Universität Bremen

Fachbereich 3: Mathematik und Informatik

Bachelorarbeit

Erkennen und Vermessen von Meereslebewesen auf Stereo-Kamerabildern mit einem neuronalen Netz

Timo Jasper
Matrikelnummer: 4223376

16.09.2019

1. Gutachter: Prof. Dr.-Ing. Udo Frese
2. Gutachter: Prof. Dr. Ralf Bachmayer

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig angefertigt und nicht anderweitig zu Prüfungszwecken vorgelegt habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel verwendet. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Bremen, den 16. September 2019

.....

(Timo Jasper)

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Einleitung 1

1.2 Zielsetzung 1

1.3 Aufbau der Arbeit 2

2 Grundlagen **3**

2.1 DenseNet 3

2.2 MobileNetV2 3

2.3 Das Kameramodell 4

2.4 Stereoskopisches Sehen 5

2.5 Evaluations Metriken 7

2.5.1 Accuracy 8

2.5.2 Confusion Matrix 8

2.5.3 Precision 8

2.5.4 Recall 9

2.5.5 Intersection over Union 9

2.5.6 Mean Squared Error 9

2.5.7 Root Mean Squared Error 9

3 Verwandte Arbeiten **11**

3.1 Stereo-Kameras zur Unterwasser Tiervermessung 11

3.2 Maschinelles Lernen zur Erkennung von Unterwasserlebewesen 11

3.3 Neuronale Netze zur Segmentierung 12

4 Realisierung **13**

4.1 Übersicht 13
4.2 Eingesetzte Mittel 13
4.2.1 Hardware 13
4.2.2 Software 14
4.3 Erzeugung des Datensatzes 15
4.3.1 Erzeugen von Ground-Truth-Segmentierungen aus Annotationen . . . 15
4.3.2 Einteilung in Trainings, Validierungs und Testdaten 17
4.4 Segmentierung und Positionsschätzungen mittels CNN 19
4.4.1 CNN trainieren 20
4.4.2 CNN als Segmentierungsmodell 21
4.5 Objekte clustern 24
4.6 Objekte im zweiten Kamerabild wiedererkennen 24
4.7 Objekte Vermessen 27

5 Evaluation **29**
5.1 Evaluation der Segmentierung 29
5.2 Evaluation der Objekterkennung 31
5.3 Evaluation der Objektzuordnung im zweiten Bild 33
5.4 Evaluation der Objektvermessungen 34
5.5 Laufzeit 34

6 Fazit und Aussichten **35**

A Inhalte der CD **37**

Literaturverzeichnis **40**

Kapitel 1

Einleitung

1.1 Einleitung

Für die Erforschung arktischer Unterwasser Ökosysteme betreiben Biologen des Alfred Wegener Institutes nördlich der Insel Spitzbergen das Unterwasser Observatorium RemOs1 [Fischer et al. (2017)]. Mit RemOs1 werden in regelmäßigen Abständen Stereo-Kamera-Aufnahmen der Umgebung aufgenommen [Fischer et al. (2007)]. Aus Sicht der Biologie sind Informationen zur Häufigkeit des Vorkommens, als auch der Größe aller Lebewesen von Interesse. Die Aufnahmen werden daher regelmäßig von Experten manuell auf Tiere hin untersucht und die Bildpositionen gefundener Exemplare, sowie deren Art annotiert. Die Stereo-Kameras ermöglichen zusätzlich die Vermessung gefundener Tiere [Harvey and Shortis (1995)]. Aus der Gesamtheit dieser Daten können Statistiken über die Tierwelt erstellt werden. Das manuelle Auswerten von Kamerabildern ist eine monotone Aufgabe, die viel Zeit in Anspruch nimmt und hohe Präzision erfordert [Harvey et al. (2004)]. Das Erkennen von biologischen Strukturen auf Unterwasserbildern ist mit herkömmlichen Bildverarbeitungsmethoden nur begrenzt möglich. Im Rahmen der starken Entwicklung im Bereich des maschinellen Lernens in den letzten Jahren haben neuronale Netze (NN) gute Ergebnisse im Bereich der Objekterkennung auf Kameradaten erzielen können. Aufgrund dieser Erfolge ist ein automatisches Verfahren zum Erkennen von Tieren auf Aufnahmen von RemOs1 denkbar. Die Möglichkeit zur automatischen Erkennung von Lebewesen auf RemOs1 Aufnahmen würde eine enorme Zeit- und Kosteneinsparung bei der Auswertung von Unterwasseraufnahmen bedeuten und ist daher von praktischem Interesse.

1.2 Zielsetzung

Im Rahmen dieser Arbeit, soll die Möglichkeit getestet werden Unterwasser Tiere auf Stereokamerabildern des RemOs1-Observatoriums automatisch zu erkennen. Dazu soll ein neuronales Netz lernen, die entsprechenden Unterwasser-Tiere auf Bildern zu segmentieren und

deren artenspezifische Kopf und Schwanz Position zu bestimmen. Durch Nutzung der Kalibrierung der Stereokameras sollen erkannte Tierpositionen auch im zweiten Bild gefunden und die tatsächliche Länge der Tiere vermessen werden. Die Erkennung von Tieren, die sich auf Kamerabildern berühren oder überschneiden wird in dieser Arbeit nicht behandelt, da nur wenige Beispiele im verfügbaren Datensatz vorhanden sind. Das Annotieren der Daten geschieht offline, es gibt daher keine Notwendigkeit, das System echtzeitfähig zu gestalten.

1.3 Aufbau der Arbeit

In Kapitel 2 werden Technologien, die in dieser Arbeit eine Rolle spielen erklärt. In Kapitel 4 wird das Erstellen des Datensatzes, sowie die einzelnen Schritte des Verfahrens erläutert. Anschließend wird in Kapitel 5 das Verfahren evaluiert und die Ergebnisse diskutiert. Als Letztes werden in Kapitel 6 die Ergebnisse der Arbeit zusammengefasst.

Kapitel 2

Grundlagen

In diesem Kapitel werden Grundlagen und Technologien, die eine wichtige Rolle in der Arbeit Spielen erklärt. Es werden zum Verständnis die grundlegendenen Eigenschaften von Technologien erklärt, jedoch keine detaillierten theoretischen Hintergründe. Diese Arbeit setzt Kentnisse zu künstlichen neuronalen Netzen (NN), fully convolutional neural networks (CNN) und deren Training voraus. Eine Einführung in diese Themen würde den Rahmen der Arbeit überschreiten.

2.1 DenseNet

Densely Connected Convolutional Networks (DenseNet) sind CNN, nach der Architektur von Huang et al. (2017). Diese Architektur besteht aus Blöcken von Schichten, den Dense-Blöcken. Innerhalb jedes Dense-Blockes existieren nicht nur die üblichen Verbindungen von einer Schicht zur nächsten, sondern jede Schicht erhält als Eingabe, die Ausgaben aller vorherigen Schichten. Die Idee hinter DenseNet ist das Einsparen von Kanälen, die lediglich zum weiterreichen von Informationen dienen, indem Verbindungen zu tiefer liegenden Schichten direkt geschaffen werden. DenseNets bestehen also aus Schichten mit sehr wenigen Kanälen, wodurch geringere Netzgrößen möglich sind. Die geringe Größe von DenseNets ermöglicht das Trainieren auf kleineren Datensätzen und mit geringeren Hardwareanforderungen.

2.2 MobileNetV2

MobileNetV2 ist eine CNN Architektur von Sandler et al. (2018), die darauf ausgelegt ist, auf schwacher Hardware zu Arbeiten und somit CNN-Anwendungen auf Mobil-Geräten zu ermöglichen. Der Grundgedanke ist es, die Komplexität von Netzen zu verringern, indem Einsparungen vorgenommen werden, die möglichst wenig Einfluss auf das Endergebnis haben. MobileNetV2-Netze bestehen aus Blöcken von Schichten, den *Inverted residual* Blöcke. Diese Blöcke verwenden anstelle von herkömmlichen Konvolution-Schichten mit d_i Kernel pro

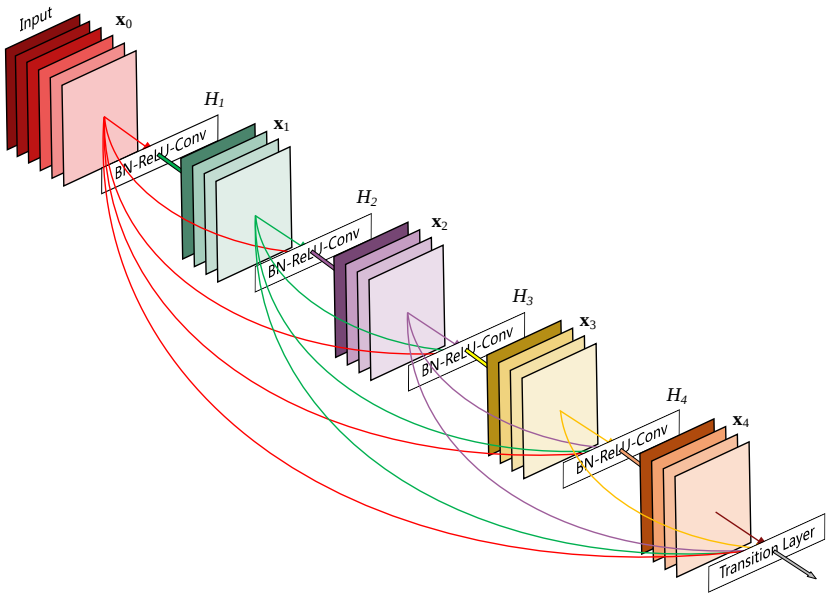

Abbildung 2.1 Darstellung der Verknüpfungen eines Denseblock. Grafik übernommen aus Huang et al. (2017) Seite 1

Ausgabe-Kanal, sogenannte *depthwise separable convolutions* mit einer Konvolution-Schicht mit nur einem Ausgabe-Kanal pro Eingabe-Kanal und anschließend eine 1x1 Konvolution-Schicht mit der gewünschten Anzahl an Ausgabe-Kanälen. Konvolution-Schichten mit der Eingabegröße $h_i \times w_i \times d_i$ und der Ausgabegröße $h_j \times w_j \times d_j$ verwenden einen Kernel der Form $K \in \mathbb{R}^{k \times k \times d_i \times d_j}$ wodurch sich ein Rechenaufwand von $h_i \cdot w_i \cdot d_j \cdot k^2$ ergibt. Die zwei Schichten der *depthwise separable convolution* verwenden nur eine Berechnung pro Ausgabekanal pro Pixel und reduzieren den Rechenaufwand damit auf $h_i \cdot w_i \cdot d_i(k^2 + d_j)$. *Depthwise separable convolution* verringert bei einer Kernelgröße von $k = 3$ den Rechenaufwand der Schicht um den Faktor 8 bis 9, während die Accuracy sich nur geringfügig verschlechtertHoward et al. (2017). Die *Inverted residual blocks* sind zusätzlich nach der Philosophie entworfen, dass sich zu erlernende Eigenschaften auf Hyperebenen geringerer Dimension durch lineare Funktionen Darstellen lassen. Um MobileNetV2 nach dieser Eigenschaft zu Trainieren, enthalten die *Inverted residual* Blöcke Schichten mit linearen Aktivierungsfunktionen, die die Dimensionen der Eingabe so verzerren sollen, dass nachfolgende nichtlineare Schichten die Informationen leichter unterteilen können.

2.3 Das Kameramodell

Sollen Informationen aus Kamerabildern gewonnen werden ist es häufig nötig, das Verhalten der Kamera nachzustellen oder umzukehren. Kameras bilden Objekte der realen Welt auf zweidimensionale Pixelraster (Bilder) ab. Als Modell zur Darstellung der Kameraabbildung wird das Lochkameramodell verwendet. Kameras werden in der Praxis durch eine Kameramatrix und Verzerrungsparameter dargestellt. Die Darstellung 2.3 zeigt anschaulich das

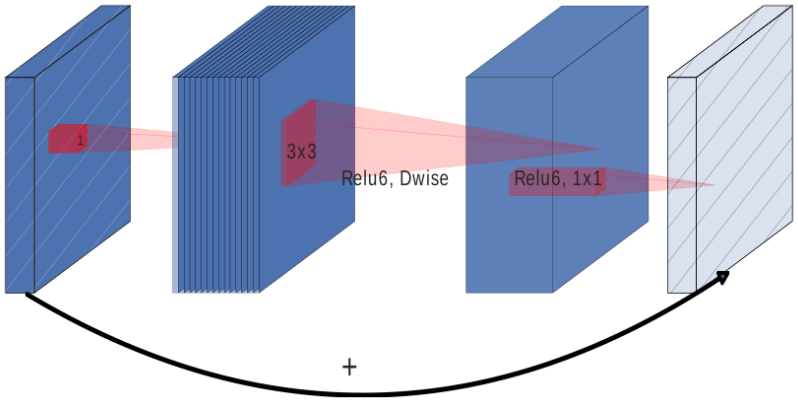

Abbildung 2.2 Depthwise separable convolution Block. Grafik übernommen aus Sandler et al. (2018), Seite 3

Loch-Kamera-Modell. Die 3×3 Kameramatrix wird aus der Kamera-Brennweite und dem optischen Zentrum des Bildes bestimmt und stellt die Abbildung von Weltkoordinaten auf Bildkoordinaten dar. Die üblicherweise überwiegenden Verzerrungen sind die radiale und tangentiale Verzzerung. Die tangentiale Verzerrung ist, die Abweichung des tangentialen Punktes vom Mittelpunkt des Bildes, hervorgerufen durch ungenaue Positionierung von Linse zu Sensorfeld. Die radiale Verzerrung beschreibt Krümmungen der Lichtstrahlen durch die Linse., siehe Abb.

