ERKENNUNG VON BOOTEN IN SEENOT AUF DRONENAUFNAHMEN MITTELS PHASE-ONLY TRANSFORM

Therese von Wulffen[1*†], **Vitus Benson**[1*†]

[1] Georg-August-Universität, Göttingen

* {therese.vonwulffen, vitus.benson}@stud.uni-goettingen.de
† Gleicher Anteil, Reihenfolge mittels Zufallszahlengenerator bestimmt.

ABSTRACT

Boote sind für das menschliche Auge auf Dronenaufnahmen leicht erkennbar. Im Falle ziviler Search-and-Rescue-Operationen (SAR) auf dem Mittelmeer können Dronen zur Erkennenung von Booten in Seenot verwendet werden. Zeit ist ein kritischer Faktor und kann durch semiautomatische Analyse des Bildmaterials eingespart werden. Wir verstehen Boote als Anomalien im regelmäßigen Wellenmuster und untersuchen, ob und wie die Phase-Only Transform (PHOT, Aiger & Talbot (2012)), die auf dieser Abstraktion aufbaut, sinnvoll zur Automatisierung der Bootserkennung verwendet werden kann. Im Rahmen dessen entstand eine Python-Bibliothek, mit der unsere Experimente einfach zu replizieren sind. Unsere Ergebnisse sind zweiseitig. Zum Einen kann ein PHOT-basierter Ansatz qualitativ hochwertigere Vorhersagen treffen, als dies ein paar andere klassische Methoden der Bildverarbeitung tun. Zum Anderen finden wir viele Limitierungen, die unsere PHOT-basierte Methodik als nicht reif für den praktischen Einsatz darstellen.

1 EINLEITUNG

Im SOLAS-Abkommen der Vereinten Nationen von 1974 steht, Seenotrettung habe unabhängig von der Nationalität der zu Rettenden zu erfolgen[1]. Nachdem die Mare Nostrum Operation der Italienischen Marine im Herbst 2014 endete[2] ist in den internationalen Gewässern zwischen Libyen und der Europäischen Union keine staatliche Seenotrettungsorganisation mehr aktiv. Stattdessen versuchen zivile Seenotretter mit Search-and-Rescue-Operationen Menschenleben zu retten.

Trotz großer Spendenzuläufe haben diese limitierte Ressourcen. Um die Effektivität der Einsätze zu steigern, ist es von großer Wichtigkeit möglichst schnell Boote in Seenot zu orten. Auf dem Mittelmeer sind dies meist Boote von Flüchtenden, die in Libyen gestartet sind. In der Regel sind diese Schiffe nicht hochseetauglich (meist sind es Schlauchboote), haben unzureichend Wasser und Rettungswesten an Bord und sind selten mit Funkgeräten ausgestattet. Dies macht die Erkennung der Schiffe am Horizont mittels Fernglas zu einer ausgesprochen schwierigen Arbeit.

Deshalb sucht bereits seit 2017 die deutsche NRO Sea-Watch e.V. mit einem Aufklärungsflugzeug[3] aus der Luft die internationalen Gewässer vor der Küste Libyens nach Booten in Seenot ab. Diese Flüge sind sehr kostspielig und können aus diesem Grund nicht rund um die Uhr durchgeführt werden. Mit Kameras bestückte Dronen könnten diese Aufgabe übernehmen. Die Augsburger NRO SearchWing baut preiswerte und hochseetaugliche Dronen, die genau diese Aufgabe durchführen sollen.

[1] https://www.bundestag.de/resource/blob/479394/d98949a58425eea7edecdf34f7442cb4/wd-2-215-14-pdf-data.pdf
[2] http://www.marina.difesa.it/EN/operations/Pagine/MareNostrum.aspx
[3] https://sea-watch.org/das-projekt/moonbird/

Das auf einem Dronenflug aufgenommene Bildmaterial muss ausgewertet werden. Entweder kann dies bereits voll-automisch an Bord der Drone geschehen oder aber im Nachhinein semi-automatisch. Aufgaben der Bilderkennung werden heute meist mit Deep-Learning-Ansätzen gelöst. Diese funktionieren umso besser, je mehr annotierte Daten vorhanden sind. Für diesen konkreten Anwendungsfall gibt es aber kaum Daten. Auch benötigen Deep-Learning-Modelle Grafikprozessoren, um zügig Berechnungen durchzuführen. An Bord einer preiswerten Drone sind diese jedoch sowohl zu teuer als auch zu energieintensiv.

