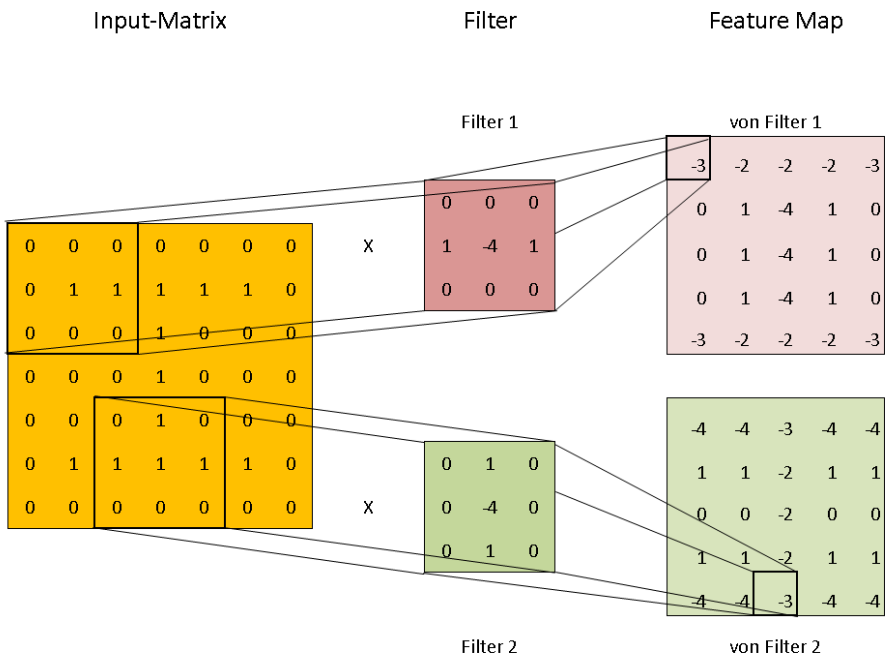

Abbildung 4.15: Beispiel einer Faltung (convolution) eines 2-D Input-Bildes (gelb) mit zwei Filtern der Größe 3 x 3. Rechts im Bild sind die resultierenden Feature Maps gezeigt, die durch das Verschieben des Filters über das gesamte Bild und die Berechnung des Skalarproduktes mit dem Bildausschnitt erzeugt werden.

Im Anschluss an die Faltung wird eine (meist nicht-lineare) Aktivierungsfunktion auf diese Feature Maps angewendet. Eine Aktivierungsfunktion kann als eine Art Transferfunktion angesehen werden. Im Falle des biologisch inspirierten Neuronalen Netzwerkes wurden in der Vergangenheit oft sigmoidale Funktionen verwendet, deren Ergebnis als das „aktiv“ oder „nicht-aktiv“-Sein eines Neurons interpretiert werden kann.

Im Falle der ConvNets wird zumeist die ReLU-Funktion (Rectified Linear Unit siehe Abbildung 4.16) verwendet (LeCun, Yann, Bengio and Hinton, 2015), da diese einige Vorzüge gegenüber den klassischen Aktivierungsfunktionen (z.b. Sigmoid-Funktion oder Tangens Hyperbolicus), wie eine schnellere Berechnung und die Erhaltung des Fehlersignals beim Rückwärtspass während des Trainings, besitzt (Krizhevsky et al., 2012).

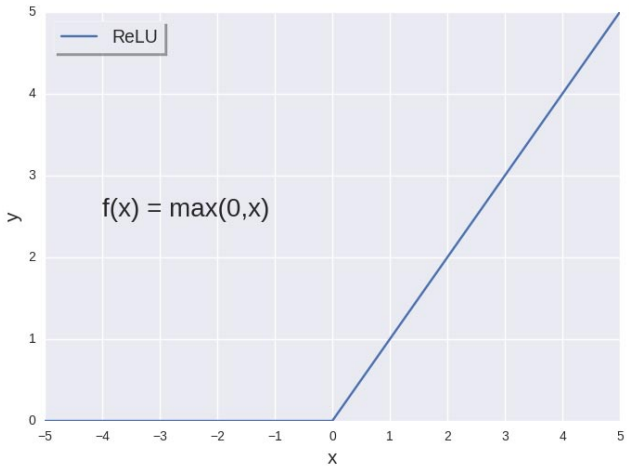

Abbildung 4.16: Verwendete nicht-lineare Aktivierungsfunktion ReLU (Rectified Liniear Unit)

Das Resultat aus der Anwendung der Aktivierungsfunktion auf die Einträge der Feature Maps wird Aktivierungs-Maps genannt. Diese werden, ähnlich der Struktur eines Bildes, zu einer dreidimensionalen Matrix zusammengefasst (siehe Abbildung 4.17).

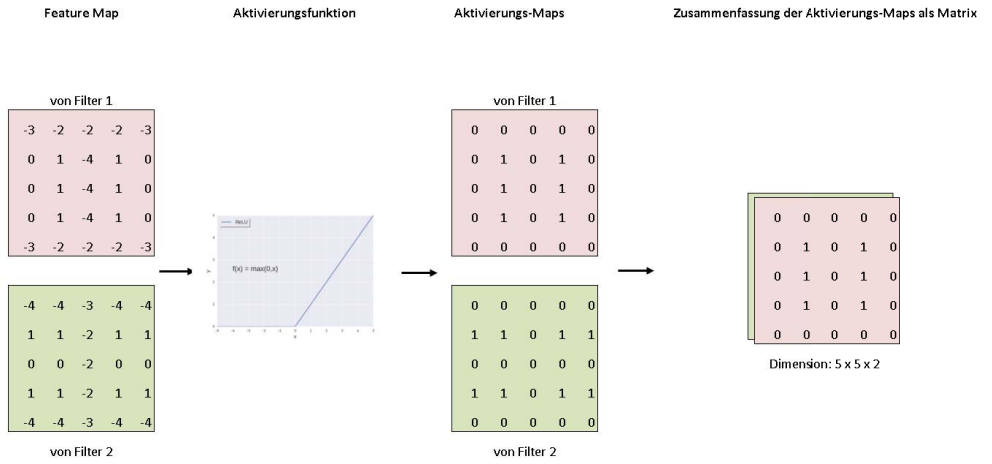

Abbildung 4.17: Transformation der Feature Maps zu den Aktivierungs-Maps durch elementweise Anwendung der Aktivierungsfunktion. Die resultierenden Aktivierungs-Maps werden zu einer dreidimensionalen Matrix zusammengefasst (in diesem Fall mit den Dimensionen 5 x 5 x 2).

Die Filter des nachfolgenden Convolutional-Layers hätten somit die Tiefe 2 bei erneut frei wählbarer Breite und Höhe.

Abbildung 4.18 zeigt beispielhaft den Aufbau für ein ConvNet mit mehreren hintereinander gereihten Convolutional Layern und einem Farbbild als Input (mit den drei Farbkanälen rot, grün und blau). Der erste Layer besteht aus 96 unterschiedlichen Filtern mit den Dimensionen h1 x b1 x 3 (h1 und b1 sind frei wählbar). Die Faltung der 96 Filter mit dem Input-Bild resultiert in 96 unterschiedlichen Feature Maps (bzw. Aktivierungs-Maps, nach Anwendung der Aktivierungsfunktion). Zusammengefasst zu einer dreidimensionalen Matrix, ergeben diese 96 Aktivierungs-Maps der einzelnen Filter eine Matrix mit der Tiefe 96. Die dritte Dimension der Filter des darauffolgenden Layers ist somit auf 96 festgelegt, da ein Filter durch die gesamte Tiefe der vorhergehenden Matrix schaut (in diesem Fall die 96 Aktivierungs-Maps des ersten Layers, die zu einer dreidimensionalen Matrix zusammen gefasst sind). Ebenso folgt für die Filter der folgenden Layer, dass sich deren Tiefe stets an der Tiefe der Matrix der vorhergegangenen Aktivierungs-Maps richtet.

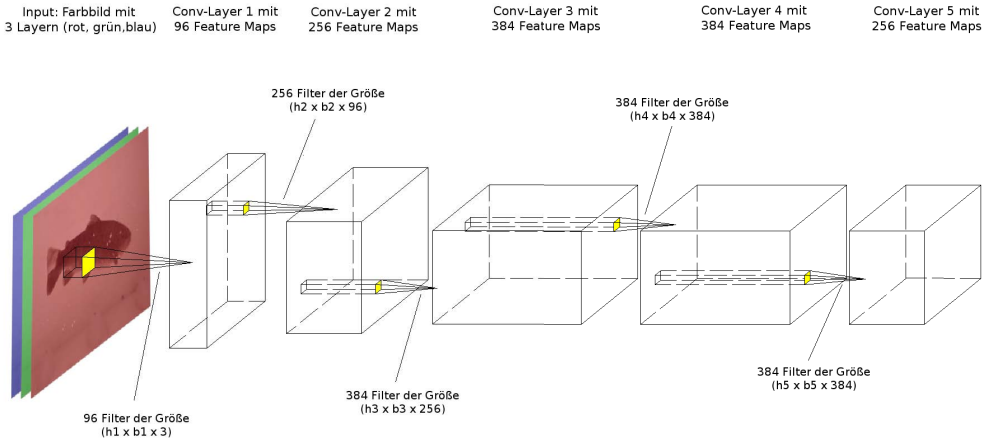

Abbildung 4.18: Aufbau eines ConvNets mit mehreren hintereinander gereihten Convolutional-Layern und verschiedenen Anzahlen an Filtern. Aus der Anzahl an Filtern des vorherigen Layers und der daraus resultierenden Anzahl an Feature-Maps ergibt sich die Tiefe der Filter des nächsten Layers. Die gelben Bereiche symbolisieren die Größe der Filter und somit deren Sichtbereich.