Die Bestimmung von Kameramatrix und Verzerrungs-Parameter nennt man intrinsische Kamerakalibrierung. Bei einer Kalibrierung werden geometrische Gleichungen auf Punkten in Bild-Koordinaten, deren tatsächliche Lage zueinander bekannt ist angenähert.

2.4 Stereoskopisches Sehen

Beim stereoskopischen sehen wird die selbe Szene aus zwei versetzten Blickwinkeln betrachtet, durch den Unterschied der zwei Bilder wird dann eine räumliche Sicht erzeugt. Bei Menschen kann das Gehirn aus den visuellen Reizen beider Augen eine räumliche Wahrnehmung erzeugen. Bei künstlichem stereoskopischen Sehen werden zwei Kameras versetzt aufgestellt und erzeugen dadurch versetzte Aufnahmen der selben Szene, dieser Aufbau wird als Stereo-Kamera bezeichnet.

Um die räumliche Position eines Objektes relativ zu den Kameras zu bestimmen, werden beide Kameramatrizen, sowie die relative Pose der Kameras zueinander benötigt. Ein Punkt in einem der Kamerabilder kann durch das Inverse der zugehörigen Kameramatrix auf eine Linie im Raum abgebildet werden. Das für den den Punkt im Bild verantwortliche physische Objekt befindet sich dabei an einer, noch unbestimmten Stelle, dieser Linie. Wird der selbe Punkt im zweiten Kamerabild bestimmt, so können zwei Linien erzeugt werden, deren Schnittpunkt die Position des Objektes im Raum darstellt, siehe Abb. 2.4. Die Linien

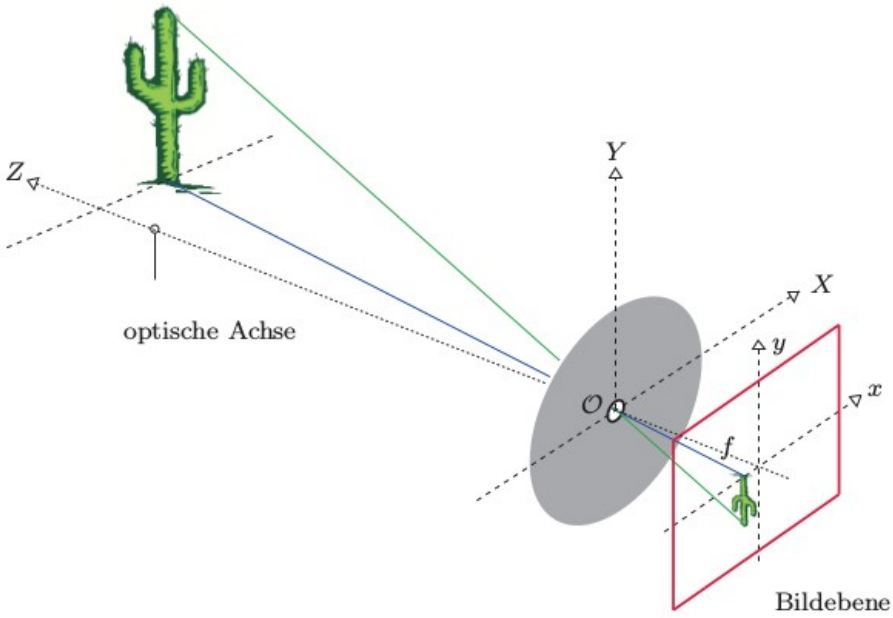

Abbildung 2.3 Darstellung des Loch-Kamera-Modells. Lichtstrahlen eines Objektes der realen Welt verlaufen durch das optische Zentrum und treffen auf das Sensorfeld, dadurch entsteht ein gespiegeltes zweidimensionales Abbild der wirklichkeit. Abb. übernommen aus Burger and Burge (2015), Seite 6.

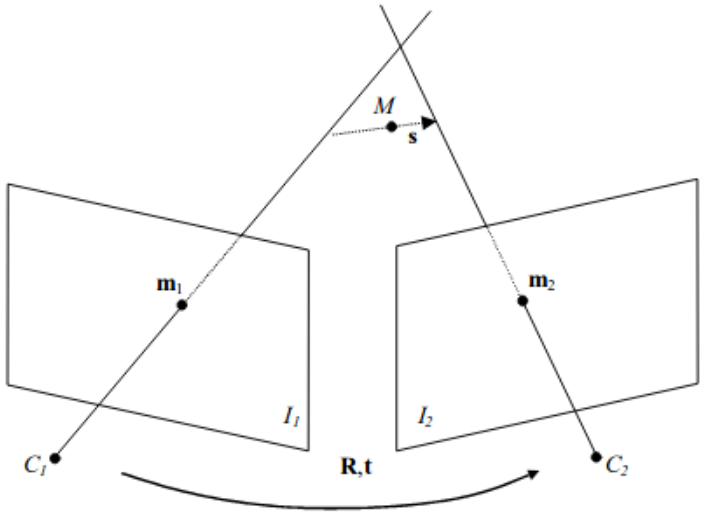

Abbildung 2.4 Darstellung der Stereo-Triangulation, übernommen aus Schreer (2005)

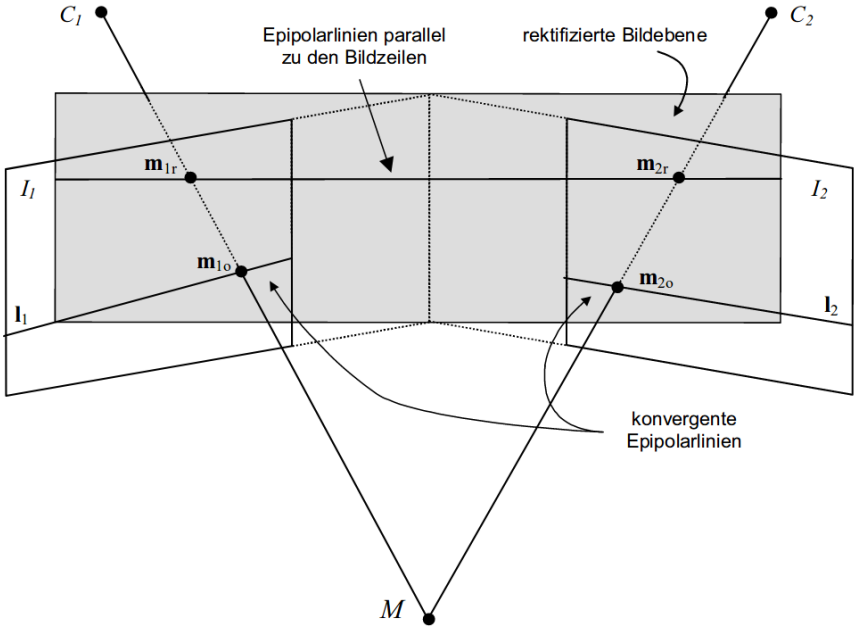

Abbildung 2.5 Darstellung der Stereo-Triangulation, in grau zu sehen ist, wie die Bildebenen beider Kameras durch die Stereo-Rektifizierung angeglichen sind, sodass durchgängie Epipolarlinien entstehen, Gratik übernommen aus Schreer (2005), Seite 106

müssen dabei entsprechend, der Pose der Kameras zueinander, versetzt und rotiert werden. Das Problem kann als Dreieck der Positionen beider Kameras und der Objektposition im Raum angesehen werden, das bestimmen der Objektposition wird daher auch als Stereo-Triangulation bezeichnet. Ein Punkt p auf einem der Kamerabilder kann auf eine Linie im Raum abgebildet werden. Die Kameramatrix der zweiten Kamera kann diese Linie auf die Koordinaten des zweiten Kamerabildes abbilden und somit eine zweidimensionale Linie (Epipolarlinie) im zweiten Kamerabild bestimmen, auf dem der zu p korrespondierende Punkt liegen muss. Durch bekannte Kamerakalibrierungen und bekannter Ausrichtung zueinander können die Kamerabilder Stereo-Rektifiziert werden, dabei werden die Bildebenen beider Kameras angeglichen. Abb. 2.5 stellt die Stereo-Rektifizierung anschaulich dar.

2.5 Evaluations Metriken

In diesem Abschnitt werden Bewertungsmaße eingeführt, die im Rahmen der Arbeit eine Rolle spielen.

In dieser Arbeit wird überwachten Lernen eingesetzt, es werden also Vorhersagen des gelernten Verfahrens mit, erwarteten Ergebnissen (Ground Truth) verglichen. Die folgenden Metriken bieten die Möglichkeit, den Vergleich von Vorhersage und Ground Truth durch wenige Kennzahlen darzustellen.

		Vorhersage	
		Katze	Maus
Real	Katze	2	0
	Maus	1	1

	Vorhersage	
	Katze	Maus
Katze	TP	FN
Maus	FP	TN

Tabelle 2.1 Links: Confusion Matrix für 2-Klassen $(Katze, Maus)$ Klassifikation mit Ground Truth $\{Katze, Maus, Katze, Maus\}$ und Vorhersage $\{Katze, Katze, Katze, Maus\}$. Rechts: Positionen von TP,FN,FP,TN in der Matrix. Zu erkennen ist dass korrekt vorhergesagte Beispiele in die Werte auf der Hauptdiagonalen einfließen, während FP vertikal und FN horizontal dazu liegen (Aussage bezieht sich auf die Erkennung der Klasse $Katze$).

2.5.1 Accuracy

Die Accuracy sag aus, wie viele Vorhersagen richtig sind. Die Accuracy kann für Klassifikationsprobleme angewandt werden und wird berechnet durch die Anzahl korrekter Vorhersagen in Relation zur Anzahl aller Vorhersagen, siehe Formel 2.1.

$$Accuracy = \frac{|\text{korrekte Vorhersagen}|}{|\text{alle Vorhersagen}|} \tag{2.1}$$

2.5.2 Confusion Matrix

Die Confusion Matrix ist eine tabellarische Darstellung der Differenzen von Ground Truth und Vorhersage bei Klassifikationsproblemen, siehe 2-Klassen Beispiel in Tabelle2.1. Die Spalten der Confusion Matrix stehen für die Klassen der Vorhersage, die Zeilen für die Klassen des Ground Truth. In den sich ergebenden Feldern steht die Anzahl der Überschneidungen von Vorhersage und Ground Truth für die jeweiligen Klassen. Die nachfolgenden Begriffserläuterungen werden für Formeln der nachfolgenden Metriken benötigt, jedoch nicht zwingend zum Verständnis der Metriken.

Vorhersagen c' für eine wahre Klasse c können unterteilt werden in **True Positives(TP)**, **True Negatives(TN)**, **False Positives(FP)**, **False Negatives(FN)**. Dabei sind TP alle korrekten Vorhersagen mit $c' = c$. TN sind Vorraussagen, die korrekt als $c' \neg c$ vorhergesagt sind. FP sind fälschlich als $c' = c$ klassifiziert. FN c' die fälschlich als $c' \neq c$ klassifiziert wurden.