Es ist also sinnvoll, auch mit klassischen Bildverarbeitungsmethoden die Aufgabenstellung, Boote in Seenot auf Dronenaufnahmen zu erkennen, anzugehen. Wir folgen dabei der Intuition, dass Dronenaufnahmen auf hoher See eine relativ regelmäßige Struktur aufweisen. Die Wellen geben ein kontinuierliches Muster, höchstens durch die Sonnenreflektion unterbrochen. Boote stellen dann Anomalien in diesem regelmäßigen Muster dar. Anomalieerkennung ist ein gut erforschtes Gebiet. Wir nutzen die Phase-Only Transform (PHOT) (Aiger & Talbot, 2012), die ursprünglich zur Anomalieerkennung auf Oberflächen entwickelt wurde, um Boote auf See mit Luftaufnahmen zu erkennen.

2 FRAGESTELLUNG

Erkennung von Booten. Ziel ist es, Boote auf Dronenbildern zuverlässig zu erkennen. Die Drone fliegt dabei etwa 300-500m über dem Meer und die Kamera hat eine Auflösung von 8mp, sodass einzelne Boote etwa zwischen 30x30 und 100x100 Pixel groß sind. Zielfunktion der Erkennung ist es, kein Boot zu übersehen, also keine falschen Negativklassifizierungen (*False Negatives*) zu erlauben. Eine Nebenbedingung ist, dabei ein erträgliches Maß an falschen Positivklassifizierungen (*False Positives*) zu haben, eine grobe Richtung sind hier eine falsche Klassifizierung alle 2 Fotos, um einen operativ sinnvollen Dienst zu bieten. Die zweite Nebenbedingung betrifft die Laufzeit, die Erkennung sollte mit einer Geschwindigkeit von etwa einem verarbeiteten Bild pro Sekunde (1 FPS) arbeiten. So könnten die Berechnungen idealerweise irgendwann an Bord der Drone selbst laufen, anstatt offline nach dem Flug ausgeführt zu werden.

Datensätze. Es gibt bisher keinen Datensatz von Fotos mit Booten in Seenot auf dem Mittelmeer, die durch eine Drone aufgenommen wurden. Vermutlich wird es einen hinreichend großen auch nie geben, da wohl nicht mehr als zehn probierte Überfahrten von Geflüchteten pro Tag stattfinden, die wenigsten davon schaffen es in internationale Gewässer auf dem Mittelmeer und noch weniger werden dann von einer zivilen Seenotrettungsorganisation entdeckt. Dieser Mangel an Daten ist auch ein Grund, warum Deep Learning nicht praktikabel für die Aufgabenstellung ist.

Wir approximieren die Aufgabenstellung durch mehrere Datensätze. Primär verwenden wir den Datensatz Bodensee, einen Datensatz der NRO SearchWing, die mit einer Drone Fotos von Segelschiffe auf dem Bodensee aufgenommen hat. Die Bilder sind 3280x2464 Pixel groß und kommen mit annotierten Rechtecken, die angeben, wo in den Bildern sich Boote befinden. Der Datensatz Bodensee enthält knapp 3500 Bilder. Um die Generalisierbarkeit unseres Ansatzes zu testen, verwenden wir außerdem den Seagull (Marques et al., 2015) Datensatz und Aufnahmen des Moonbird Flugzeuges der deutschen NRO SeaWatch e.V..

3 PHASE-ONLY TRANSFORM

Phase-Only Transform. Aiger & Talbot (2012) nutzen die PHOT um Anomalien in Bildern zu erkennen. Speziell zeigen Bai et al. (2014), dass die PHOT gut geeignet ist, um Schäden auf Mikrochips zu erkennen und nutzen erfolgreich die PHOT um Unregelmäßigkeiten auf metallischen Oberflächen zu erkennen. In unserem konkreten Anwendungsfall sind die Boote als Anomalien im regelmäßigen Muster der Wellen anzusehen.