Die gelernten Filter können als eine Art Feature-Detektoren angesehen werden. Da die Filter des ersten Convolutional-Layers ihrerseits von den Dimensionen als kleine

Bilder angesehen werden können (Filterhöhe, -breite und 3 Kanäle tief), lassen sich die gelernten Filter direkt wie ein Bild betrachten. Abbildung 4.19 zeigt die 96 Filter des ersten Convolutional-Layers des verwendeten AlexNets. Die Filter ähneln sogenannten Gabor-Filtern. Sie detektieren verschieden ausgerichtete Kanten im Bild aufgrund von Intensitäts- oder Farbwechseln.

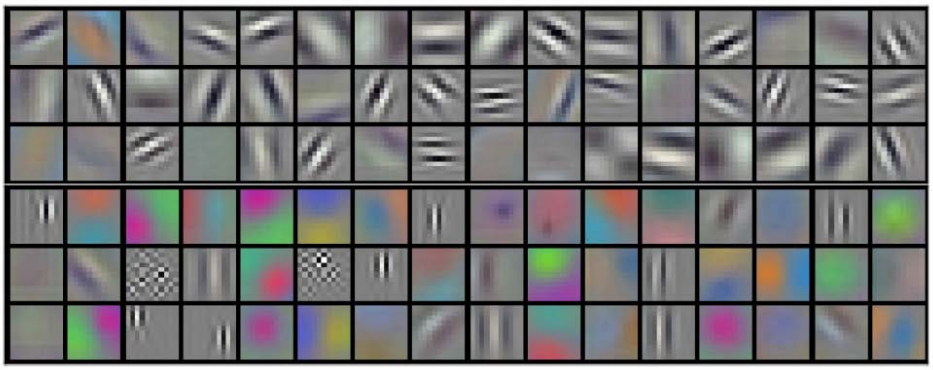

Abbildung 4.19: Visualisierung der 96 Filter des ersten Convolutional Layers des verwendeten AlexNets. Abbildung aus Krizhevsky et al. (2012).

Betrachtet man die Aktivierungs-Maps, die von einzelnen Filtern erzeugt werden, so kann man besser verstehen auf was ein einzelner Filter reagiert. Abbildung 4.20 zeigt beispielhaft das Ergebnis der Faltung von zwei ausgewählten Filtern mit einem Input-Bild. Im oberen Filter ist ein vertikaler Kontrastübergang zu sehen, im unteren ein horizontaler Kontrastübergang. Alle Bildausschnitte, die dem jeweiligen Filter ähneln, ergeben bei der Bildung des Skalarproduktes einen positiven Wert und sind rechts in den Feature Maps als helle Bereiche erkennbar.

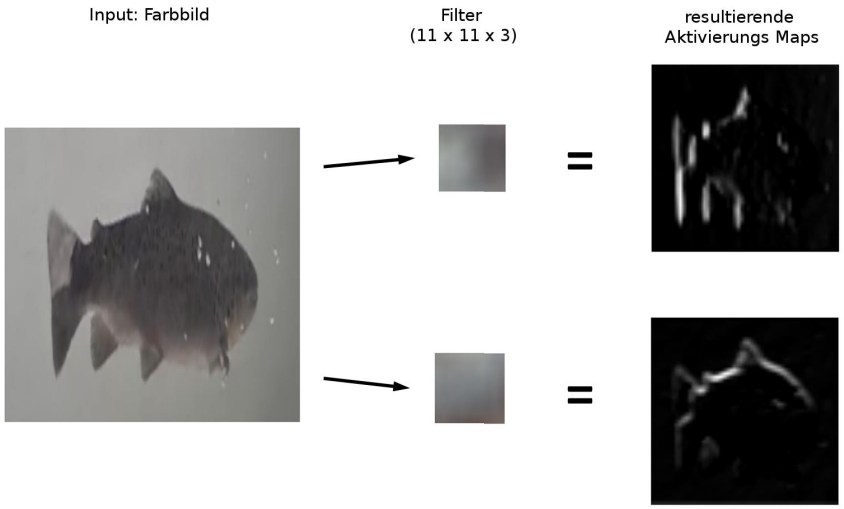

Abbildung 4.20: Visualisierung der Aktivierungs-Maps für zwei ausgewählte Filter und dem entsprechenden Input-Bild. Wie im oberen Filter zu erkennen, sucht dieser nach vertikalen Kontrastübergänge, der untere entsprechend nach horizontalen Kontrastübergängen. Die Bildausschnitten, in denen der Kontrastübergang von hell zu dunkel ähnlich dem Filter ist, sind in der Aktivierungs-Map als helle Bereiche erkennbar.

Die Features, die ein ConvNet mit seinen Filtern erkennt, werden von Layer zu Layer komplexer. Im ersten Layer sind hauptsächlich Kanten-Detektoren zu finden. Die Feature Maps enthalten somit Informationen über Objektkanten, Farbübergänge etc. Die Filter des zweiten Layers, die durch all diese Feature Maps hindurchschauen können, können die Informationen zu komplexeren Formen zusammensetzen (wie z.B. unterschiedliche Texturen, geometrische Figuren wie Kreise, etc.). Die darauf folgenden Layer können aus diesen einfachen Formen und Strukturen ihrerseits noch komplexere Formen zusammensetzen, so dass Filter allgemein mit größerer Tiefe des Netzwerkes immer komplexere und detailreichere Repräsentationen von Objekten des Trainingsdatensatzes erkennen. In Abbildung 4.21 ist die zunehmende Komplexität der erlernten Filter anhand von Gesichtern veranschaulicht.

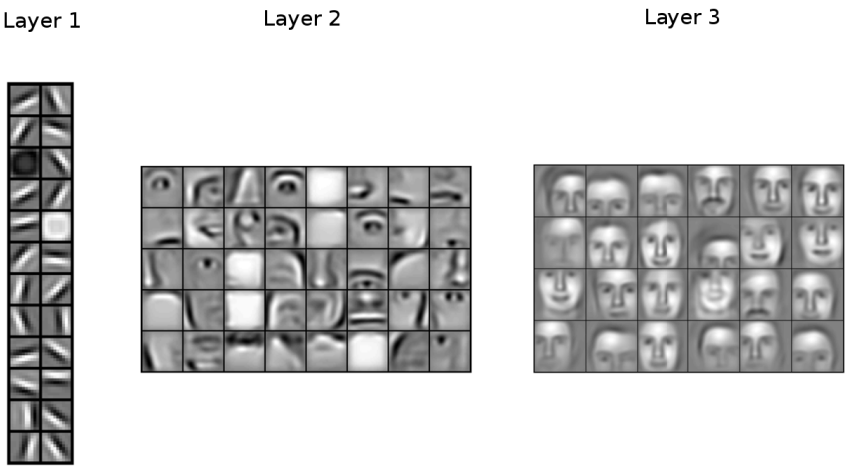

Abbildung 4.21: Veranschaulichung der mit zunehmender Tiefe der Convolutional-Layer größer werdenden Komplexität der gelernten Filter am Beispiel von Gesichtern. Suchen Filter des ersten Layers rein nach Kontrastübergängen, erkennen Filter des zweiten Layers bereits einfache Formen wie Augen und Nasen. Die Filter des dritten Layers können diese Informationen schon zu relativ komplexen Gesichtern zusammensetzen. Abbildungen aus Lee et al. (2009).

Durch das Lernen dieser Filtern, die unverändert über das gesamte Volumen des vorherigen Layers gefaltet (= *convolved*) werden, kann die Anzahl der zu lernenden Parameter im ConvNet gegenüber einem normalen Neuronalen Netzwerk drastisch reduziert werden.

Als Beispiel der oben bereits genannte Fall eines Input-Bildes mit den Dimensionen 256 x 256 x 3 Pixeln:

- In einem normalen voll-verknüpften Neuronalen Netzwerk wäre jedes Neuron des ersten inneren Layers mit allen 196.608 (256*256*3) Pixeln verknüpft und jede dieser Verknüpfungen hätte ein zu optimierendes Gewicht. Schon alleine bei 100 Neuronen im ersten Layer (einer durchaus kleinen Zahl) besäße der erste Layer über 19 Millionen zu lernende Parameter.
- Ein ConvNet mit einem ersten inneren Layer mit 96 Filtern der Größe 11 x 11 x 3 (wie im verwendeten AlexNet benutzt), besitzt 34.848 (= 96*11*11*3) zu lernenden Parameter im ersten Layer. Die Anzahl ist zudem unabhängig von der Größe des Input-Bildes.

Der Grund, warum ConvNets zusätzlich invariant gegenüber räumlichen Verschiebungen von Objekten sind, ist das sogenannte *Max-Pooling* (Zusammenfassung der Maxima).