2.5.3 Precision

Die Precision beschreibt wie wahrscheinlich eine Vorhersage zutreffend ist. Die Precision wird für jede Klasse seperat berechnet und ist die Anzahl aller korrekten Vorhersagen der Klasse c in Relation zur Anzahl aller Vorhersagen von c, siehe Formel 2.2.

$$Precision = \frac{|\text{TP}|}{|\text{TP} \cup \text{FP}|} \tag{2.2}$$

2.5.4 Recall

Der Recall beschreibt wie viele Vorkommen einer Klasse durch die Vorhersage erkannt werden. Der Recall berechnet sich nach dem Anteil der korrekten Vorhersagen der Klasse c von allen Vorkommen von c, siehe Formel2.3.

$$Recall = \frac{|\mathrm{TP}|}{|\mathrm{TP} \cup \mathrm{FN}|} \tag{2.3}$$

2.5.5 Intersection over Union

Die Intersection over Union (IoU) wird benutzt um die Überschneidung von Vorhersage und Ground Truth, einer Klasse c, zu bewerten. Inbesondere wird dabei sowohl die genauigkeit, wie bei der Precision, als auch die Erkennungsrate wie beim Recall, mit einbesogen. Zur Berechnung, wird der Schnitt aus Vorhersage und Ground Truth in Verhältnis zur Vereinigung der vorhergesagt und wahren Vorhersagen von c.

$$IoU = \frac{|\mathrm{TP}|}{|\mathrm{TP} \cup \mathrm{FP} \cup \mathrm{FN}|} \tag{2.4}$$

2.5.6 Mean Squared Error

Der Mean Squared Error (MSE) ist ein Maß zu Bestimmung der Abweichung von Schätzungen. Der MSE wird beispielsweise verwendet um zwei Mengen von Vektoren V, W, mit jeweils n Vektoren miteinander zu vergleichen.

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (v_i - w_i)^2 \qquad \text{mit } v \in V, w \in W \tag{2.5}$$

2.5.7 Root Mean Squared Error

Der Root Mean Squared Error (RMSE) ist wie der MSE ein Maß zur Bestimmung der Abweichung von Schätzungen. Im Unterschied zum MSE gibt der RMSE den Fehler in der Einheit der Schätzungen an. Formel zur bestimmung des MSE zwei Mengen von Vektoren V, W:

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (v_i - w_i)^2} \qquad \text{mit } v \in V, w \in W \tag{2.6}$$

Kapitel 3

Verwandte Arbeiten

In diesem Kapitel werden Arbeiten vorgestellt, die möglicherweise relevante Ergebnisse für diese Arbeit enthalten. Die erste Arbeit beschreibt die Verwendung von Stereo-Kameras zur Unterwasservermessung. Die zweite und dritte Arbeit stellen präsentieren Verfahren zur automatischen Klassifikation von Fischen. In der letzten Arbeit wird ein CNN verwendet um Bilddaten zu Segmentieren, was ein häufiges Vorgehen in der medizinischen Bildverarbeitung ist.

3.1 Stereo-Kameras zur Unterwasser Tiervermessung

Harvey and Shortis (1995) erproben in ihrer Arbeit das Vermessen von Fischen in Unterwasserumgebungen, in 4-10m Entfernung. Zur Vermessung benutzen sie kalibrierte Stereo-Kameras, anstelle von Verfahren mit nur einer Kamera. Kern des Verfahrens war es, die bekannte Brennweite und das optische Zentrum der Kameras sowie deren Ausrichtung zueinander zu verwenden, um Schnittpunkte von virtuellen Lichtstrahlen beider Kameras als Punkte im dreidimensionalen Raum zu bestimmen. Da Kanten in Unterwasseraufnahmen in der Regel unscharf sind und zu ungenauen Positionsangaben in den Aufnahmen führen, benutzen Punkte an festgelegten Stellen auf den Fischen als Positionsangaben. Harvey und Shortis konnten dabei eine Messgenauigkeit von 95.4% erreichen.

3.2 Maschinelles Lernen zur Erkennung von Unterwasserlebewesen

Siddiqui et al. (2017) entwickelten ein CNN zur Klassifikation von 16 verschiedenen Fischarten. Als Datensatz benutzen sie 1647 Bilder verschiedener Unterwasser Aufnahmestationen. Für jede Fischart standen dabei etwa 100 Bilder zur Verfügung. Für ihre Arbeit nutzten sie verschiedene CNNs, die bereits auf dem ImageNet Datensatz (Deng et al. (2009)) vortrainiert wurden. Durch die vortrainierten Netze ergab sich überhaut erst die Möglichkeit auf

einem so kleinen Datensatz Ergebnisse erzielen zu können. Die besten Ergebnisse konnten sie mit einem ResNet-152 erzielen, dass sie benutzen um Merkmale in den Bildern zu erkennen und Klassifizierten die Bilder mit einer Support Vector Mashine (SVM) auf den Merkmalen. Siddiqui et al. erreichten eine Accuracy von 94.3% auf ungesehenen Testdaten.

Qin et al. (2016) haben ein Live-Verfahren zur Klassifikation von Fischen entwickelt. Das Verfahren benutzt gezielt Vorwissen, um die Eingabedaten drastisch zu verbessern. Es werden in Sequenzen von Bildern, die Hintergründe erkannt. Dies ist möglich, da der Hintergrund im Vergleich zu Objekten sich nicht bzw. nur über lange Sequenzen hinweg ändert. Starke Änderungen im Bild weisen daher auf ein Vordergrund Objekt hin, das vom Hintergrund separiert dargestellt wird. Der separierte Vordergrund wird als Eingabe für ein sehr kleines CNN verwendet, dass wie bei Siddiqui et al. (2017) zur Erzeugung von Merkmalen genutzt wird, die anschließend von einer SVM zu Klassifikation genutzt werden. Quin et al. verwenden einen sehr großen Datensatz bestehend aus 27370 Beispielen zu 23 Klassen. Das Verfahren erreicht eine Accuracy von 94,3%.

3.3 Neuronale Netze zur Segmentierung

Milletari et al. (2017) stellen die Verwendung von CNN zur Segmentierung von tiefer gelegenen Gehirnregionen vor, dies ist interessant für diese Arbeit, da die Gehirnregionen wie die Tiere in dieser Arbeit nur einen kleinen Teil der des gesamten Bildvolumens ausmachen. Milletari et al. verwenden verschiedne CNNs mit dem typischen Aufbau von abwechselnd Konvolution und Pooling Schichten. Die Segmentierung wird durch konvolution Schichten mit $N \times 1 \times 1$ Kernels erzeugt. Die erzeugte Segmentierung wird anschließend wieder auf Eingabegröße skaliert, indem wieder Schichten mit höherer Auflösung an die Segmentierungsschichten angehangen werden. Die angehangenen Schichten enthalten zusätzliche Verbindungen zu früheren Schichten gleicher Auflösung. Das interessante am Vorgehen von Milletari et al. ist, dass die sehr großen Beispielsbilder in kleinere Bilder aufgeteilt werden, von denen immer nur eine Teilmenge Verwendet wird, die in Hinsicht auf Klassenverteilung angepasst werden kann.

Kapitel 4

Realisierung

In diesem Kapitel wird die Entwicklung des Tier-Erkennungs-Verfahrens genauer beschrieben. Es werden die eingesetzte Hard- und Software, die Verarbeitung des Datensatzes und die einzelnen Schritte des Verfahrens genauer erläutert.

4.1 Übersicht

Das Verfahren soll eine Automatisierung der bislang rein manuellen Annotation von Bildpaaren ermöglichen. Die Eingabe und Ausgabe des Verfahrens müssen daher, denen des manuellen Annotationsverfahrens entsprechen. Die Eingabe sind Paare von Stereo-Kamera-Aufnahmen, die Ausgabe ist eine CSV Datei mit Annotationen, die erkannte Objekte den Aufnahmen zuordnen und Informationen zur Klasse, sowie Kopf-, Schwanz-Position und reale Objektlänge enthalten. Das Verfahren wird in vier Schritte unterteilt, die semantischen Segmentierung der einzelner Bilder, sowie Positionsschätzungen von Kopf- und Schwanzpositionen, dem extrahieren einzelner Objekte aus der Segmentierung des vorherigen Schrittes, dem Zuordnen erkannter Objekte zu Positionen im zweiten Bild und der Vermessung von Objekten mittels Triangulation.

Die einzelnen Schritte werden in den folgen Abschnitten näher beschrieben.

4.2 Eingesetzte Mittel

In diesem Abschnitt werden Hardware-Systeme und Software Bibliotheken vorgestellt die bei der Realisierung des Verfahrens eingesetzt wurden.

4.2.1 Hardware

Das Trainieren von NN ist grundsätzlich ein rechenintensives Verfahren mit hohen Anforderungen an das System. Insbesondere der Speicher von Grafikkarten ist ausschlaggebend, da

das Trainieren von Netzen auf dem massiv parallelen Aufbau von Grafikkarten-Prozessoren besonders schnell umgesetzt werden kann. Für die Berechnung auf der Grafikkarte muss das gesamte Modell des NN in den Grafikspeicher passen. Für diese Bachelorarbeit steht folgendes Tower-PC-System zum Trainieren von Modellen zur verfügung:

- CPU: AMD Ryzen5-1600 3.2GHz 6Kerne
- RAM: 16GB DDR4
- GPU: GTX1060 6GB VRAM

Für die Evaluierung wird das Verfahren zudem auf einem Laptop ohne dedizierte Grafikkarte getestet.

- CPU: Intel Core i5-3320M 2.6GHz 4Kerne
- RAM: 12GB DDR3
- GPU: Intel Ivybridge Mobile (unterstützt keine Berechnungen von NN)

4.2.2 Software

Zum auslagern der berechnungen auf die Grafikkarte werden zwei NVIDIA Tools benötigt.

- CUDA : stellt Techniken zur Verfügung, die das berechnen auf Grafikkarten ermöglichen
- cuDNN : stellt spezielle Funktionen zum effektiven berechnen von NN auf Grafikkarten zur verfügung.

Das Verfahren wird in Python3 implementiert, da Python auf allen gängigen Betriebssystem einsetzbar ist und sämtliche benötigten Bibliotheken zur verfügung stellt. Folgende Python-Bibliotheken werden in der Arbeit verwendet:

- keras
- tensorflow
- openCV
- numpy
- matplotlib
- h5py
- scikit-learn

Abbildung 4.1 Beispiele aus den verfügbaren Daten.

4.3 Erzeugung des Datensatzes

In diesem Abschnitt wird die Herkunft, die Verarbeitung und Unterteilung der verfügbaren Daten beschrieben.

4.3.1 Erzeugen von Ground-Truth-Segmentierungen aus Annotationen

Die Beispielsdaten für diese Arbeit wurden vom Alfred-Wegener-Institut Helmholtz Center für Polar- und Meeresforschung bereitsgestellt. Die Aufnahmen wurden von RemOs1 einem Teilsystem des Svalbard Unterwasser Observatoriums in ca. 11 Metern Tiefe, vor der Insel Tiefbergen aufgenommen. Zur Darstellung der Vielfältigkeit der Aufnahmen sind in Abbildung 4.1 einige Beispiele zu finden. Die Daten bestehen aus zeitlich geordneten Stereo-Kamera Aufnahmen mit zugehörigen Annotationen über die Position und Art von erkannten Tieren auf den Aufnahmen. Die Daten stammen aus dem Zeitraum vom 1.Oktober.2017 bis 16.Januar.2018. Die original Annotationen wurden in einem zweischrittigen Verfahren erzeugt. Im ersten Schritt wurde die jeweils linke Aufnahme eines Stereo-Bildpaares, manuell auf Lebewesen hin untersucht. Wurde ein Lebewesen entdeckt wurde die Art des Tieres, sowie die Bildpositionen vom artenspezifischen Kopf und Schwanz des Tieres, bestmöglich bestimmt. In einem zweiten Schritt wurde ebenfalls manuell für alle Lebewesen aus dem linken Bild, mögliche korrespondierende Positionen im rechten Bild gesucht und die Art ggf. noch genauer

Klasse	Okt.	Nov.	Dez.	Jan.	Summe
Fish-Gadus morhua	380	118	181	266	945
Fish-unidentified species	143	15	13	7	178
Pelagic crustacea-Amphipoda sp.	23	35	40	33	131
Pelagic crustacea-unidentified species	3	0	0	0	3
unidentified object	11	22	10	8	51
Benthic crustacea-Sclerocrangon boreas	36	172	67	47	322
Benthic crustacea-Hyas araneus	10	173	206	399	788
Fish-Myoxocephalus scorpius	2	8	9	5	24
Benthic crustacea-unidentified species	2	0	0	0	2
Chaetognatha-Chaetognath sp.	2	13	21	107	143
Pelagic crustacea-Mysis oculata	11	15	148	21	195
fish	3	1	0	0	4
Jellyfish-unidentified species	1	18	7	34	60
Benthic crustacea-Pagurus sp.	1	92	1	9	103
pinnipedia	1	0	0	0	1
Pteropod-Clione limacina	1	1	3	12	17
benthic crustacea	1	2	0	0	3
Jellyfish-Aglantha digitale	1	0	2	5	8
Jellyfish-Mnemiopsis leidyi	1	0	3	1	5
Jellyfish-Cyanea capillata	1	0	0	0	1
Jellyfish-Beroe sp.	1	0	3	64	68
Apendicularia-Apendicularia sp.	1	0	0	0	1
Ophiuroidea	1	0	7	0	8
Jellyfish-Euplokamis dunlapae	1	0	0	1	2
Pelagic crustacea-Copepoda sp.	1	0	0	1	2
Jellyfish-Pleurobrachia pileus	1	0	0	3	4
Jellyfish-Mertensia ovum	1	0	0	6	7
Jellyfish-Bolinopsis infundibulum	1	0	0	0	1
	642	685	721	1029	3077

Tabelle 4.1 Verteilung der Arten über die originalen Annotationen.

bestimmt. Die originalen Daten bestehen insgesamt aus 4918 Bildpaaren mit Annotationen zu 3077 erkannten Objekten. Die genaue Verteilung der Daten ist der Tabelle 4.2 zu entnehmen.