Gegeben ein Farbbild, besteht unsere PHOT-basierte Detektionsmethode aus folgenden Schritten:

1. Bild in Graustufen umwandeln

2. PHOT (Algorithmus 1) anwenden
3. Pixel der Ausgabe über einen Schwellenwert in Binäre Zahlen umwandeln
4. Das Binärbild mittels der morphologischen Operationen Erosion und Dilatation bearbeiten (Serra, 1983)
5. Konturen mithilfe der Method von Suzuki et al. (1985) erkennen
6. Minimale Begrenzungsboxen um die einzelnen Zusammenhangskomponononenten der Konturen finden

Die so berechneten Begrenzungsboxen sind die aus der PHOT entstehenden Vorhersagen von Schiffen. Ein zeitaufwendiger Bestandteil der Methodik sind die zwei diskreten Fourier-Transformationen während der PHOT. Deshalb haben wir verschiedene Implementationen der diskreten Fourier-Transformation (FFT) für unseren Anwendungsfall getestet.

Algorithmus 1 Die Phase-Only Transform

Eingabe: Graustufen-Bild $I(x, y)$ von Größe $m \times n$
Berechne die Fourier-Transformierte $\mathcal{F}(u, v)I$
for $u, v \in \{1, \ldots, m\} \times \{1, \ldots, n\}$ **do**
$\bar{\mathcal{F}}(u, v) = \frac{\mathcal{F}(u,v)I}{\sqrt{Re(\mathcal{F}(u,v)I)^2 + Im(\mathcal{F}(u,v)I)^2}}$
Beende for
Berechne die inverse Fourier-Transformierte $\hat{\mathcal{F}}(u, v)\bar{\mathcal{F}}$
Ausgabe: PHOT-transformiertes Graustufen-Bild

Grundsätzlich gibt es viele unterschiedliche Varianten der FFT, die jeweils für bestimmte Dimensionen der zu transformierenden Matrix besonders schnell sind. Einen Überblick über die wichtigsten Methoden bietet Plonka-Hoch et al. (2018a). Die Entscheidung, welche gängige FFT-Implementation für eine gegebene Matrix am schnellsten ist, kann auch algorithmisch getroffen werden. FFTW (Frigo & Johnson, 1998) geht genau das an. Neben den einfach zu implementierenden FFTs der Python-Bibliotheken NumPy und SciPy, experimentieren wir mit der FFTW und mit einer Fourier-Transformation für dünnbesetzte Matrizen.

Fourier-Transformation für dünnbesetzte Matrizen. Das Ziel der PHOT ist es, ein Bild zu erhalten, in dem Anomalien durch ein höhere Pixelzahl hervorgehoben werden. Wir erwarten also ein Bild, bei dem die meisten Pixel (die, die das Meer repräsentieren sollen) einen sehr niedrigen Wert annehmen. Es sollten also nur wenige Einträge in unserem durch die PHOT transformierten Bild, Einträge haben, die einen bestimmten Schwellenwert ϵ überschreiten. Also erhalten wir eine dünnbesetzte Matrix, bei der wir alle Werte, die kleiner als ϵ sind, als Null betrachten. Es liegt daher nahe, einen FFT-Algorithmus zu verwenden, der dieses Wissen ausnutzt.

Auf der Grundlage von Plonka-Hoch et al. (2018b) hat eine Co-Autorin in von Wulffen (2019) eine zweidimensionale Fourier-Transformation entwickelt, die die Sparsity unseres transformierten Bildes ausnutzt (siehe unter modules.algorithms.sparesefft die Implementation von *sparseifft* und *sparseifft2*). Voraussetzung für die Anwendung des Algorithmus ist, dass es sich bei dem Bild um eine Matrix $\in \mathbb{C}^{N \times M}$ handelt, bei der N und M Zweierpotenzen sind. Daher gibt es die Möglichkeit die Bilder, die transformiert werden sollen, in Zweierpotenzen zu zerschneiden und anschließend nach der PHOT wieder zusammenzufügen (siehe die Funktion *setcutting* im FFTProposalDetector in modules.detectors.fft).