Beim Max-Pooling wird mit einem manuell definierten Fenster (in der Regel 2 x 2) mit einer festgelegten Schrittweite über die Aktivierungs-Maps gefahren und das Maximum aller Werte innerhalb des Fensters als neuer Wert übernommen. Wie in Abbildung 4.22 dargestellt, reduziert dies die Größe der Feature Map und sorgt auch dafür, dass Verschiebungen im Bild und in den Aktivierungs-Maps durch das Pooling egalisiert werden.

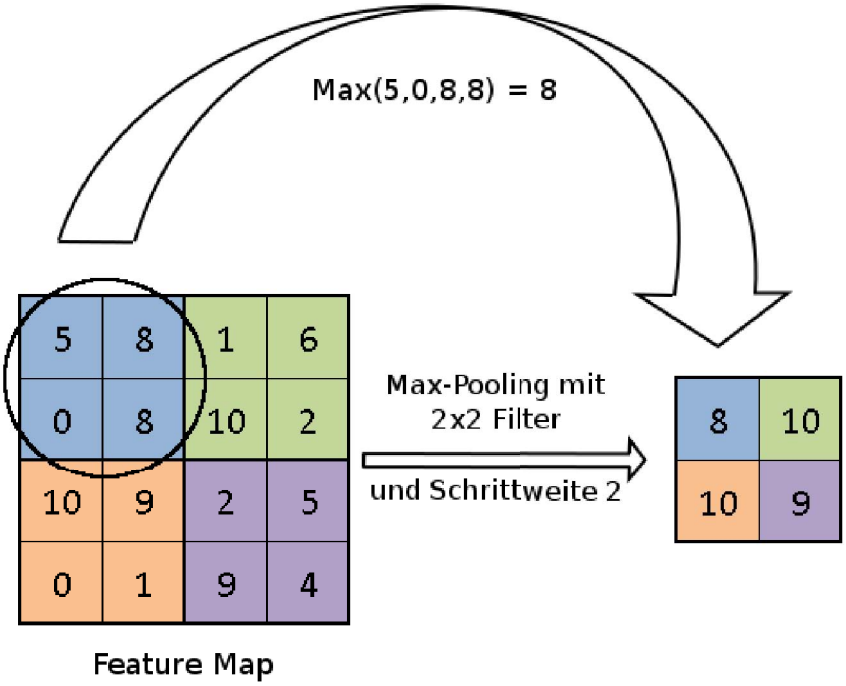

Abbildung 4.22: Veranschaulichung des Max-Poolings anhand einer 4 x 4 großen Aktivierungs-Map (links). Das Ergebnis eines 2 x 2 Filters und Schrittgröße 2 ist auf der rechten Seite zu sehen.

Die Architektur des verwendeten ConvNets ist in Abbildung 4.23 zu sehen und entspricht dem Netzwerk des ImageNet-Gewinners von 2012 (bzw. einer leicht angepassten Version, siehe Abschnitt 4.1.3.2).

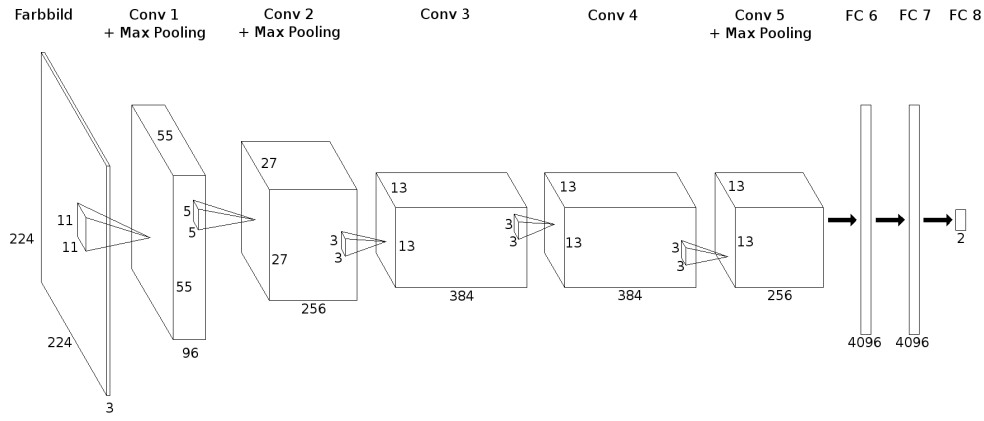

Abbildung 4.23: Verwendete Architektur des in FishNet verwendeten ConvNets. Ist eine angepasst Version des AlexNets (Krizhevsky et al., 2012). Die Zahlen an den Volumen kennzeichnen die Anzahl und Größe der Aktivierungs-Maps. Die Filtergröße jedes Layers ist ebenfalls gekennzeichnet. Unter den fully-connected Layern ist die Anzahl an Neuronen pro Layer angegeben.

4.1.3.2 Anpassung und Training des Convolutional Neural Networks

Auch wenn ein ConvNet weniger zu lernende Parameter enthält als ein vergleichbares normales Neuronales Netz, ist das Training eines neu initialisierten Netzwerkes sehr zeitintensiv und benötigt eine sehr große Anzahl an Trainingsbeispielen (Trainingszeiten von 1-2 Wochen auf mehreren Grafikkarten sind nicht ungewöhnlich und die Größe des ImageNet-Datensatzes besteht z.B. aus 1.5 Millionen Bildern). In der Praxis wird deswegen häufig ein, an einem großen Datensatz trainiertes ConvNet als Basis herangezogen und an die spezielle Fragestellung angepasst. Dies ist möglich, da sich gezeigt hat, dass diese bereits trainierten Netzwerke als nützliche Feature-Extraktoren fungieren und sich anhand dieser extrahierten Features neue Klassifikatoren trainieren lassen (Donahue et al., 2014; Razavian et al., 2014; Yosinski et al., 2014).

Für FishNet wurde das sogenannte *fine-tuning* (= Feinabstimmung) angewendet. Dabei werden einige der letzten Layer (je nach Größe des verfügbaren Datensatzes) neu initialisiert und für die restlichen Layer die Gewichte (Filter) eines bereits trainierten Netzwerkes verwendet. Der Idee ist, dass (wie oben beschrieben) die Filter der ersten Layer eines ConvNet nur grobe Strukturen erfassen und diese für viele

Datensätze relativ ähnlich sind (wie z.B. die Kanten-Detektoren etc.). Im diesem speziellen Fall wurden die letzten drei Layer (FC 6 bis FC 8) neu trainiert und die Convolutional-Layer vom bereits trainierten AlexNet – in der Version von Caffe (Jia et al., 2014)- übernommen. Der letzte Layer wurde zudem in seiner Größe von 1000 auf 2 reduziert, da es in diesem Fall nur zwei Klassen zu unterschieden gibt (Fisch/kein Fisch).

4.1.3.3 Erzielte Genauigkeit

Da die im Rahmen dieser Masterarbeit entwickelte Software in allen Versionen das beste Bild (nach Abschnitt 3.3) jedes Fisches in einem Video abspeichert, standen zum Zeitpunkt des Umstiegs eine große Anzahl an Bildern als Trainingsdaten zur Verfügung. Es wurde ein Datensatz aus circa 26.000 Bildern pro Klasse aus allen bisherigen Datensätzen (Flüssen, sowie Zeiträumen) als Trainingsdatensatz zusammengestellt. Des Weiteren wurde der Trainingsdatensatz durch horizontale Spiegelung aller Bilder künstlich vergrößert (insgesamt somit ca. 100.000 Bilder). Andere übliche Verfahren zur Vergrößerung des Trainingsdatensatzes, wie Rotation, Verschiebung (Ciregan et al., 2012) oder leichte Veränderung der Pixelwerte (Krizhevsky et al., 2012), wurden nicht angewendet. Zum einen, da der Trainingsdatensatz bereits relativ groß war (für das fine-tuning), und zum anderen, da durch vorhergegangene Schritte der Software sichergestellt ist, dass die Objekte immer gleich ausgerichtet und zentriert sind. Des Weiteren wurde keine Unterscheidung zwischen trübem und klarem Wasser getroffen und nur ein einzelner Klassifikator trainiert.
Von den rund 100.000 Bildern wurden 80 % für das Training des ConvNets verwendet, 10 % für Validierung während des Trainings und 10 % zum Testen des trainierten Netzwerkes. Das Ergebnis der Auswertung ist in Tabelle 4.4 zu sehen. Die Genauigkeit der Klassifizierung liegt bei rund 97 %.

		Vorhersage	
		Fisch	Kein Fisch
Wahre Klasse	Fisch	4962	247
	Kein Fisch	25	5183

Tabelle 4.4: Konfusionsmatrix der Testdaten-Auswertung des ConvNets

Dies ist ein sehr zufriedenstellender Wert, auch wenn die Rate der falsch klassifizierten Fische (als „kein Fisch“ klassifiziert) wesentlich höher ist, als die der „kein Fisch“-Objekte (als „Fisch“ klassifiziert). Die Sensitivität (oder auch Richtig-Positiv-Rate bzw. Trefferquote genannt) liegt immerhin noch bei 95,5 %. In Abbildung 4.24 sind einige Beispiele von falsch klassifizierten Fischen des Testdatensatzes gezeigt.