Die Verteilung der Annotationen über die 28 Klassen zeigt, wie ungleichmäßig das Vorkommen mancher Klassen ist. Trainieren von NN anhand wenigen oder gar nur einem einzigen Beispiel ist offensichtlich nicht Zielführend. Die 28 Klassen werden, auf die 5 Klassen Fish, Crustacea, Chaetognatha, Jellyfish und Unidentified-Object abstrahiert um ausreichend Beispiele pro Klasse zu erhalten. Bei der Abstrahierung wird angenommen, dass die zusammengeführten Klassen ähnliche bildliche Eigenschaften aufweisen. Die Klasse Unidentified-Object ist dabei ein Spezialfall und enthält Objekte die nicht erkannt werden konnten oder nicht zu den anderen Klassen zugeordnet werden konnten (pinnipedia, Ophiuroidea, Pteropod-Clione limacina). Die Unidentified-Objects sind also nicht nur stark unterrepräsentiert, sondern bestehen auch noch aus unterschiedlich aussehenden Arten, gute Ergebnisse für diese Klasse

würden also Objekte dieser Klasse für möglicherweise passendere Klassen Vorhersehen.

Die Annotationen sind in erheblichem Maße mit Öbjekt unvollständig"gekennzeichnet, in solchen Fällen sind die Positionen für Kopf und Schwanz häufig, nährungsweise identisch und entsprechen nicht den tatsächlichen Positionen im Bild.

Die Qualität der Annotationen ist für viele biologische und statistische Aussagen sicherlich geeignet, für die Eingabe in ein NN sind sie jedoch nicht direkt nutzbar. Einerseits müssen die Positionsinformationen als Lerndaten einheitlicher sein, außerdem müssen entsprechend dem Aufbau des Netzes, Annotationen, in Form einer semantischen Segmentierung in Ausgabe-Auflösung des CNN vorliegen und die Positionen relativ zu jedem Pixel der Segmentierung angegeben sein 4.4. Ein automatisches Generieren solcher Ground-Truth-Segmentierungen aus den original Annotationen ist nicht möglich, da einerseits viele Positionsangaben der Annotationen nicht korrekt sind und außerdem Kopf und Schwanz Positionen kein Aufschluss über Breite oder Windung von Objekten geben. Für dieses Problem wurde im Rahmen dieser Arbeit ein einfaches Labeltool auf openCV Basis implementiert, mit welchem die original Annotationen als Segmentierung auf den Bildern dargestellt werden und manuell nachgebessert werden können. Mithilfe des Labeltools wurden sämtliche Bilder, für die Objekte annotiert waren manuell segmentiert. Die Segmentierung für Bilder ohne annotierte Objekte ist automatisch erzeugt worden, indem alle Pixel als Hintergrund klassifiziert wurden und die Vektoren als Nullvektoren initialisiert wurden. Abbildung 4.2 und 4.3 veranschaulichen die durch das Labeln erzeugten Segmentierung und Vektoren. Die Beispiele für ein CNN müssen wie folgt aufgebaut sein. Dabei ist S die Segmentierung und K die Klassifizierung für n verschiedene Klassen.

Eingabe: Array der Dimension $(Bildweite, Bildhöhe)$ mit Kanälen für rgb

Ausgabe: Array der Dimension $(S_{weite}, S_{höhe})$

mit Kanälen für $K_0, .., K_{n-1}, Kopf_x, Kopf_y, Schwanz_x, Schwanz_y$

.

4.3.2 Einteilung in Trainings, Validierungs und Testdaten

Der Datensatz wird in Trainings, Validierungs und Testdaten unterteilt, dies ist wichtig, da überwachte Lernverfahren sich auf den Trainingsdaten anpassen. NN die trainiert werden um später Vorhersagen auf ungesehenen Daten zu treffen, müssen auch auf ungesehenen Daten, den Testdaten, bewertet werden. Um bereits während des lernens zu erkennen wie gut das Netz auf ungesehene Daten optimiert ist, werden zusätzlich noch Validierungsdaten bestimmt, auf denen das Netz zwar nicht lernt, anhand derer jedoch während des Trainings der Fortschritt mitverfolgt werden kann. Das Testset sollte groß genug gewählt werden um repräsentative Aussagen zuzulassen, das Validierungsset sollte einerseits repräsentativ sein, andererseits jedoch, möglichst klein um das Training nicht unnötig in die Länge zu ziehen.

Idealerweise sollte das Testset als unabhängiger Datensatz aufgenommen werden, in diesem

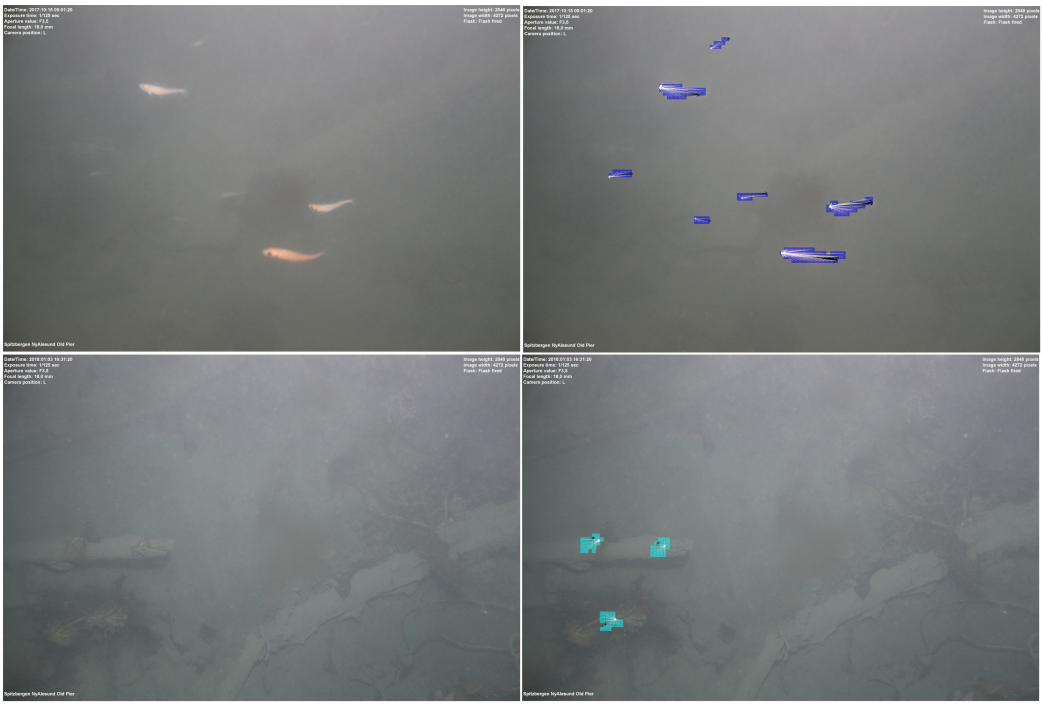

Abbildung 4.2 Visualisierung der Ground Truth Segmentierungen und Positionsvektoren von jedem Ausgabepixel zur Objekt Position. Vektoren, die zum Kopf zeigen in Weiß, Vektoren zum Schwanz in Schwarz. Die Klassen werden durch Farben repräsentiert.

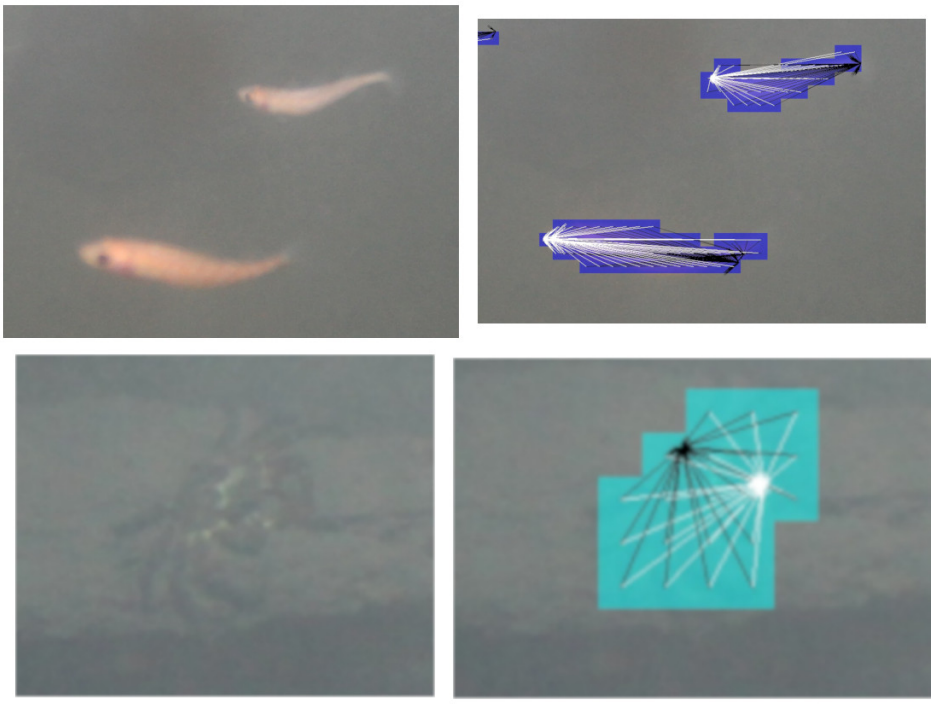

Abbildung 4.3 Vergrößerte Ansicht der Ground Truth Visualisierung.

speziellen Fall mit zeitlich geordneten Daten wäre es das beste, die Daten eines bestimmten Zeitraumes zum Testen zu verwenden. In Tabelle 4.2 lässt sich jedoch erkennen, dass sich das vorkommen der Klassen über die Zeit ändert. Ein fester Zeitraum als Testdaten würde also nicht repräsentativ für alle Klassen sein. Als Alternative wird jedes zehnte Bildpaar ohne Objekt-Annotationen den Testdaten und jedes hundertste den Validierungsdaten hinzugefügt. Das selbe Verfahren wird für die Bildpaare mit Objekt-Annotationen angewandt. Dadurch bleibt das Verhältnis der Beispiele mit Objekten in den Testdaten gleich dem des gesamten Datensatzes, außerdem repräsentieren alle Datensets, das vorkommen von Klassen unabhängig von der Jahreszeit.

4.4 Segmentierung und Positionsschätzungen mittels CNN

In diesem Abschnitt wird auf das Training des CNN, sowie der anschließenden Nutzung als Teil des Verfahrens eingegangen.

Die Bilder stellen eine große Vielfalt an Lichtverhältnissen, Wasserqualität, Hintergünden und Objektausprägungen dar. Herkömmliche Bildverarbeitungsverfahren sind nur schwer für solch variable Daten optimierbar. Neuronale Netze sind in der Lage, sich entsprechend der Vielfalt der Daten auf denen sie trainiert werden, auf Probleme einzustellen, selbst wenn diese von vielen Einflussfaktoren abhängig sind. Speziell für Bilddaten mit hoher variabilität als Eingabe, beweisen sich CNNs, die die Lokalität von Strukturen in Bildern ausnutzen, regelmäßig als das Mittel der Wahl [Russakovsky et al. (2015)]. Aufgrund dieser Eigenschaft wird für die Objekterkennung in dieser Arbeit ein CNN auf die Erkennung von Tieren auf Unterwasseraufnahmen Trainiert. Das CNN soll nach dem Training in der Lage sein, die Position und Art möglicher Tiere auf den Eingabebildern zu erkennen. Für das Training stehen nur einige tausend Beispielsbilder ohne große Vielfalt an Strukturen zur Verfügung. Ein CNN mit millionen von Parametern zu Trainieren, das komplex genug wäre um Bildeigenschaften lernen zu können ist daher eher unwahrscheinlich. Zur Lösung dieses Probelems wird Transfer-Learning verwendet und ein ‚auf dem ImageNet Datensatz vortrainiertes, Netz als Grundlage für ein neues Netz zum erkennen von Tieren verwendet.