Zudem haben wir einen Algorithmus implementiert, der lediglich ausnutzt, dass das Bild vor der inversen Fourier-Transformation während der PHOT zuvor Fourier-Transformiert worden ist. Schließlich gilt für den ersten Eintrag eines Fourier-tranformierten Vektors $\mathbf{x}$, also für $\hat{\mathbf{x}} = \mathbf{F}_N \mathbf{x} = (\hat{x}_k)_{k=0}^{N-1}$, mit der Fourier-Matrix

$$\mathbf{F}_N := \left(\omega_N^{jk}\right)_{j,k=0}^{N-1} \quad , \quad \omega_N := e^{\frac{-2\pi i}{N}}$$

$\hat{\mathbf{x}}_0 = \sum_{k=0}^{N-1} x_k$ da sich in der ersten Zeile von $\mathbf{F}_N$ nur Einsen befinden. Sobald nun also $\hat{\mathbf{x}}_0 < \epsilon$ gilt, wissen wir, dass **x** der Nullvektor gewesen ist. Daher geben wir in diesem Fall einfach den Nullvektor aus und andernfalls nutzen wir eine klassische inverse Fouriertransfomation (siehe unter modules.algorithms.sparesefft die Implementation von *simpleifft* und *simpleifft2*).

FFTW Muss man die selbe Fourier-Transformation oft auf Matrizen der gleichen Art ausführen, so kann es sich lohnen, in einem ersten Schritt eine FFT zu finden, die diese Art von Matrizen besonders schnell verarbeitet. In einer Planungsphase bricht FFTW das zu erledigende FFT-Problem auf optimale Art und Weise in viele kleine Subroutinen auf. Diese Teilen-und-Erobern-Strategie ist typische für FFT-Varianten. Gleichzeitig generiert FFTW für jede Subroutine die entsprechenden Matrizen der FFT mittels Codeerzeugung. So kann nach der Planungsphase die FFT auf Matrizen der gleichen Dimensionen zeiteffizient durchgeführt werden.

Hyperkomplexe Fourier-Transformation für Farbbilder. Bisher benutzt unsere PHOT lediglich eine Graustufen-Version des Input-Bildes. Normalerweise liegt das Inputbild jedoch in Farbe vor, bei der Umwandlung könnten also Informationen verloren gehen, die für die Erkennung von Anomalien nützlich sind. Ell & Sangwine (2006) stellen eine Fourier-Transformation von Farbbildern auf Basis der hyperkomplexen Fourier-Transformation vor. Dabei werden die drei Kanäle eines RGB-Fotos in den YUV-Farbraum umgewandelt und dann als die drei imaginären Bestandteile eines Quaternions aufgefasst. Ferner wird die FFT auf Quaternionen auf zwei komplexe FFTs zurückgeführt.

Guo et al. (2008) nutzen dies und erweitern die PHOT auf Farbbilder. Im Frequenzraum besteht das transformierte Signal in Polarform dann aus Amplitude, Frequenz und Achse. Wir adaptieren die PHOT in diesen Raum indem wir die Amplitude auf eins normalisieren. Nach dem Durchführen der inversen Fourier-Transformation nehmen wir den quadrierten Absolutbetrag der Pixel (jeder Pixel ist ein Quaternion) als die vorläufige Vorhersage und verfahren dann analog wie mit der Graustufen-basierten PHOT. Wir verwenden den YUV-Farbraum, um ein Bild als Quaternion darzustellen, wohingegen Guo et al. (2008) eine eigene Darstellung nutzen. Nicht ganz offensichtlich ist, wieso die Quaternion PHOT (QPHOT) nur die Information der Amplitude verwirft, und nicht auch die der Achse. Zusätzlich zu unserer funktionierenden Implementation experimentiert wir sowohl damit, als auch damit, die PHOT auf den Frequenzräumen der beiden komplexen FFTs, bevor diese in den Frequenzraum der Quaternionen-FFT umgewandelt werden, anzuwenden. Diese beiden Varianten führten jedoch im Vergleich zu unserer Implementation nicht zu guten Ergebnissen.

4 METHODIK

Projektmanagement. Wir nutzen Gitlab bereitgestellt über den Server der Gesellschaft für wissenschaftliche Datenverarbeitung Göttingen zur Versionskontrolle. Zunächst versuchten wir mit Issues und entsprechend eingerichteten Branches zu arbeiten. Im Laufe der Zeit fiel uns jedoch auf, dass dieses recht aufwendige Vorgehen für ein so kleines und agiles Projekt (in einem Team von zwei Entwickler*innen) wie dieses, etwas zu viel Überhang produziert und eine gemeinsame Versionskontrolle in der main-Branch ausreicht. Auch haben wir unser Repository nach etwa der Hälfte der Zeit neu aufgesetzt, da wir anfangs die Binärdaten ebenfalls kontrolliert haben, was die Größe des Repositorys explodieren ließ. Jetzt arbeiten wir mit einem gitignore und achten darauf, dass nur Quellcode in die Versionskontrolle aufgenommen wird. Grundsätzlich haben wir stets kleine Aufgaben verteilt und uns dabei an Scrum orientiert und uns wöchentlich einmal getroffen um sowohl gemeinsam zu arbeiten als auch in einer Art Stand-Up kurz das weitere Vorgehen zu besprechen.