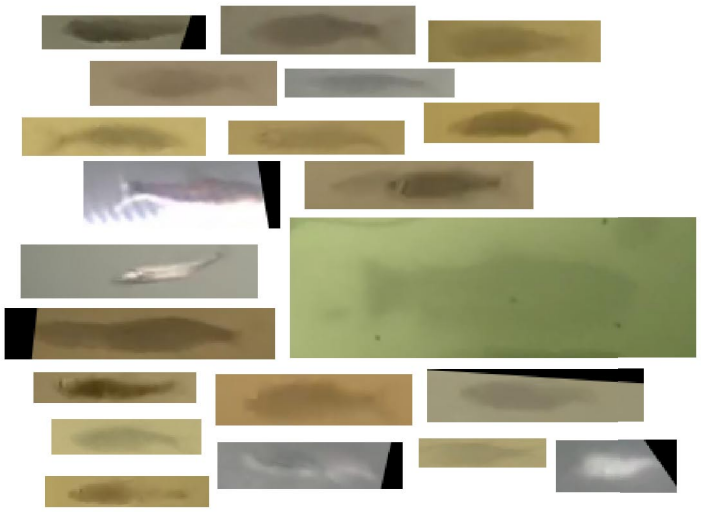

Abbildung 4.24: Beispiele falsch klassifizierter Fische des Testdatensatzes.

Die Analyse der Fehlklassifizierungen zeigt, dass diese am häufigsten bei kleinen Fischen in trübem Wasser auftreten.

4.2 Bestimmung der Wanderrichtung

Die automatisierte Bestimmung der Wanderrichtung ist relativ leicht, vorausgesetzt, dass einige Bedingungen erfüllt sind:

1. Das Video beginnt vor oder in dem Moment, in dem der Fisch in den Erfassungstunnel schwimmt.
2. Das Video endet erst, nachdem der Fisch den Erfassungstunnel wieder verlassen hat.

3. Der Fisch wurde über seinen gesamten Aufenthalt vor der Kamera korrekt verfolgt.

Treffen alle Bedingungen zu, reicht es, wenn man die Position des ersten Auftretens eines Fisches mit der des letzten vergleicht. Über die Bildkoordinaten lässt sich so berechnen, ob ein Fisch von rechts nach links, links nach rechts oder nur im Kreis geschwommen ist. Abbildung 4.25 zeigt so einen Fall beispielhaft, in dem unschwer zu erkennen ist, dass der Fisch links ins Bild geschwommen ist und den Erfassungstunnel auf der rechten Seite verlässt.

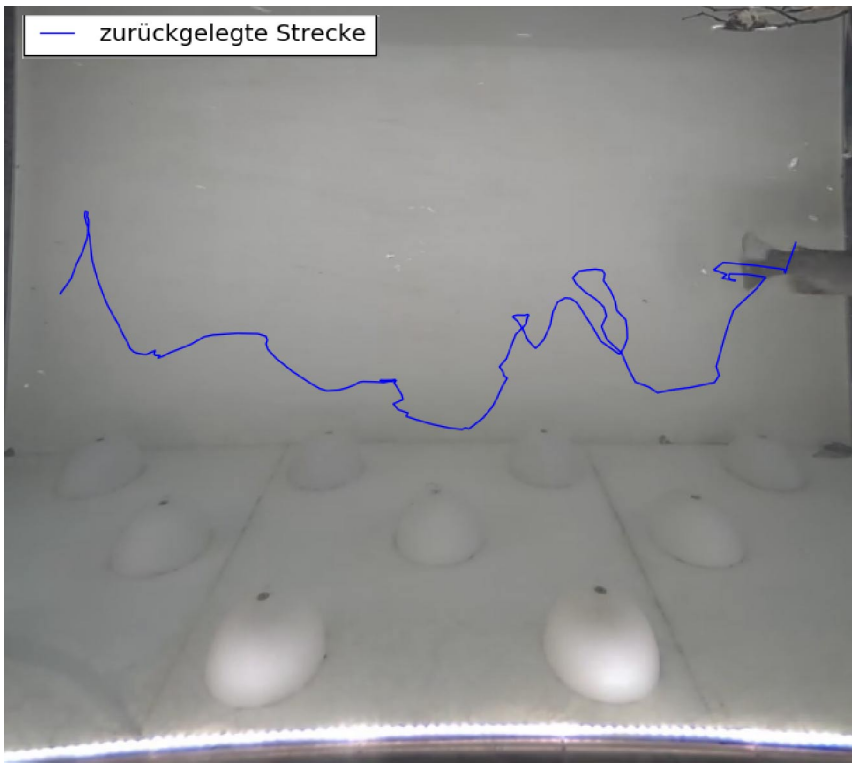

Abbildung 4.25: Beispiel für einen Fisch im Moment des Verlassens des Erfassungstunnels mit eingezeichneter Schwimm-Strecke. Wanderrichtung: links nach rechts.

Da ein Fisch, der auf der gleichen Seite hinein wie hinaus schwimmt, seinen ersten und letzten Massenschwerpunkt ziemlich wahrscheinlich nicht auf der gleichen Koordinate der x-Achse hat, sondern diese leicht gegeneinander verschobene sind (siehe Abbildung 4.26), benötigt das Verfahren zur Bestimmung der Wanderrichtung einen Grenzwert für die Strecke, die ein Fisch mindestens geschwommen sein muss, um als auf- oder abgestiegener Fisch gezählt zu werden. Dieser Wert ist frei wählbar, sollte

jedoch 30 % der Bildbreite nicht unterschreiten.

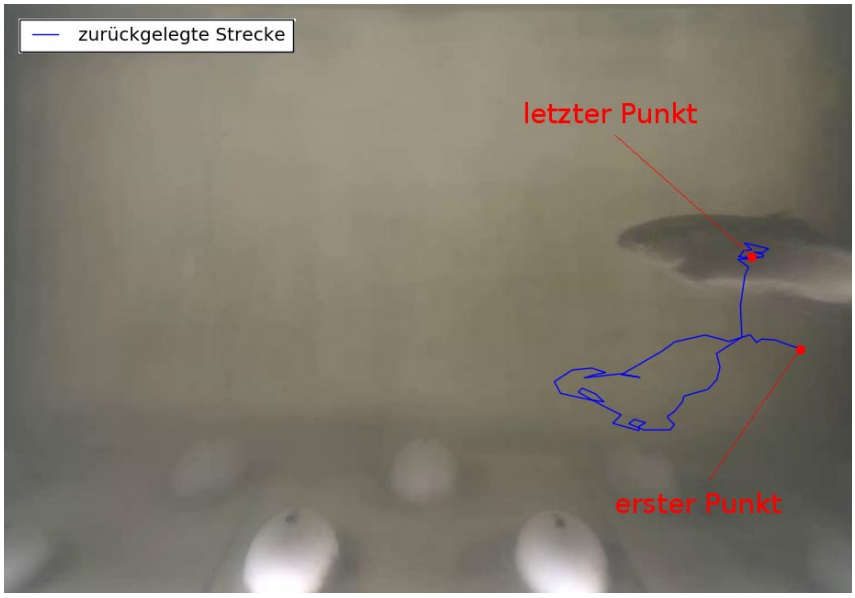

Abbildung 4.26: Zweites Beispiel für das letzte Bild des Fisches im Video. Neben der zurückgelegten Strecke (blaue Linie) sind auch der erste und letzte Punkt des Massenschwerpunktes gekennzeichnet.

Probleme hat dieses Verfahren, wenn die oben genannten Bedingungen nicht erfüllt sind. Treffen Bedingungen 1 und 2 (Video beginnt vor dem Einschwimmen des Fisches und endet erst nachdem der Fisch den Tunnel verlässt) nicht zu, ist es eventuell für den menschlichen Betrachter auf Grund des Schwimmverhaltens erkenntlich ob ein Fisch zielgerichtet auf- oder absteigt. Für ein Programm ist dies jedoch relativ schwer festzulegen. Aus diesem Grund ist es von großer Wichtigkeit für die entwickelte Software, dass die Videos unmittelbar vor dem Einschwimmen eines Fisches in den Erfassungstunnel beginnen und erst nach dem Verlassen stoppen.
Wie gut die Bedingung 3 (durchgängiges Tracking des Fisches) zutrifft hängt von mehreren Faktoren ab. Für Videos mit vereinzelten Fischindividuen funktioniert das Tracking sehr zuverlässig. Das korrekte Verfolgen einzelner Fische in Schwärmen funktioniert bisher noch nicht sehr zuverlässig, weswegen auch die automatisch bestimmte Wanderrichtung in Fall von Fischschwärmen oft falsch ist. Weitere Überlegungen zur Verbesserung des Trackings und somit der Bestimmung der Wanderrichtung werden in Abschnitt 6.2 diskutiert.