Der Grundsätzliche Aufbau aller getesteten Netze ist ein vortrainiertes Grundnetz, das bereits in der Lage sein sollte grundlegende Strukturen in Bildern, wie Kanten, Farben oder Muster zu erkennen. Die Klassifikations-Schichten am Ende der Grundnetze sind auf andere Aufgabe angepasst und werden nicht weiter verwendet. In dieser Arbeit werden MobileNetV2 und DenseNet als mögliches Grundnetz verwendet. Das CNN für das Problem der unterwasser Tiererkennung soll für 3-Kanälige Eingabe-Bilder eine semantische Segmentierung ausgeben und die Positionen des artenspezifischen Kopf und Schwanzes bestimmen. Das CNN wird daher mit einer Ausgabe für die semantische Segmentierung und einer Ausgabe für die relative Positionen, pro Segmentierungs-Pixel in Bildkoordinaten, designed:

- Eine Hohe auflösung der semantischen Segmentierung wird nicht benötigt, da sie nur

zur Klassifierung und groben Positionsbestimmung verwendet wird. Die Segmentierung wird erzeugt, indem auf die Ausgabe der letzten konvolution Schicht des Grundnetzes einige konvolution-Schichten mit der Kernel größe von 1×1 und Relu-Aktivierung aufgebaut werden. Die 1×1 konvolution-Schichten verbinden nur Neuronen der selben Position in der jeweils vorherigen Schicht, jedoch über alle Feature-Kanäle hinweg. Am Ende folgt eine konvolution-Schicht mit 1×1 Kernel, Softmax-Aktivierung und einem Kanal pro Obejktklasse. Diese letzte Schicht dient als Ausgabe und repräsentiert die Wahrscheinlichkeit jeder der Klassen, im korrespondierenden Bildbereich vorzukommen.

- Die genaue Position von Kopf und Schwanz werden durch die zweite Ausgabe bestimmt. Zur genaueren Positionsbestimmung, wird von der letzen Schicht des gekürzten Grundnetzes ein weiterer Netzverlauf abgezweigt, der ähnlich wie der Segmentierungsteil des Netzes aus 1×1 konvolution-Schichten besteht, die jedoch alle über eine lineare Aktivierungsfunktion verfügen. Die letzte 1×1 konvolution-Schicht ist mit 4 Feature-Kanälen ausgestattet, die einen Vektor von der Neuronen Position bis zur Position von Kopf bzw. Schwanz in Bildkoordinaten repräsentieren. Das CNN besteht also aus einem Grundnetz, an dessen Ende sich das Netz in den Klassifikations und Positionen Part teilt und 2 Seperate Segmentierungen für Pixelklasse und Vektor zum Kopf und Schwanz, ausgibt. In Abb. 4.5 werden einige CNN-Vorhersagen im Vergleich zum Ground Truth dargestellt.

Die Kameras nehmen Farbbilder mit einer Auflösung von 2848×4272 Pixeln auf. Alle verwendeten CNN Grundnetze verringern die Auflösung der Schichten bis zu den klassifikations Schichten um den Faktor 32. CNN lassen sich prinzipiell auf beliebig große Eingaben anwenden, da die Gewichte in den Kernel-Matritzen sitzen und daher Positionsunabhängig sind. Praktisch muss jedoch das gesamte Netz in den begrenzten Speicher geladen werden. Häufig werden Bilder dazu herunterskaliert, dadurch gehen jedoch Informationen verloren. In dem Datensatz für diese Arbeit befinden sich kleine Objekte mit einer Größe von nur wenigen Pixeln, die bei einer Skalierung verloren gehen würden. Um das Netz klein genug zu halten, ohne Informationen zu verlieren werden die Eingabebilder für die Trainingsphase in Ausschnitte von 480x480 Pixel geteilt und nacheinander durch das Netz propagiert.

4.4.1 CNN trainieren

Es werden verschiedene Versuche angestellt, bei denen das CNN Modell angepasst wird um Parameter zu finden unter denen das Netz möglichst effektiv und allgemein lernt. Ziel des Netzes ist es den Fehler in den Ausgaben möglichst weit zu reduzieren. Im Training mussten parallel zum Training des CNN auf der Grafikkarte über die CPU die nächsten Trainingsbeispiele von der Festplatte geladen werden, da der Datensatz zu groß ist um in den Arbeitsspeicher geladen zu werden. Es werden dazu Batches geladen die aus 4-10 Beispielen bestehen. Um dem ungleichen Verhältnis von Hintergrund- zu Objekt-Pixeln besteht jeder

Nr.	Grundnetz	Segmentierung	Positionen	Lernrate	Klassen Loss	Positions Loss
1	MobileNetV2	16,16	16,16	0.001	0.0205	95.6424
2	MobileNetV2	16,16	16,16	0.0005	0.0363	67.39
3	MobileNetV2	8,8,8	8,8,8	0.0005	0.0143	25.8658
4	MobileNetV2	32,32	32,32	0.0005	0.0125	27.2391
5	MobileNetV2	128,128	128,128	0.001	0.0227	59.8249
6	MobileNetV2	128,128	128,128	0.001	0.0127	33.1140
7	MobileNetV2	256,256	256,256	0.001	0.0118	17.2020
8	DenseNet	128,128	128,128	0.01	0.0824	171.5987
9	DenseNet	128,128	128,128	0.0005	0.2702	3.3352
10	DenseNet	128,128	128,128	0.001	0.0543	304.0180
11	DenseNet	128,128	256,128,64	0.0005	0.0362	243.3098
12	DenseNet	128,128	256,128,64	0.0005	0.0263	7.9524

Tabelle 4.2 Übersicht über die wichtigsten Trainings Experimente, mit Informationen zum Grundnetz, der Tiefe und Anzahl der 1×1konvolution-Schichten für Klassen und Positions Bestimmung, sowie dem Loss nach 50 Epochen Training.

Batch aus 80% Bildern mit Objektpixeln und zu 20% aus Bildern ohne Objekten. Pro Epoche werden entsprechend auf so vielen Batches Trainiert, bis jedes Bild, das Objekte enthält einmal genutzt wurde. Die Zusammensetzung der Bilder mit Objekten in den Batches wird jede Epoche zufällig neu bestimmt, zudem haben Bilder mit Objekten in jeder Epoche ein 50% Chance horizontal gespiegelt zu werden. Das Vermischen der Batches und das Spiegeln der Bilder soll für vielfältigere Eingaben sorgen, als es der Datensatz ansonsten zulässt. Für das Anpassen der Gewichte während des Trainings wurde der Adam-Optimierer genutzt.

4.4.2 CNN als Segmentierungsmodell

Das Trainierte CNN-Modell stellt den ersten Schritt des Erkennungsverfahrens dar. Das CNN-Modell wurde auf Bildern der Größe 480×480Pixel trainiert um eine größere Batchsize zu ermöglichen. Für die Anwendung im Verfahren ist es sinnvoll größere Eingaben zu verwenden, da die Kernel des Netzes in den Randbereichen der Eingabe weniger Daten bekommen und somit schlechtere Ergebnisse für die Randbereiche zu erwarten sind. Die Eingabegröße wird auf 1472×1472Pixel bestimmt. Dadurch kann ein Eingabebild in 6 CNN Eingaben unterteilt werden die iterativ durch das CNN propagiert werden. Zusätzlich überlappen sich die 6 Teilbilder um 64 bis 96 Pixel, was es ermöglicht, Vorhersagen für die überlappenden Bereiche zu stellen, die nicht nur wenig durch Randpixel betroffen sind. Die flexible Eingabegröße des CNN wird durch den fully-convolutional Aufbau des Netzes erreicht, da sich alle Gewichte in Kernels befinden und somit nicht an die Schichtgrößen gebunden sind.

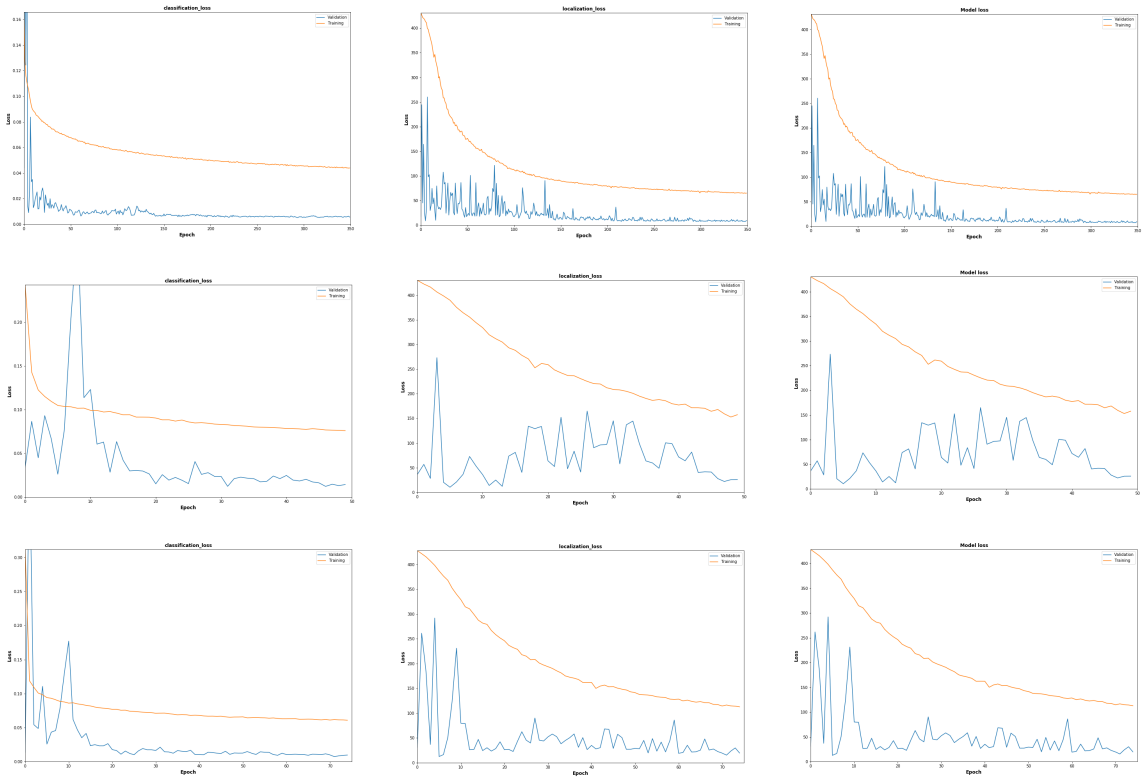

Abbildung 4.4 Loss Funktionen ausgewählter Experimente. In der oberen Zeile die Loss Funktionen des final verwendetet Modells Nr.7, mittlere Zeile: Loss Funktionen von Experiment Nr.3, untere Zeile: Loss Funktionen von Experiment Nr.6. In den Spalten befinden sich von Links nach Rechts: Lossfunktion der Klassen Ausgabe, Loss Funktion der Positions Ausgabe, gemeinsammer Loss. In Blau gezeichnet der Validation Loss, in Rot der Trainings Loss. Zu sehen ist wie das Netz aus Experiment 7 weitere 300 Epochen trainiert wurde und schließlich auf einen Fehler von 0.0085 für die Segmentierung und 9.2765 für die Positionsangaben kommt. Das untypische Verhalten einen geringeren Validierungs-Fehler als Trainings-Fehler vorzuweisen stammt daher, dass die Evaluationsdaten der realen Verteilung der Klassen entspricht, während die Anzahl der Objekt-Klassen in den Trainingsdaten künstlich erhöht ist.

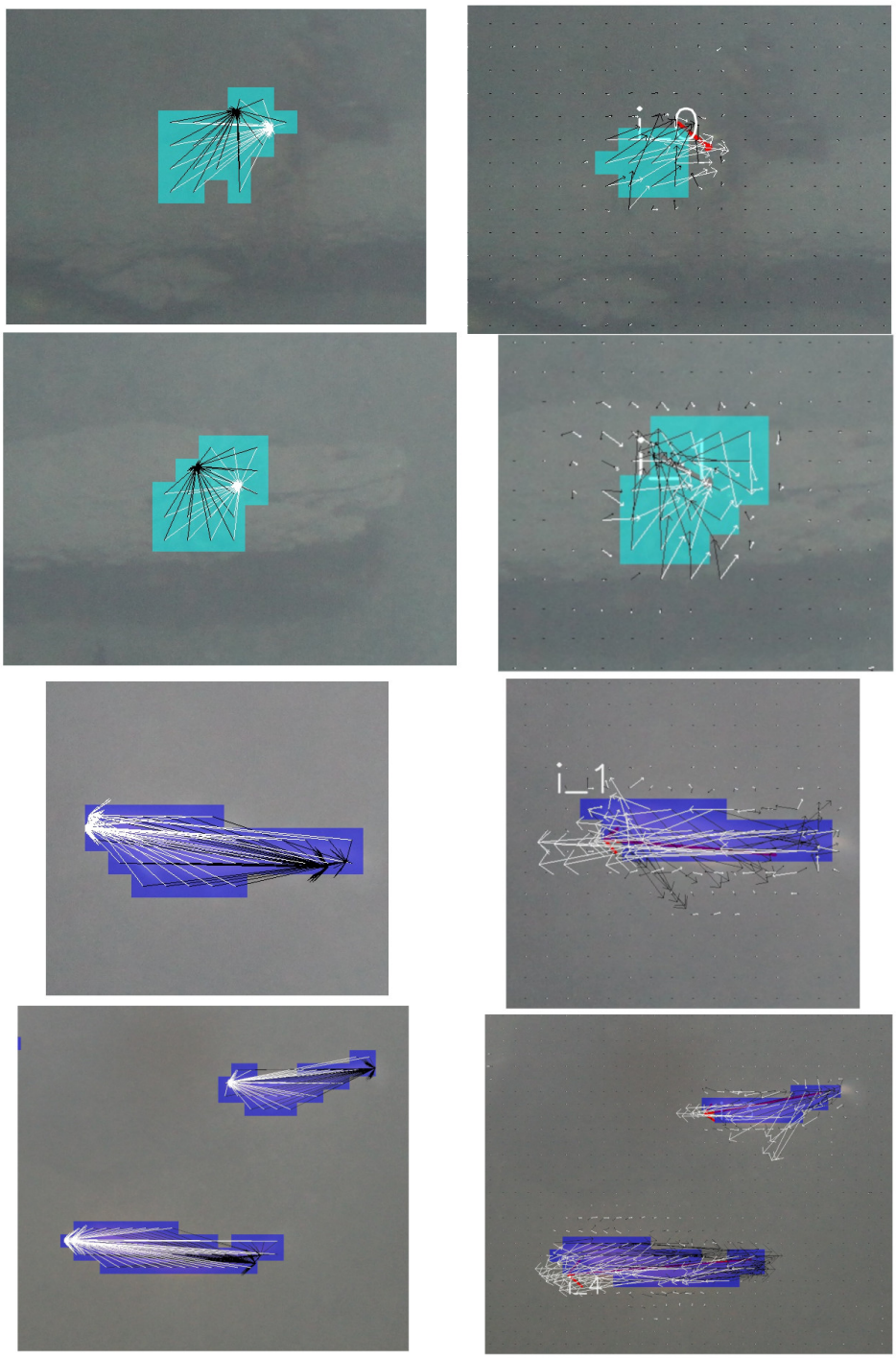

Abbildung 4.5 Darstellung von Vorhersagen des CNN als Gegenüberstellung zu entsprechenden Ground Truth Ergebnissen

4.5 Objekte clustern

In diesem Abschnitt wird gezeigt, wie aus der Segmentierung eines Bildes einzelne Objekte, bestehend aus Klasse, sowie Bildposition von Kopf und Schwanz, extrahiert werden.

Aus der Segmentierung, die bisher pro Pixel die Wahrscheinlichkeiten für jede Klasse enthält, wird pro Pixel die wahscheinlichste Klasse gewählt. In dieser neuen Segmentierung mit nur der Wahscheinlichsten Klasse pro Pixel, werden verbundene Pixel-Regionen (Cluster) gesucht, die nicht als Hintergrund klassifiziert sind. Diese Cluster stellen einzelne Objekte dar, deren Klasse als die häufigste Klasse im gefundenen Clusters definiert wird. Zum finden der Cluster wird ein Hoshen-Kopelman Algorithmus Hoshen and Kopelman (1976) implementiert, dieser liefert Cluster von, durch Nachbarschaft verbundener, Pixel gleicher Klasse. Der Algorithmus unterscheidet dabei nur Hintergrund- und Objekt-Pixel, dadurch wird das Verfahren robust gegen falsch klassifizierte Pixel, die Teil eines größeren Clusters sind. Als benachbart gelten dabei die 8 Pixel, die einen ausgewählten Pixel umschließen.

Die Pixel der Segmentierung repräsentieren aufgrund der lokalen Verbindungen des CNN die Klassifizierung für je einen 32×32Pixel Bereich des Eingabe Bildes. Die Netz Ausgaben zu Kopf und Schwanz haben den Mittelpunkt dieser 32×32Pixel Bereiche zum Ursprund. Wird ein Positionsvektor zu seinem korrespondierenden Mittelpunkt addiert, ergibt sich die positions Vorhersage in Bildkoordianten. Der Median aller Positionen eines Clusters in Bildkoordiaten ergibt die Schätzung der Kopf bzw. Schwanz Position pro Objekt. Anstelle des Durchschnitts wird der Median verwendet um robuster gegen Ausreißer bei der Schätzung von Positionen zu sein.

Durch das Clustern werden somit Objekte mit zugehöriger Klasse und Positionen im Bild ermittelt.

4.6 Objekte im zweiten Kamerabild wiedererkennen

Im folgenden wird gezeigt, wie Objekte die in einen Kamerabild gefunden wurden, auch im Bild der zweiten Kamera gefunden werden können. Ohne die zweite Positionsangabe kann der nächste Schritt nicht durchgeführt werden.

Bei der manuellen Annotation der Bilder wird zuerst das Objekt im linken Bild markiert, anschließend wird versucht das selbe Objekt im rechten Bild zu erkennen. Objekte die aufgrund der versetzt positionierten Kameras nur im rechten Bild zu sehen sind, fließen nicht in die Annotation ein. Ebenso kann für einige Objekte die im linken Bild zu sehen sind keine Position im rechten Bild zugeordnet werden und die Objekte werden ohne Korrespondenz-Punkte annotiert. Um diesem Vorgehen zu entsprechen werden die CNN Vorhersage und das Objekte-clustern auf dem linken Bild ausgeführt. Für das zuordnen von Kopf und Schwanz Positionen zu Positionen im rechten Bild, wird Block-Matching verwendet.

Es wird dabei ein Bildausschnitt (Block) T um eine Kopf bzw. Schwanz Position m her-

Abbildung 4.6 Exemplarische Darstellung der Epilines, gleicher Höhe entlang der horizontalen Bildachse.

um gebildet. Der Block wird mit sämtlichen möglichen Bildausschnitten (in Blockgröße) des rechten Bildes I verglichen. Der Ähnlichste Ausschnitt im rechten Bild sollte dann die Darstellung des selben Objektes der realen Welt, wie im linken Bild sein. Die Ähnlichkeit von Bildausschnitten wird dabei durch den normalisierten Korrelationskoeffizienten R bestimmt Bradski (2000).

$$
\begin{aligned}
R(x,y) &= \frac{\sum_{x',y'} T'(x',y') \cdot I'(x+x', y+y')}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x', y+y')^2}} \\
\text{wobei } T'(x',y') &= T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'') \\
I'(x+x', y+y') &= I(x+x', y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x'',y'')
\end{aligned}
\tag{4.1}
$$

Da bekannt ist, dass es sich um Stereo-Bildpaare handelt können einige Einschränkungen des Suchbereiches im rechten Bild vorgenommen werden. Durch die bekannte Kamerakalibrierung kann für einen Punkt p im linken Bild die Epiline im rechten Bild bestimmt werden 2.4. Für Bilder der gleichen Szene muss der korrespondierende Punkt p' im rechten Bild also auf der Epiline liegen. Durch die versetzten Positionen der Kameras ist es jedoch möglich, dass der korresponierende Punkt außerhalb des rechten Bildes liegt. Das Block-Matching kann also auf einen Bereich entlang der Epiline im rechten Bild beschränkt werden. Für die bessere Handhabung der Suche werden beide Bilder Stereo-Rektifiziert, also so verzerrt, als lägen sie beide auf der selben Bildebene siehe 2.4. Durch die Rektifizierung verläuft die Epiline zu p horrizontal im rechten Bild und zusätzlich auf der gleichen Höhe wie p, siehe Abb. 4.6. Das Block-Matching kann nun auf einem rechteckigen Bildausschnitt um die Epiline im rechten Bild herum ausgeführt werden. Als letzte Einschränkung wird die Tatsache das die Kameras nahezu Parallel ausgerichtet sind ausgenutzt. Dadurch befindet sich p, weiter rechts, als p'. Das Block-Matching kann also auf den Bereich um die Epiline beschränkt, der sich links von p befindet.

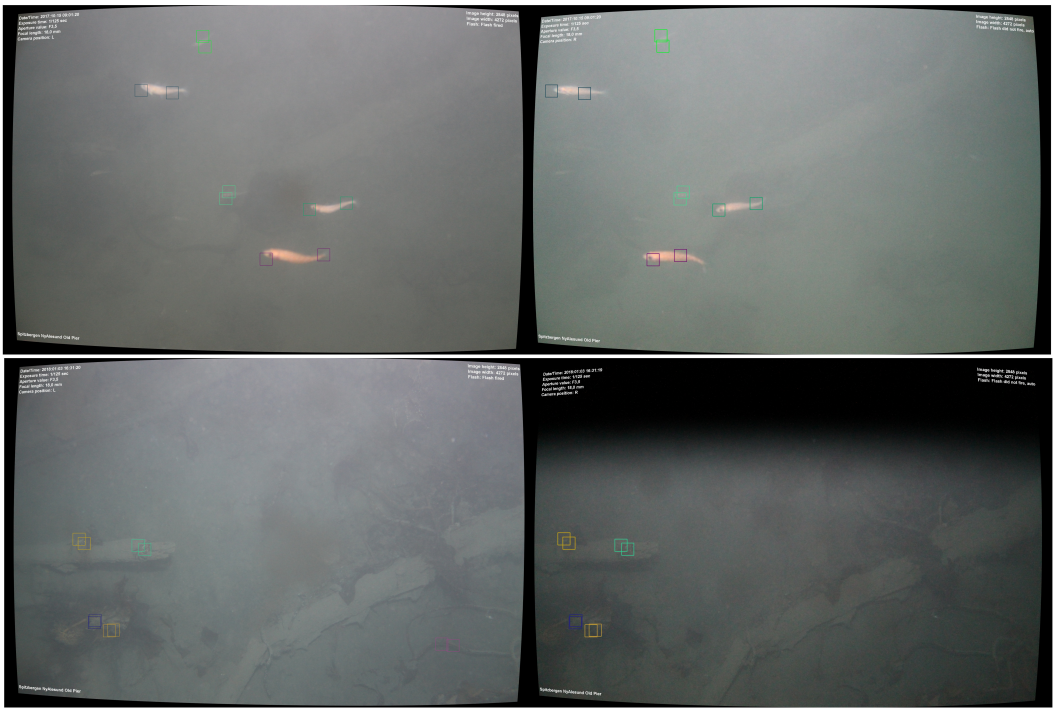

Abbildung 4.7 Ergebnisse des Block-Matching-Verfahrens. Markierte Blöcke des linken Bildes und zugeordnete Positionen im rechten Bild.

Bei der Arbeit mit den Stereo-Bildern fällt auf, dass die Bilder abgesehen vom Betrachtungswinkel deutliche Unterschiede aufweisen, die für Stereo-Bilder nicht vorkommen sollten. Zu beobachten ist die deutlich unterschiedlich Belichtung der beiden Bilder, sowie die unterschiedlichen Körperhaltungen von Tieren. Zu erkennen sind diese Unterschiede unter anderem in Abb. 4.7. Die Auffälligkeiten lassen vermuten, dass die Kameras nicht zum gleichen Zeitpunkt aufgenommen haben sondern kurz nacheinander, im Bereich von Millisekunden. Durch diese mögliche Zeitliche Differenz können Probleme bei der Triangulation der Objekt auftreten, zudem erschwert es möglicherweise die Zuordnung zum Zweiten Bild.

Da immer Paare von Punkten p'_{Kopf}, $p'_{Schwanz}$ gesucht werden, bietet sich die Möglichkeit die Ähnlichkeit von (p_{Kopf}, $p_{Schwanz}$) zu (p'_{Kopf}, $p'_{Schwanz}$) zu untersuchen um mögliche fehlerhafte Zuordnungen zu verwerfen. Es werden p'_{Kopf}, $p'_{Schwanz}$ verworfen, falls die Differenz der Vektoren aus dem linken und rechten Bild größer als der original erkannte Vektor im linken Bild. Genaue Definition:

$$|(p_{Kopf} - p_{Schwanz}) - (p'_{Kopf} - p'_{Schwanz})| \geq |p_{Kopf} - p_{Schwanz}| \tag{4.2}$$

. Dadurch werden grob falsche Ergebnisse des Block-Matchings verworfen. Beispiele für das Block-Matching, sowie verworfene Ergebnisse finden sich in Abb. 4.7

Es stehen nun für alle gefundenen Objekte, Annotation bestehend aus Klasse, sowie Kopf- und Schwanzposition in beiden Bildern zur verfügung.