5 Ergebnisse

5.1 Software FishNet

Im Rahmen dieser Arbeit konnte ein Verfahren zur automatisierten Objekterkennung in Videos des FishCam-Systems erfolgreich entwickelt und umgesetzt werden. Die entwickelte Software FishNet erkennt zuverlässig Objekte in Videos und klassifiziert diese als „Fisch“ oder „kein Fisch“. Des Weiteren wurde ein Verfahren zur automatisierten Bestimmung der Wanderrichtung entwickelt. Für die benutzerfreundliche Verwendung von FishNet wurde außerdem eine grafische Benutzeroberfläche entworfen (siehe Abbildung 5.1), über welche sich die Videoauswertung konfigurieren und starten lässt.
Verpflichtend auszuwählen sind der Ordner, der die auszuwertenden Videos enthält und die Fischaufstiegshilfe, aus der die Videos stammen. Außerdem lassen sich einige Optionen für die aktuelle Videoauswertung einstellen, die jedoch hauptsächlich zu Test- und Entwicklungszwecken eingebaut sind. In der in Abbildung 5.1 gezeigten Konfiguration erstellt FishNet zu Beginn der Auswertung eine Ordnerstruktur, in welche die Videos gemäß dem Ergebnis der Objektklassifizierung und der Bestimmung der Wanderrichtung einsortiert werden. In diese Ordner wird zusätzlich von jedem Objekt der Videos das „beste Bild des Fisches“ (nach Abschnitt 3.3) abgespeichert. Dadurch lässt sich durch schnelle visuelle Kontrolle der Bilder die Richtigkeit der Klassifizierung im Nachhinein überprüfen.

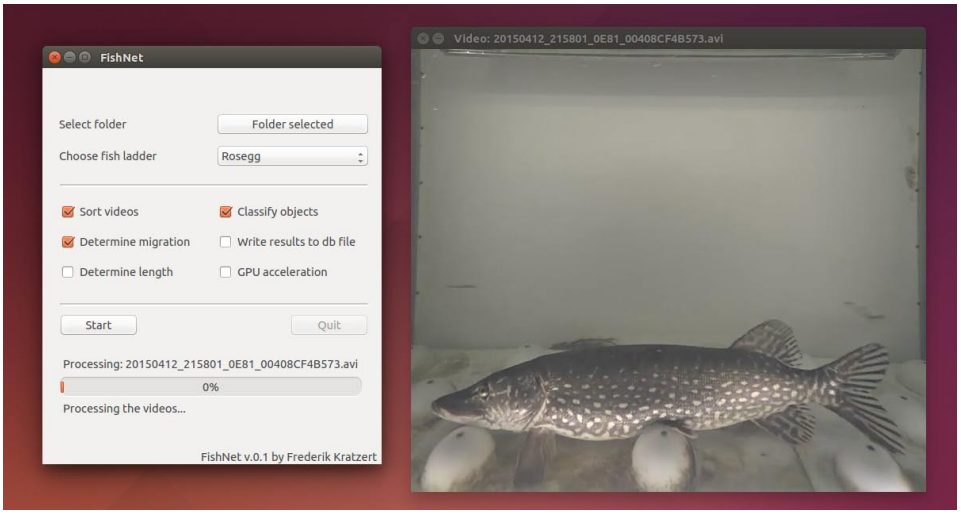

Abbildung 5.1: Ansicht der Benutzeroberfläche von FishNet während der Auswertung eines Videos. Auf der linken Seite ist das Kontrollfenster zur Auswahl der Videos, des Fischaufstieges sowie den möglichen Einstellungen zu sehen. In dem Fenster auf der rechten Seite läuft das analysierte Video mit.

Das Auswählen der Fischaufstiegshilfe ist nötig, da alle Kameras leicht unterschiedlich eingebaut sind. Deshalb variiert der Bildausschnitt, der vom gesamten Kamerabild bei der Auswertung betrachtet wird (siehe Kapitel 3), von Fischaufstiegshilfe zu Fischaufstiegshilfe. Da dies so ist, muss für jeden Fischaufstieg einmalig manuell ausgewählt werden, welcher Bereich für die Auswertung zu betrachten ist. Damit dies für neu hinzukommende Fischaufstiege einfach und benutzerfreundlich möglich ist, wurde eine weitere grafische Oberfläche entwickelt, über die das Hinzufügen neuer Fischaufstiegshilfen interaktiv möglich ist (siehe Abbildung 5.2).

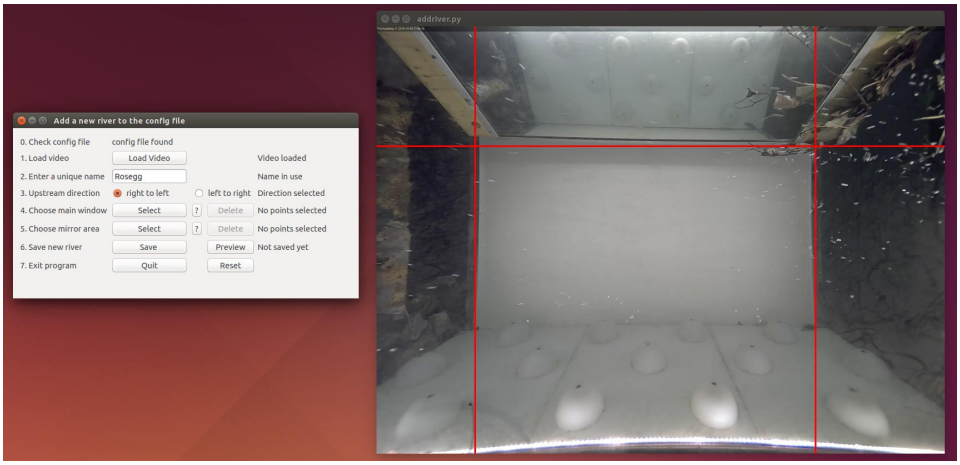

Abbildung 5.2: Ansicht der Benutzeroberfläche zum Hinzufügen neuer Fischaufstiegshilfen. Das Programm überprüft automatisch die korrekte Eingabe aller Werte und ob der Name bereits in der Config-Datei vorhanden ist. Der zu betrachtende Bereich lässt sich per Mausklick auswählen.

Nach anfänglicher Entwicklung in MATLAB ist die aktuelle Version von FishNet komplett in Python 2.7 geschrieben. Die wichtigsten Zusatzpakete sind: Numpy, OpenCV, SciKit-Image, Pandas, Caffe und PyQt4.

Die Auswertungsgeschwindigkeit der Videos mit FishNet liegt circa bei Echtzeit (rund 12 Bilder/Sekunde auf einem Intel i7-4790 @ 3.6 GHz), wobei das Programm lediglich einen Prozessor beansprucht. Bei Vorhandensein von mehreren Prozessoren kann das Programm mehrfach gestartet werden und ohne Beeinträchtigung der Auswertungsgeschwindigkeit mehrfach parallel laufen.

FishNet läuft, bei korrekter Installation von Python 2.7 und allen benötigten Paketen unter Linux und Windows. Mac OS wurde nicht getestet.

Da Python eine open source Programmiersprache ist und auch alle Pakete zur freien Nutzung lizenziert sind, kann FishNet auch für die kommerzielle Nutzung verwendet werden.

5.2 Anwendung von FishNet

Bereits seit gut 3 Jahren sind FishCam-Systeme zur videobasierten Überwachung von Fischaufstiegshilfen (FAH) im Einsatz. Fast genauso lange läuft die Computer-unterstützte Videoauswertung, auch wenn die automatisierte Objekterkennung, wie im Rahmen dieser Masterarbeit vorgestellt, erst seit ca. 2 Jahren in den unterschiedlichen Entwicklungsstufen genutzt wird. Mittlerweile existieren 14 FishCam-Systeme die parallel in verschiedenen FAH zu Einsatz kommen. Abbildung 5.3 zeigt die unterschiedlichen Standorte an denen in der Vergangenheit (rot), im Moment (grün) oder bereits für die Zukunft fest geplant (magenta) FishCam-Systeme zur Überwachung von FAH eingesetzt werden. Der Großteil der überwachten FAH befinden sich in Kärnten an der Drau (Kraftwerk Kellerberg, Lavamünd, Paternion, Rosegg und Schwabeck) bzw. in Salzburg an der Salzach (Kraftwerk Bischhofshofen, Sohlstufe Lehen, St. Johann, Werfen-Pfarrwerfen und Urreiting).

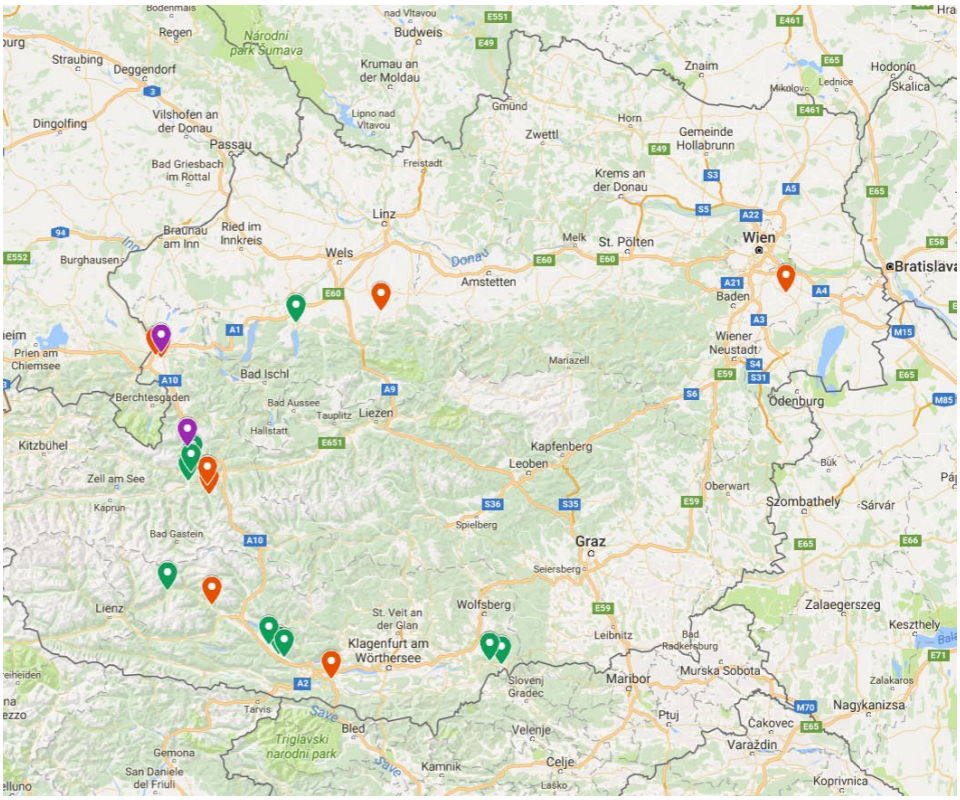

Abbildung 5.3: Übersichtskarte der FishCam-Standorte von abgeschlossenen (rot), laufenden (grün) und bald beginnenden (magenta) Monitoringprojekten.

Die Summe aller Monate und aller Anlagen die mit FishCam-Systemen überwacht wurden, beläuft sich mittlerweile auf über 120 Daten-Monate. Dabei wurden 1,3 Millionen Videos mit insgesamt über 10 Terabyte Videomaterial aufgenommen und ausgewertet. Der Anteil der Videos, die zumindest einen Fisch enthalten, beläuft sich auf rund 4,5 % und entspricht rund 6 % der Speichergröße (siehe Abbildung 5.4).

Übersicht über die Gesamtdaten (Stand August 2016)

Anzahl der Videos ohne Fisch/e	Größe in [GB]	Videos mit Fisch/en	Größe in [GB]
1.263.999	9.494,70	59.898	601,05

Gesamtzahl d. Videos:	1.323.897
Gesamtgröße aller Videos in [GB]:	10.095,75
Anteil d. Videos mit Fisch	4,52 [%]
Anteil d. Speichergröße d. Fisch-Videos:	5,95 [%]

Abbildung 5.4: Auflistung der genauen Zahlen an ausgewerteten Videos.

Diese Zahlen sind ein Beleg dafür, dass eine Auswertung der Monitoringdaten ohne FishNet eine kaum zu bewältigende Aufgabe wäre. Die automatisierte Vorsortierung der Daten reduziert den manuellen Arbeitsaufwand dermaßen, dass lediglich der geringe Anteil an Videos mit zumindest einem Fisch für die Arten- und Längenbestimmung gesichtet werden muss.

6 Ausblick

Das Ziel des Forschungsprojektes der VERBUND Hydropower AG ist, das gesamte Monitoring von FAH so weit wie möglich zu automatisieren. Ziel der Masterarbeit war lediglich die automatisierte Objekterkennung, trotzdem soll im weiteren Projektverlauf versucht werden auch die restlichen Aufgaben des Monitorings (Längen- und Artenbestimmung) zu automatisieren. Des Weiteren könnte die Genauigkeit der Auswertung, vor allem der korrekten Zählung der Auf- und Abstiege, in der Zukunft verbessert und das im Rahmen dieser Masterarbeit entworfene Verfahren überarbeitet werden. Dazu zählt in erster Linie die Verbesserung der Segmentierung von Hintergrund und Vordergrund, da diese die Grundlage für viele weitere Programmbausteine bildet. Für einige dieser Punkte gibt es schon konkrete Überlegungen, die im Folgenden kurz erläutert werden sollen.

6.1 Verbesserung der Objekterkennung

Wie bereits erwähnt, ist das korrekte erkennen der Objekte im Bild eine Voraussetzung für die Genauigkeit aller weiteren Schritte. Aus diesem Grund sollte hier nach Verbesserungsmöglichkeiten gesucht werden, die eventuell nicht rein pixelbasiert sind, sondern das gesamte Bild bzw. Bildregionen analysieren. Es sollte außerdem untersucht werden, ob andere, nicht-statische, Hintergrundmodelle, die die Veränderung von Bild zu Bild analysieren, eventuell besser funktionieren. Auch im Bereich der Bildsegmentierung gibt es mehr und mehr vielversprechende Ansätze mit ConvNets (Chen et al., 2015; Liu et al., 2015; Long et al., 2015), jedoch ist, auf Grund des hohen Hardwareanspruches bzw. der benötigten Berechnungszeit pro Bild, fraglich,

ob diese Methoden für die Auswertung so großer Datenmengen praktikabel sind. Auf jeden Fall sind weitere Recherchen und Untersuchungen im Bereich der Bildsegmentierung in Vorder- und Hintergrundbereiche notwendig.

6.2 Erweiterung des Tracking-Algorithmus

Das vorgestellte Verfahren zum Verfolgen von Objekten von Bild zu Bild klappt für den Fall einzelner oder weniger Fische im Bild sehr gut. Probleme treten allerdings bei Fischschwärmen auf, in denen sich Fische häufig im Bild überschneiden. In diesem Fall wird in der Vordergrundmaske nur ein Objekt erkannt, das beide Fische zusammenfasst. Einer der beiden Fische bekommt in diesem Fall kein (oder ein anderes) Objekt in der Vordergrundmaske zugewiesen. Dies kann automatisch korrigiert werden, wenn sich die Fische im Bild nicht mehr überschneiden und wieder zwei Objekte erkannt werden, muss es allerdings nicht. Yang et al. (2005) präsentieren eine Möglichkeit auch mit solchen Fällen umzugehen (siehe Abbildung 6.1). Bewegen sich zwei Objekte A und B aufeinander zu und überlagern sich ab einem gewissen Zeitpunkt, wird das eine verbleibende Objekt als Gruppe AB markiert und als solche für die nächsten Bilder des Videos verfolgt. Für die in der Gruppe enthaltenen Objekte A und B werden spezielle Features berechnet, damit im Falle der erneuten Trennung des Objektes AB in zwei einzelne Objekte diese wieder korrekt den Objekten A und B zugeordnet werden können. Zu dieser Methodik wurden bereits vielversprechende Untersuchungen während der Entwicklung der früheren Versionen von FishNet in MATLAB durchgeführt. Mit dem Umstieg auf Python wurden diese allerdings vorerst zurückgestellt, da das vorrangige Ziel die möglichst genaue Objektklassifizierung war und nicht die korrekte Zählung der auf- und absteigenden Fische. Das Zählen erfolgt bisher zusammen mit der Artenbestimmung in der manuellen Auswertung der vorsortierten Videos auf Basis von Expertenwissen/-sichtung. Mit dem Wunsch nach möglichst vollständiger Automatisierung muss jedoch auch der Tracking-Algorithmus um diese Fälle erweitert werden.

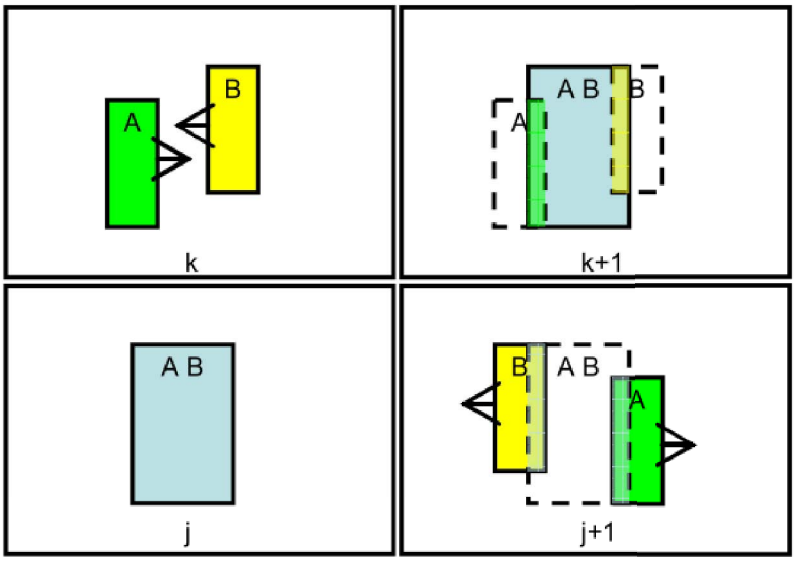

Abbildung 6.1: Zwei sich im Bild überlagernde Objekte A und B werden zu einer Gruppe zusammengefasst. In dem Moment, wo sich die Gruppe wieder auflöst und zwei Objekte erkennbar sind, werden diese mit den Gruppenmitgliedern A und B verglichen und entsprechend zugeordnet. Abbildung aus Yang et al. (2005).