4.7 Objekte Vermessen

In diesem Abschnitt wird, der letzte Schritt des Verfahrens, die Vermessung von zuvor gefundenen Objekten beschrieben.

Wie in den Grundlagen 2.4 erläutert können Punkte, deren Position in Bildern mehrer Kameras bekannt sind, bei bekannter Lage der Kameras zueinander, sowie bekannter intrinsicher Kalibrierung im 3D Raum trianguliert werden. In diesem Verfahren werden sowohl Kopf- als auch Schwanzposition eines Objektes trianguliert. Die resultierenden Koordinaten verstehen sich als Maßeinheiten der realen Welt (entsprechend dem Referenzsystem bei der Kalibrierung), in diesem Fall Millimetern. Die Länge eines Objektes ist daher der Abstand von triangulierer Kopf- zu Schwanzposition.

Kapitel 5

Evaluation

In diesem Kapitel werden die Ergebnisse des Verfahrens evaluiert. In den folgenden Abschnitten werden die Ergebnisse der einzelnen Schritte des Verfahrens evaluiert.

Die Evaluation erfolgt auf den Testdaten des NN, da die Ergebnisse anderenfalls nicht repräsentativ für ungesehene Daten sein würde.

5.1 Evaluation der Segmentierung

Die vom CNN ausgegebene Segmentierung wird in diesem Schritt mit der manuell erstellten Ground Truth Segmentierung verglichen, dazu werden sowohl Metriken verwendet, als auch eine visuelle Gegenüberstellung. Die Accuracy beträgt $\approx 0{,}9990428109391247$ (Formel 2.5.1), was jedoch nur auf das häufige Vorkommen der Hintergrundklasse zurückzuführen sein könnte.

Tabelle 5.1 stellt Metriken zu den einzelnen Klassen dar, die Background Erkennung fällt, wegen der ungleichen Klassen-Verteilung durch sehr gute Werte auf, was jedoch nur wenig Einfluss auf eine Erfolgreiche Objektsegmentierung hat. Die Fish Klasse hat mit ca. 0,15 eine geringe Precision, es werden also offensichtlich viele Pixel fälschlich als Fish vorhergesagt. Aus der Confusion Matrix der Segmentierung in Abb. 5.1 lässt sich ablesen, dass diese falsch erkannten Pixel mit nur wenigen Ausnahmen in Wahrheit Background Pixel sind. Der Recall

Klasse	**Precision**	**Recall**	**IoU**
Background	0,99983514	0,99922333	0,99905872
Fish	0,14762916	0,53790614	0,13101369
Crustacea	0,24097859	0,46001168	0,18784267
Chaetognatha	0,1025641	0,10084034	0,05357143
Unidentified	0,0	0,0	0,0
Jellyfish	0,01926782	0,08333333	0,01589825

Tabelle 5.1 Metriken zu den semantischen Segmentierungen über einzelnen Klassen, erzeugt aus CNN-Vorhersage und Ground-Truth der Testdaten.

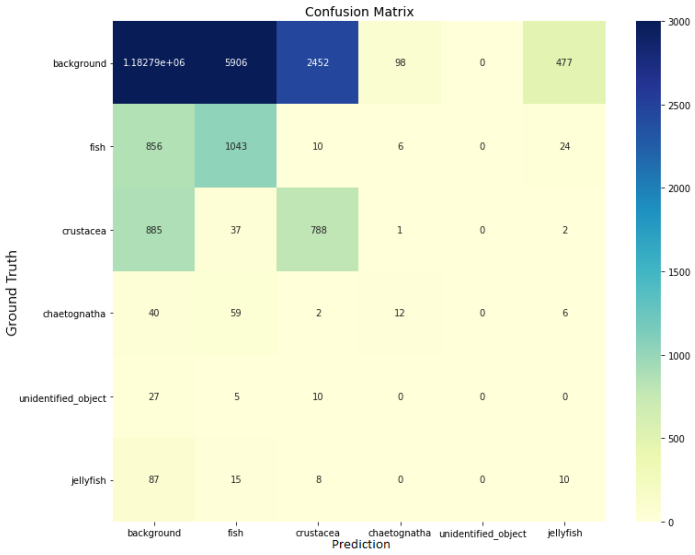

Abbildung 5.1 Confusion Matrix der Segmentierungen der Testdaten

der Fish Klasse ist mit ca. 0,54 noch kein gutes Ergebnis für eine Segmentierung. Für das Verfahren als ganzes ist es von Interesse, ob sich die nicht erkannten Pixel auf komplett nicht erkannte Fische beziehen oder ob lediglich pro Fisch 46% der Pixel nicht als Fisch erkannt wurden. Fehlen die Pixel pro Fisch, so kann das Fish-Objekt möglicherweise aus den verbleibenden 54% der Pixel rekonstruiert werden. Die nicht erkannten Fish Pixel sind nahezu ausnahmslos als Background Pixel vorhergesagt worden.

Die Crustacea Klasse weißt ähnliche Metriken wie die Fish Klasse auf, Unterschiede sind eine leicht höherer Precision und leicht geringerer Recall scheinen. Die IoU ca. beträgt 0,19 und ist damit immernoch weit von guten einem guten Segmentierungsergebnis entfernt. Ebenfalls ähnlich zur Fish Klasse korrelieren die Crustacea und die Background Klasse sehr stark miteinander, sodass viele Crustcea Pixel als Background und viele wahre Background Pixel als Crustacea vorhergesagt wurden. Anzumerken ist jedoch, dass sowohl bei Fischen als auch bei Krebstieren die Anzahl der fälschlich als Background Vorhergesagten Pixel weit größer als die der wahren Objekte, jedoch ist diese Anzahl noch immer nur ein winziger Teil der wahren Background Pixel.

Die übrigen drei Klassen weisen nochmals deutlich schlechte Ergebnisse als die der bisherigen Klassen, die Chetognatha Klasse kann zwar mit eine ähnlichen Precision wie die der Fish Klasse vorweisen, jedoch beträgt die IoU wegen zu vielen Verwechslungen vor allem mit der Fish und der Background Klasse nur ca. 0,07. Die Jellyfish Klasse präsentiert ähnlich schlechte Metriken, ohne nennenswerte Unterschiede. Es fällt auf, dass die Klassen Fish und Crustacea, mit höherem tatsächlichem Vorkommen bessere Ergebnisse erziehlen. Die Unidentified Object Klasse weißt die auffälligen 0,0 Werte in den Metriken auf. Bei der Unidentified Object Klasse handelt es sich um die Klasse mit den wenigsten Beispielen, was schlechte Ergebnisse wahrscheinlicher macht. Bei den Unidentified Objects handelt es sich jedoch um keine natürliche Klasse und die Vorhersagen für Unidentified Objects würden in einem idea-

len CNN als ihre wahren Klassen Vorhergesagt werden, die jedoch nicht dem Ground Truth entsprächen.

Als nächstes wird die Vektorenausgabe des CNN evaluiert. Die vorhergesagten Vektoren werden pixelweise mit den Vektoren des Ground Truth verglichen. Beim Trainieren des CNN wurde nach dem MSE [2.5.6] trainiert, da dieser effizient zu berechnen ist. Für die Evaluation wird der RMSE [2.5.7] verwendet um den Fehler in Pixelkoordinaten zu erhalten und somit anschaulicher zu gestalten. Der RMSE_{global} über alle Vektoren, aller Testdaten beträgt $\approx 3{,}18$ Pixel. Zu beachten ist dabei, dass die Ground Truth Vektoren für Hintergrund-Pixel Nullvektoren sind. Dieser RMSE alleine ist nicht sehr aussagekräftig, da nicht klar ist wie stark der Fehler für Vektoren ist, die ein Objekt anstelle des Hintergrundes Markieren. Zur besseren Differenzierbarkeit werden $\text{RMSE}_{positives} \approx 68{,}0$ Pixel, über alle Vektoren, deren Segmentierung ein Objekt erkannt hat und $\text{RMSE}_{truePositives} \approx 91{,}56$ Pixel über alle Vektoren, deren Segmentierung korrekterweise ein Objekt erkannt hat, bestimmt. Nun wird erkennbar, dass der Fehler für Vektoren die auf Objekte und nicht Hintergrund verweisen deutlich steigt. Für Vektoren die auf tatsächlich vorhandene Objekte verweisen steigt der Fehler nochmals erheblich. $\text{RMSE}_{positives}$ ist für das Verfahren am relevantesten, da die Vektoren, die diesen Fehler erzeugen bei der Objekterkennung Einfluss auf die Objektpositionen nehmen.

5.2 Evaluation der Objekterkennung

In diesem Teil der Evaluation werden die durch clustern bestimmten Objekte mit Objekten des Ground Truth verglichen.

Da der Ground Truth aus einer Segmentierung und den zugehörigen Vektoren besteht, werden die Referenz Objekte erzeugt indem das gleiche Objekte-clustern-Verfahren auf den Ground Truth Daten ausgeführt wird. Durch das manuelle erstellen der Ground Truth Segmentierung ist bekannt, dass pro Objekt immer eine zusammenhängende Segmentierung vorliegt, außerdem verweisen alle Vektoren der Objekt Segmentierungen auf exakt den selben Punkt im Bild. Es kann daher kein Fehler bei der Extraktion von Ground Truth Objekten aus der Ground Truth Segmentierung entstehen.

Um die vorhergesagten Objekte nicht nur durch ihre Anzahl oder Größe mit den Ground Truth Objekten vergleichen zu können, werden Objekte einander zugeordnet, wenn ihre zugehörigen Segmentierungen sich in mindestens einem Pixel überschneiden. Konkret werden bildweise für alle Objekte des Ground Truth, ein Objekt der Vorhersage gesucht, mit dem es eine Überschneidung gibt. Wird kein überschneidendes Objekt gefunden, so wird das Ground Truth Objekt mit einem neuen Objekt der Klasse Background verglichen. Für Objekte ohne Überschneidung wird zudem kein Positionsfehler berechnet, da der Positionsfehler nur für Vorhergesagte Objekte bestimmt wird. Jedes vorhergesagte Objekt kann höchstens einem wahren Objekt zugeordnet werden, vorhergesagte Objekte ohne Zuordnug werden mit Background-Objekten verglichen, wieder ohne Positionsfehler. Background Objekte sind da-

Klasse	Precision	Recall	IoU
Background	0,0	0,0	0,0
Fish	0,11419069	0,76865672	0,11039657
Crustacea	0,14041746	0,49006623	0,12251656
Chaetognatha	0,1	0,18181818	0,06896552
Unidentified	0,0	0,0	0,0
Jellyfish	0,0625	0.16666667	0,04761905

Tabelle 5.2 Metriken zu erkannten Objekten über einzelnen Klassen, erzeugt aus durch clustern gefundenen Objekten und Ground-Truth der Testdaten.

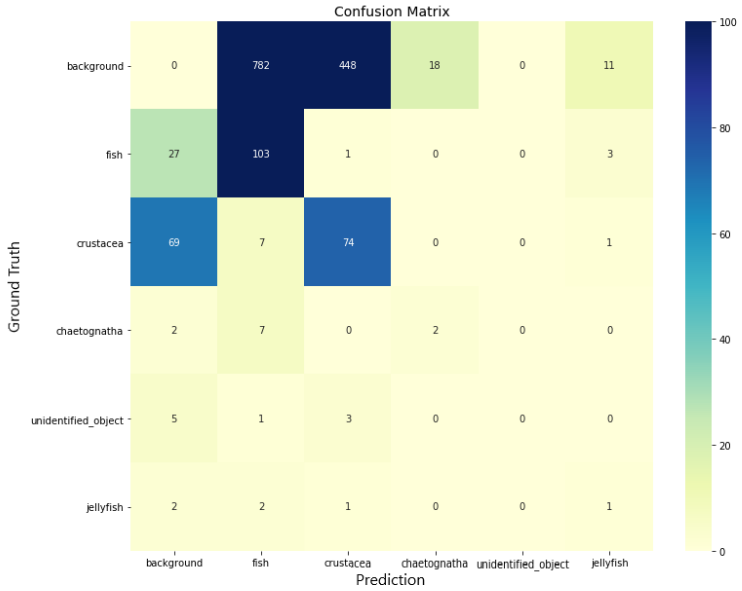

Abbildung 5.2 Confusion Matrix der Objekte der Testdaten

her nur Platzhalter um den Metriken nicht die False-Positive und False-Negative Objekte vorzuenthalten.