6.3 Automatische Längenbestimmung

Die Längenbestimmung der auf- und absteigenden Fische zählt zu den weiteren Aufgaben des Monitorings von FAH, die in der Zukunft möglichst automatisiert werden soll. Bisher wird die Länge der Fische manuell bestimmt. Hierfür wird der Fisch über den Spiegel im Raum positioniert (siehe Abbildung 6.2) und die tatsächliche Länge des Fisches über bekannte Maßstabsbeziehungen im Erfassungstunnel in Abhängigkeit von seiner Position im Raum bestimmt (Kratzert, 2016). Die räumlichen Maßstabsbeziehungen, der Abstand des Erfassungstunnels zur Kamera und der Umrechnungsfaktor für die Bestimmung der tatsächlichen Länge aus der Fischlänge im Bild, werden über die bekannten Abstände der am Boden montierten Halbkugeln ermittelt und müssen einmalig für jeden eingebauten Erfassungstunnel berechnet werden.

Abbildung 6.2: Über den Spiegel lässt sich der Abstand des Fisches zur Kamera bestimmen. Mit dieser Information und den bekannten Abmessungen im Erfassungstunnel lässt sich die Länge des Fisches im Bild in eine tatsächliche Länge in Zentimeter umrechnen.

Die Idee ist, dass sich dieser Ablauf (Positionierung des Fisches im Raum über den Spiegel, Umrechnung der gemessenen Länge aus der Seitenansicht in eine tatsächliche Länge) gleichermaßen automatisieren lässt. So könnte im Anschluss an die Objektklassifizierung die Längenbestimmung für alle als „Fisch“ klassifizierten Objekte durchgeführt werden. Des Weiteren könnte man überlegen die Länge nicht nur anhand eines Bildes zu berechnen, sondern, um mögliche Ungenauigkeiten zu minimieren, die Länge aus dem Mittelwert mehrere Bilder abzuschätzen. Da der Fisch für die Längenbestimmung so ausgestreckt wie möglich sein sollte, könnte man diejenigen Bilder nehmen, die den höchsten benchmark-Wert (nach Abschnitt 3.3) aufweisen.
Nach diesem Verfahren wurden bereits erste Tests zur automatisierten Längenbestimmung vorgenommen. Anhand eines kleinen Datensatzes, für den die Länge der Fische bereits von Experten bestimmt wurde, wurde die Machbarkeit und Genauigkeit der Methodik untersucht. Ausgewählt wurden allerdings nur Videos mit genau einem Fisch und in klarem Wasser, in dem der Fisch gut im Spiegel zu erkennen ist. Für diese Fälle hat sich gezeigt, dass eine Genauigkeit von rund 3 cm gegenüber der

vom Experten aus dem Video heraus bestimmten Länge erreicht werden kann.
Die Grundvoraussetzung ist allerdings, dass sich der Fisch im Spiegel lokalisieren lässt. Da viele Flüsse Österreichs über mehrere Monate des Jahres allerdings so trübes Wasser aufweisen, dass man den Fisch im Spiegel nicht erkennen kann, ist die Frage inwieweit sich die Längenbestimmung vollumfänglich automatisieren lässt. Ein weiteres Problem sind Videos, in denen viele Fische gleichzeitig durch den Erfassungstunnel schwimmen und eine zusätzliche Zuordnung von Objekten in der Seitenansicht und im Spiegel notwendig ist Abbildung 6.3).

Abbildung 6.3: Problemsituationen für die automatische Längenbestimmung: Viele Fische gleichzeitig im Bild und im Spiegel, die korrekt zugeordnet werden müssen (links) und trübes Wasser, in dem Fische im Spiegel nicht zu erkennen sind (mittig und rechts).

6.4 Bestimmung der Fischart

Die Bestimmung der Fischart auf Basis von Bildern aus Videos ist sicherlich eine der komplexesten Aufgaben, die in Zukunft umgesetzt werden soll. Zwar haben ConvNets gezeigt, dass sie in der Lage sind eine Vielzahl von Objekten zu unterschieden (siehe Abschnitt 4.1.3), jedoch stets anhand von Bildern in denen die Objekte gut fokussiert und detailreich dargestellt sind. Das dies in den Bildern aus dem FishCam-System nicht immer der Fall ist, wurde im Rahmen der hier präsentierten Arbeit mehrfach gezeigt. Deshalb stellt sich die Frage, ob die Bestimmung der Fischart, ähnlich wie die Objektklassifizierung, mittels eines ConvNets und eines Bildes pro Fisch möglich ist.
Das große Problem ist, dass es selbst für den menschlichen Betrachter teilweise unmöglich ist, die Fischart anhand einzelner Bilder aus den Videos zu bestimmen. Gerade im trüben Wasser ist es oft nötig ganze Videosequenzen zu sichten, um un-

terschiedliche Details eines Fisches, anhand denen sich die Fischart bestimmen lässt, auszumachen. Zusatzinformationen zum Bestand im Gewässer und den aktuell häufig auftretenden Fischarten helfen dem menschlichen (wissenden) Betrachter bei seiner Entscheidung. Unterschiedliche Ansätze sind deshalb für die automatisierte Fischartenbestimmung denkbar:

- Bestimmung der Fischart anhand von mehreren ausgewählten Bildern oder kleinen Videosequenzen.
- Kombination eines ConvNets zur Artenbestimmung mit Expertenwissen zum Wanderverhalten der vorkommenden Fischarten. Expertenwissen steht dabei für zusätzliche Informationen zum Bestand der Fische im Gewässer, Zeiträume in denen spezielle Fischarten wandern, Größen die für bestimmte Fischarten unwahrscheinlich bis unmöglich sind, etc.

Gerade der zweite Ansatz klingt dabei vielversprechend, da der Rechenaufwand bei der Auswertung ganzer Videosequenzen pro Fisch ein limitierender Faktor in der Anwendung darstellen könnte.
Möglich wäre es, aus den bisher gewonnenen Monitoringdaten eine Datenbank zu erstellen aus der sich zeitraums- und flussspezifische Verteilungen zum Wanderverhalten der Fischarten auslesen lassen. Um die jährliche Veränderung zu berücksichtigen, wäre eine stetige Aktualisierung der Daten möglich. Allerdings stellt sich die Frage, wie sich die beiden Systeme (ConvNet und Expertenwissen) kombinieren lassen. Folgende Möglichkeiten wären denkbar:

- Direkte Verknüpfung der Klassenwahrscheinlichkeit aus dem ConvNet mit einer Klassenwahrscheinlichkeit aus der Expertendatenbank. Letztere könnte aus den historischen Monitoringdaten über den Zeitpunkt des Videos und den Fluss (-typ) bestimmt werden.
- Erstellen und Trainieren eines zweiten Models, dass hochrangige Repräsentation eines Bildes (z.B. der Werte des letzte Vektor des ConvNets), mit Zusatzinformationen verbindet. Dies könnten nicht nur die zeitraums- und flussspezifischen Statistiken zum Wanderverhalten einzelner Fischarten sein, sondern z.B. auch Informationen zur Fischgröße.

In jedem Fall ist die Entwicklung einer Methode zur verlässlichen Bestimmung der Fischart eine große Herausforderung. Umso mehr, wenn diese über das gesamte Jahr bei klaren und trüben Bedingungen) einsatzfähig sein soll.

7 Fazit

Monitoring von Fischaufstiegshilfen (FAH) ist eine notwendige, bisher jedoch zeit- und kostenaufwendige Aufgabe. Das FishCam-System ist ein Versuch Zeitaufwand und Kosten zu senken und gleichzeitig den Stress auf Fische zu reduzieren. Dies soll zum einen über eine kontaktfreie, video-basierte Überwachung der FAH, zum anderen durch eine automatisierte Auswertung der aufgezeichneten Videodaten geschehen.
Im Rahmen dieser Masterarbeit konnten erfolgreich erste Schritte der automatisierten Auswertung entwickelt und umgesetzt werden. Zu diesen zählen das Erkennen von Objekten in Videos, das Verfolgen von Objekten von Bild zu Bild sowie die Klassifizierung dieser Objekte. Die entworfene Software FishNet wurde komplett in Python geschrieben und ist so aufgebaut, dass sich weitere Funktionen (wie die Längen- und Artenbestimmung) problemlos implementieren lassen.
Obwohl sich FishNet in der stetigen Weiterentwicklung befindet, wurde die Software bereits für diverse Monitoringprojekte unterschiedlicher FAH über den Zeitraum der letzten 2 Jahre verwendet. Dabei wurden von der Software automatisch aus mehr als 1,3 Millionen Videos (über 10 Terabyte) die fürs Monitoring relevanten Videos aussortiert und der manuelle Aufwand (bei der Längen- und Artenbestimmung) auf ein erträgliches Maß reduziert werden.

Literaturverzeichnis

Alsmadi, M. K. S., Omar, K. B., Noah, S. A., and Almarashdah, I. (2009). Fish Recognition Based On the Combination Between Robust Features Selection, Image Segmentation and Geometrical Parameters Techniques using Artificial Neural Network and Decision Tree. *(IJCSIS) International Journal of Computer Science and Information Security*, 6(2):215–221.

BMLFUW (2012). *Leitfaden zum Bau von Fischaufstiegshilfen*. PhD thesis, Bundesministerium für Land- und Forstwirtschaft, Umwelt und Wasserwirtschaft, Wien, Wien.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24(421):123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Broida, T. J. and Chellappa, R. (1986). Estimation of Object Motion Parameters from Noisy Images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(1):90–99.

Cadieux, S., Michaud, F., and Lalonde, F. (2000). Intelligent system for automated fish sorting and counting. *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, 2:1279–1284.

Caruana, R., Karampatziakis, N., and Yessenalina, A. (2008). An empirical evaluation of supervised learning in high dimensions. *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 96–103.

Chen, L.-C., Barron, J. T., Papandreou, G., Murphy, K., and Yuille, A. L. (2015). Semantic Image Segmentation with Task-Specific Edge Detection Using CNNs and a Discriminatively Trained Domain Transform.

Ciregan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3642–3649. 2012 IEEE Conference.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological) (1958): 215-242.*, 20(2):215–242.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *ICML*, 32:647–655.

Donaldson, R. J., Dyer, R. M., and Krauss, M. J. (1975). An Objective Evaluator of Techniques for Predicting Severe Weather Events. In *Preprints, Ninth Conf. on Severe Local Storms*, pages 321–326, Boston. Amer. Meteor. Soc.

EU-Kommission (2000). Richtlinie 2000/60/EG des Europäischen Parlaments und des Rates vom 23. Oktober 2000 zur Schaffung eines Ordnungsrahmens für Maßnahmen der Gemeinschaft im Bereich der Wasserpolitik (Wasserrahmenrichtlinie). *Amtsblatt Nr. L 327*, (vom 22.12.2000):1–72.

Hanselmann, M., Köthe, U., Kirchner, M., Renard, B. Y., Amstalden, E. R., Glunde, K., Heeren, R. M. A., and Hamprecht, F. A. (2009). Toward digital staining using imaging mass spectrometry and random forests. *Journal of Proteome Research*, 8(7):3558–3567.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition.

Hu, M.-K. (1962). Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8:179–187.

image development team, S. (2016). scikit-image Dokumentation http://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.regionprops Letzter Zugriff 05.08.2016.

Jabbar, E. K., Sadiq, A. T., and Ibrahim, N. J. (2014). Foreground detection by using multi features. *IOSR Journal of Computer Engineering*, 16(2):8–12.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature

Embedding. *Proceedings of the ACM International Conference on Multimedia*, pages 675–678.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35.

Karpathy, A. (2014). What I learned from competing against a ConvNet on ImageNet. http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/ Letzter Zugriff 30.08.2016.

Kratzert, F. (2016). Methodik zur manuellen Längenbestimmung im FishCam-Monitoringsystem. Universität für Bodenkultur, Wien.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1097–1105.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval research logistics quarterly*, pages 83–97.

Landesfischereiverband-Oberösterreich (2013). Heimische Fischarten http://www.lfvooe.at/kontakt/service-shop/poster-von-suesswasserfischen/ Letzer Zugriff 03.08.2016.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Yann, Bengio, Y. and Hinton, G. E. (2015). Deep learning. *Nature Methods*, 13(1):35–35.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *Proceedings of the 26th Annual International Conference on Machine Learning ICML 09*, 2008:1–8.

Liu, Z., Li, X., Luo, P., Change, C., and Tang, L. X. (2015). Semantic Image Segmentation via Deep Parsing Network. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 1377–1385.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

Mader, H. and Kratzert, F. (2016). FishCam & FishDef, Forschungsprojekt Fischmonitoring, Zwischenbericht. Technical report, Universität für Bodenkultur, Universität für Bodenkultur, Wien.

Mader, H., Mayr, P., and Kraml, J. (2016). Monitoring the fish migration through. In Webb, J., Costelloe, J., Casas-Mulet, R., Lyon, J., and Stewardson, M., editors, *Proceedings of the 11th International Symposium on Ecohydraulics*, Melbourne, Australia. University of Melbourne.

Matai, J., Kastner, R., and Cutter, G. R. (2012). Automated Techniques for Detection and Recognition of Fishes using Computer Vision Algorithms. *NOAA Technical Memorandum*, pages 35–37.

Munkres, J. (1957). Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5:32–38.

Nery, M. S., Machado, A. M., Campos, M. F. M., Pádua, F. L. C., Carceroni, R., and Queiroz-Neto, J. P. (2005). Determining the appropriate feature set for fish classification tasks. *Brazilian Symposium of Computer Graphic and Image Processing*, 2005:173–180.

Opitz, D. and Maclin, R. (1999). Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, 11:169–198.

Otsu, N. (1979). A Threshold Selection Method from Gray Level. *IEEE Transaction on Systems, Man and Cybernetics*, 9(1):62–66.

Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: An astounding baseline for recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 512–519.

Rodrigues, M. (2010). Automatic fish species classification based on robust feature extraction techniques and artificial immune systems. *Bio-Inspired Computing: Theories and Applications (BIC-TAC), 2010 IEEE Fifth International Conference on. IEEE*, pages 1518–1525.

Rova, A., Mori, G., and Dill, L. M. (2007). One fish, two fish, butterfish, trumpeter: Recognizing fish in underwater video. *MVA2007 IAPR Conference on Machine Vision Applications, May 16-18, 2007, Tokyo, JAPAN*, pages 404–407.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Sánchez, J. and Perronnin, F. (2011). High-dimensional signature compression for large-scale image classification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1665–1672.

Schaefer, J. T. (1990). The Critical Success Index as an Indicator of Warning Skill.

Tcheslavski, G. V. (2009). Morphological Image Processing : Basic Concepts. pages 1–13.

The MathWorks (2016). Matlab Dokumentation http://de.mathworks.com/help/vision/ref/assigndetectionstotracks.html Letzter Zugriff 10.06.2016.

Weichel, T. (2011). *Entwicklung eines Werkzeuges zur systematischen Bewertung der Grundlagen von Hochwassergefahrenkarten*. PhD thesis, Technische Universität Kaiserslautern.

Woschitz, G., Eberstaller, J., and Schmutz, S. (2003). *Mindestanforderungen bei der Überprüfung von Fischmigrationshilfen (FMH) und Bewertung der Funktionsfähigkeit*. Österreichischer Fischereiverband, Wien.

Yang, T., Li, S. Z., Pan, Q., and Li, J. (2005). Real-time Multiple Objects Tracking with Occlusion Handling in Dynamic Scenes. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1(60172037).

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking. *ACM Computing Surveys*, 38(4):13–es.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems 27 (Proceedings of NIPS)*, 27:1–9.

Zhang, D. and Lu, G. (2002). Shape based image retrieval using generic fourier descriptor. *Signal Processing: Image Communication*, 17(10):825–848.

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Anhang

Auflistung der Feature-Subsets

Auflistung der sechs Feature-Subsets als Resultat der sequentiellen Featureselektion nach Abschnitt 4.1.2.4. Die gewählten Generic Fourier Descriptor (siehe Abschnitt 4.1.2.3) sind folgender Maßen gekennzeichnet: FD + Radial Frequenz + Winkel Frequenz

Feature-Subsets für klares Wasser:

1. Circularity, Aspect Ratio, Rectangle Fit Factor, Energy, mean H ventral, mean Cr dorsal, mean Cb ventral, benchmark, FD 1 2
2. Circularity, Rectangle Fit Factor, Aspect Ratio, Inertia, mean H ventral, mean S dorsal, fourth Hu-Moment, benchmark, FD 1 2
3. Circularity, mean Cr ventral, mean H ventral, mean Cr dorsal, mean S dorsal, second Hu-Moment, benchmark, csi, FD 1 2
4. Circularity, Rectangle Fit Factor, Aspect Ratio, Energy, mean Cr ventral, mean H ventral, sixth Hu-Moment, benchmark, FD 1 2
5. Circularity, Aspect Ratio, mean Cb dorsal, mean H ventral, mean H dorsal, mean S dorsal, sixth Hu-Moment, benchmark, csi, FD 0 10, FD 0 12, FD 1 2
6. Circularity, mean Cr ventral, mean H ventral, mean S dorsal, mean Cb ventral, benchmark, csi, FD 1 2

Feature-Subsets für trübes Wasser:

1. Circularity, Area Perimeter Ratio, Aspect Ratio, Inertia, mean Cr ventral, mean Cr dorsal, mean Cb ventral, mean Cb dorsal, mean H ventral, mean H dorsal, mean V ventral, mean S dorsal, benchmark, csi, FD 0 10
2. Circularity, Area Perimeter Ratio, Aspect Ratio, mean Cr ventral, mean Cr dorsal, mean Cb ventral, mean Cb dorsal, mean H ventral, mean H dorsal, mean V ventral, mean S dorsal, benchmark, fourth Hu-Moment
3. Area Perimeter Ratio, mean Cr ventral, mean Cr dorsal, mena Cb ventral, mean Cb dorsal, mean H dorsal, mean V ventral, mean S dorsal, second Hu-Moment, benchmark, csi
4. Circularity, Aspect Ratio, Area Perimeter Ratio, mean Cr ventral, mean Cr dorsal, mean Cb ventral, mean Cb dorsal, mean H ventral, mean V ventral, mean S dorsal, benchmark, csi, FD 0 12
5. Circularity, Aspect Ratio, Area Perimeter Ratio, mean Cr ventral, mean Cr dorsal, mean Cb ventral, mean Cb dorsal, mean H ventral, mean H dorsal, mean V ventral, mean S dorsal, benchmark, csi
6. Circularity, Area Perimeter Ratio, Aspect Ratio, Inertia, mean Cr ventral, mean Cr dorsal, mean Cb ventral, mean Cb dorsal, mean H ventral, mean H dorsal, mean V ventral, mean S dorsal, benchmark, csi, FD 0 12