Bei der Gegenüberstellung von Ground Truth und vorhergesagter Segmentierung fällt auf, dass die False-Positive Pixel häufig auf Unterwasserpflanzen und Stöckern liegen, also tatsächlichen Konturen, die bei Betrachtung der größeren Umgebung der Pixel leicht von Tieren unterschieden werden können. Betrachtet man nur eine kleiner Umgebung um fie FP-Pixel herum weisen diese durchaus Ähnlichkeiten zu den Flanken eines Fisches auf. Es besteht also die Möglichkeit, dass das rezeptive Feld des CNN für solche Objekte nicht groß genug ist, bzw. die Eingabe zu hochaufgelöst. Jedoch würden bei herunterskalierten Eingaben die kleineren Objekte verloren gehen, die bereits jetzt nur selten erkannt werden. Die unterschiedlich Größe der erkennbaren Objekte ist also offensichtlich ein Problem.

Die Accuracy für die Vorhergesagten Objekte beträgt gerundet 0,115 und enthält keine Beeinflussung durch True-Positives des Hintergrundes. Dadurch die Accuracy deutlich repräsentativer für das Verfahren, wobei die Beispiele für Objekt-Klassen im Datensatz noch immer unausgeglichen sind, wodurch beispielsweise Fehler der Fish Klasse stärker gewichtet sind. Im

Vergleich zu den Metriken der Segmentierunegn weisen die Metriken der Objekte in Tabelle 5.2 höhere Recall Werte. Die höheren Recall Werte spiegeln wieder, dass Objekte erzeugt werden konnten, auch wenn deren Cluster unvollständig waren. Insbesondere werden Fische mit einer Wahrscheinlichkeit von knapp 77% gefunden und Krebstiere immerhin mit einer Wahrscheinlichkeit von 49%.

Die Precision für die Crustacea Klasse ist gesunken und ansonsten grob bei den Werten der Segmentierung geblieben. Die Confusion Matrix, der Objekte, in Tabelle 5.2 zeigt ein gestiegenes Verhältnis von True-Positives zu False-Positives der Klassen Fish und Crustacea, dies lässt sich wohl durch False-Positive Objekte mit kleinen Clustern erklären.

Der RMSE_{Objeke} der Positionsangaben der Vorhergesagten Objekte beträgt gerundet 67,19 Pixel, dabei wurden, wie bereits beschrieben, nur True-Positive Objekte in den Fehler einbezogen, da der Klassifikations-Fehler bereits, in die Evaluation, einbezogen wurde.

5.3 Evaluation der Objektzuordnung im zweiten Bild

Im entwickelten Verfahren werden Objekte die im linken Bild als Tier erkannt wurden im rechten Bild gezielt gesucht. Die Evaluation dieser Zuordnung gestaltet sich schwierig, da Referenzdaten nicht direkt vorliegen. Die originalen Annotationen des Datensatzes haben zwar das entsprechende Format, allerdings wurden die Positionsangaben beim Segmentieren häufig verändert, ein Vergleich ist also nicht möglich. Die Positionen von Objekten die, wie in Abschnitt 5.2, aus Clustern der Ground Truth Segmentierung gewonnen wurden haben zwar die korrekte Positionen, jedoch keine Referenz auf das entsprechende Objekt im zweiten Bild. Um die Objektzuordnung dennoch bewerten zu können, werden gefundene Positionen im rechten Bild manuell zu Referenzpositionen zugeordnet. Die Referenzpositionen sind die Objektpositionen, die durch clustern der Ground Truth Segmentierung erzeugt wurden. Die manuelle Zuordnung ist sehr arbeitsintensiv und wird daher nur für 40 Objekte ausgeführt. Die 40 Objekte werden dabei zu gleichen Teilen unterteilt in Objekte der Klasse Fish und Objekte der Klasse Crustacea, dadurch soll ermöglicht werden den Positionsfehler für Objekte die in Bewegung sind (Fische) und Objekte die häufig still stehen (Krebstiere) zu differenzieren. Es wird differenziert um einschätzen zu können, wie sich der Fehler durch die zeitliche Differenz der Stereo-Aufnahmen verändert, da dieser auf stehende Objekte keinen Einfluss haben dürfte. Der RMSE der Positionen im rechten Bild beträgt ca. 42,23 Pixel für die Krebstiere und ca. 143,82 Pixel für die Fische. Bei der Zuordnung der Objekte ist klar geworden, dass ein höherer Fehler für bei den Fischen bereits dadurch zustande kommt, dass Fische sich näher zu Kamera befinden, als Krebstiere, wodurch Fische größere Bildbereiche einnehmen und somit auch schnell größere Fehler entstehen. Eine Aussage zum Fehler durch die zeitliche Differenz wird daher nicht getätigt. Der mittlere RMSE der beiden Klassen beträgt ca. 93,03 Pixel und liegt damit noch deutlich über dem RMSE von ca. 67,19 Pixel der Objekte im linken Bild. Das schlechtere Ergebnis im rechten Bild soll durch subjektiven Beobachtungen erklärt

werden. Die Ergebnisse des Block-Matchings scheinen einerseits sehr robust darin zu sein, die Positionen im rechten Bild, selbst bei stark veränderter Beleuchtung und teilweise sogar bei veränderten Posen der Tiere zu erkennen. Auffällig ist jedoch auch, dass ungenau erkannte Positionen, die nicht auf oder an den Tierdarstellungen liegen, sondern im monotonen Grau des Hintergrundes häufig zu Ausreißern bei der Position führen.

5.4 Evaluation der Objektvermessungen

Es ist bereits klar, dass die Objektpositionen ungenau sind und die Qualität der Vermessungen sehr start mit den Positionsfehlern korellieren, dennoch. Die Messergebnisse werden mit den Messergebnissen der originalen Annotationen verglichen, dazu werden 20 der Annotationen herangezogen, für die auch eine Vermessung vorliegt. Die Bildpaare der 20 Referenzen werden als Eingabe in das Verfahren gegeben. Aus den Ausgaben werden manuell die zugehörigen Objekte zu den 20 Referenzen gesucht. Dabei wurden 17/20 Referenzen durch das Verfahren erkannt, jedoch nur für 11 der 17 erkannten Objekte wurde auch eine Position im rechten Bild gefunden. Aus den übrigen 11 Objekten wurde eines offensichtlich fehlerhaft mit einer Länge von 10,426m bestimmt und wurde ebenfalls nicht mit berücksichtigt, der RMSE der Längen der übrigen 10 Objekte beträgt ca. 41,56mm, bei einem Durchschnitt der 20 Referenzlängen von gerundet 100,5mm. Der Fehler macht also einen Großteil der tatsächlichen Längen aus, zudem wird für einige der Objekte keine valide Position im rechten Bild gefunden und daher auch keine Distanz berechnet, obowhl dies für die Referenzen möglich war.

5.5 Laufzeit

Die Laufzeit des Verfahrens wurde auf den zur Verfügung stehenden Systemen getestet, siehe 4.2.1. Es wurde dabei die Berechnungsdauer für 48 Bildpaare, also der täglichen Datenproduktion gemessen. Für die Berechnung auf dem Tower-PC wurden ca. 3,0 Minuten benötigt, wobei 24 Sekunden für das Laden des CNN-Modells enthalten sind. Auf dem Laptop, also durch Berechnung auf der CPU wurden ≈35,6 Minuten benötigt, mit 25 Sekunden für das Laden des CNN-Modells. Der Test zeigt wie viel effektiver die Berechnungen auf der GPU ausgeführt werden können.

Kapitel 6

Fazit und Aussichten

Ziel der Arbeit war es ein Verfahren zu Entwickeln mittels dem Tiere auf Stereo-Kamera-Aufnahmen des Unterwasserobservatoriums RemOs1 automatisch erkannt und vermessen werden könnten. Das Enwickelte Verfahren weist Ungenauigkeiten auf, die den Einsatz als vollautomatisches Erkennungswerkzeug verhindern. Es hat sich gezeigt, dass die erkannten Positionen der Lebewesen zu ungenau sind, als dass eine robuste Vermessung mittels Stereo-Triangulation durchgeführt werden könnte. Der Positionsfehler ist bereits in der Ausgabe des neuronalen Netzes zu finden und kann in den nachfolgenden Schritten nicht vollständig verbessert werden. Das Verfahren benutzt für die Bestimmung von Positionen zum Teil Informationen die aus Bildbereichen stammen die weit von der gesuchten Position entfernt liegen, wodurch große Ungenauigkeiten möglich werden. Eine Netzausgabe, bei der jeder Ausgabepixel anzeigt ob er selber eine gesuchte Position ist währe möglicherweise Zielführender. Die Pixel-Klassifikation des eingesetzten neuronalen Netzes hat gezeigt, dass das Erkennen der gesuchten Tierarten durchaus möglich ist. Gerade Tierarten, für die eine größere Anzahl an Beispielen zur Verfügung stand, werden regelmäßig erkannt. Das Erstellen und Trainieren auf weiteren Daten kann zu Verbesserungen führen. Kleine und seltene Tierarten sind naturgemäß deutlich unterrepräsentiert, weswegen sie im Training untergehen. Um die Erkennung durch das neuronale Netz für alle gesuchten Tierarten zu ermöglichen sollte ein ausgeglichenerer Datensatz erstellt werden. Von den Krebstieren der Testdaten konnten 49% und von den Fischen 76,9% erkannt werden. Diese Zahlen sind nicht gut genug für Aussagen zum Tierbestand, jedoch könnte das Verfahren begleitend zur manuellen Erkennung eingesetzt werden um interessante Bildbereiche zu Markieren. Zusätzlich hat das Wiedererkennen von Objekten im zweiten Bild mittels Block-Matching, trotz zeitlicher Differenz der Stereoaufnahmen, sehr gut funktioniert und könnte den zweiten Schritt des manuellen Verfahrens durchaus ersetzen.

Anhang A

Inhalte der CD

Datei	**Beschreibung**
Bachelorarbeit.pdf	Kompilierte PDF dieser Bachelorarbeit
README.md	Anleitung zur Einrichtung der entwickelten Software
Archiv	**Inhalt**
Bachelorarbeit.zip	Die für die Erstellung der PDF notwendigen Quelldateien
PredictionPipeline.zip	Die Quelldateien für das in dieser Arbeit entwickelte System inklusive Konfigurationsdatei mit den Kamerakalibrierungen und Modell des Neuronalen Netzes
HelferSkripte.zip	Die Quelldateien, die unter anderem zum Labeln der Daten, erstellen von Datensätzen oder Trainieren der Netze genutzt wurden.
BeispielBilder.zip	Einige Beispiel Bildpaare zum erproben der Software

* die Inhalte der CD können alternativ unter folgendem Link heruntergeladen werden. https://drive.google.com/open?id=1KE_bK2skNmVg5eaK0mke_zJlRih5_bqb

Literaturverzeichnis

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Burger, W. and Burge, M. (2015). Digitale bildverarbeitung - eine algorithmische einführung mit java, 3. auflage. In *X.media.press*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Fischer, P., Schwanitz, M., Loth, R., Posner, U., Brand, M., and Schröder, F. (2017). First year of practical experiences of the new arctic awipev-cosyna cabled underwater observatory in kongsfjorden, spitsbergen. *Ocean Science*, 13(2):259–272.

Fischer, P., Weber, A., Heine, G., and Weber, H. (2007). Habitat structure and fish: assessing the role of habitat complexity for fish using a small, semiportable, 3-d underwater observatory. *Limnology and Oceanography: Methods*, 5(9):250–262.

Harvey, E., Fletcher, D., Shortis, M., and Kendrick, G. (2004). A comparison of underwater visual distance estimates made by scuba divers and a stereo-video system: Implications for underwater visual census of reef fish abundance. *Marine and freshwater research*, 55:573–580.

Harvey, E. and Shortis, M. (1995). A system for stereo-video measurement of sub-tidal organisms. *Marine Technology Society Journal*, 29(4):10–22.

Hoshen, J. and Kopelman, R. (1976). Percolation and cluster distribution. i. cluster multiple labeling technique and critical concentration algorithm. *Phys. Rev. B*, 14:3438–3445.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861.

Huang, G., Liu, Z., v. d. Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.

Milletari, F., Ahmadi, S.-A., Kroll, C., Plate, A., Rozanski, V., Maiostre, J., Levin, J., Dietrich, O., Birgit, E.-W., Bötzel, K., and Navab, N. (2017). Hough-cnn: Deep learning for

segmentation of deep brain regions in mri and ultrasound. *Computer Vision and Image Understanding*, 164:92–102.

Qin, H., Li, X., Liang, J., Peng, Y., and Zhang, C. (2016). Deepfish. *Neurocomput.*, 187(C):49–58.

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 (IJCV)*, 115(3):211–252.

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. (2018). Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381.

Schreer, O. (2005). Stereoanalyse und bildsynthese. *Fraunhofer HHI*.

Siddiqui, S., Salman, A., Malik, I., Shafait, F., Mian, A., Shortis, M., and Harvey, E. (2017). Automatic fish species classification in underwater videos: Exploiting pretrained deep neural network models to compensate for limited labelled data. *ICES Journal of Marine Science*, 75.