Kantenerkennung mittels Sobel-Filter. Der Sobel-Operator ist ein Gradientenoperator und wird in der Bildverarbeitung zur Detektion von Kanten eingesetzt. Das Prinzip besteht darin, den Gradienten eines Bildes an einem Punkt x durch die Vektorsummation der Gradienten in der Umgebung von x zu schätzen. Die Vektorsummationsoperation liefert eine Mittelung über Messrichtungen des Gradienten. Um das ganze Bild zu verarbeiten wird der Sobel-Filter über das Bild cross-korreliert (bzw. gefaltet). Für mehr Details verweisen wir auf Sobel & Feldman (1968).

Heuristiken. Im Bodensee Datensatz sind die Schiffe meist weiß und die Wellen dunkler. Mittels einem simplen Schwellwert erhält man also bereits eine relativ gute Entscheidungsgrenze. Konkret erkennen wir Boote als zusammenhängende Pixel, deren Helligkeit und Größe jeweils einen gewissen Schwellenwert überschreiten. Wir bezeichnen diesen Ansatz als Basic-Detektor.

Parameteroptimierung. Die verschiedenen Algorithmen kommen mit Parametern, die optimal zu setzen sind. Dies sind insbesondere die Schwellwerte, die in unterschiedlichen Schritten verwendet werden. Um möglichst gute Ergebnisse zu erzielen optimieren wir diese Parameter mit dem Nelder-Mead-Verfahren (Nelder & Mead, 1965). Konkret minimieren wir $1 -$ F-Beta-Score mit $\beta = 2$. Ist diese Zielfunktion null, so erreichen wir sowohl 100% Recall als auch 100% Precision, also ein perfektes Ergebnis. Unsere Hyperparameter sind mitunter nicht auf ganz $\mathbb{R}$ definiert. Im Nelder-Mead-Verfahren kann diese Nebenbedingung jedoch nicht explizit verwendet werden. Stattdessen definieren wir nichtlineare Transformationen unserer Parameter und können so unsere Nebenbedingungen berücksichtigen. Um gute Startwerte für das Nelder-Mead-Verfahren zu finden, nutzen wir eine Rastersuche auf einem Gitter verschiedener Parameterkombinationen.

Tools. Die meisten Bildverarbeitungen werden in OpenCV (Pulli et al., 2012) durchgeführt, Plots wurden mit Matplotlib (Hunter, 2007) erstellt, als schnelle FFT wurde pyFFTW (Gomersall, 2016) verwendet und Berechnungen wurden mittels NumPy (Walt et al., 2011) und SciPy (Virtanen et al., 2020) implementiert. Als Klasse für Quaternionen wurde Numpy Quaternion (Boyle, 2019) genutzt. Dieser Bericht wurde in Sharelatex, bereitgestellt über den Server der Gesellschaft für wissenschaftliche Datenverarbeitung Göttingen, geschrieben.

5 ERGEBNISSE

Unserer verschiedenen Datensätze haben wir in Trainings- und Testdaten aufgeteilt. Auf den Trainingsdaten haben wir zunächst die verschiedenen Detektoren mithilfe der Parameteroptimierung optimiert und anschließend auf unseren Testdaten evaluiert.

Qualität der Vorhersagen. Um eine Aussage über die Qualität unserer Vorhersagen zu machen, vergleichen wir die von unseren Detektoren berechneten Begrenzungsboxen, die vorhergesagten Schiffe, mit denen zu den Daten annotiereten Rechtecken, also den tatsächlichen Schiffen. Die Ergebnisse präsentieren wir in Tabelle 1 und Tabelle 2. Dabei steht *TP (True Positves)* für richtig detektierte Boote, *FP (False Positives)* für vorhergesagte Boote, die keine sind, und *FN (False Negatives)* für Boote, die unser Detektor übersehen hat. Der *Recall* gibt den Anteil der detektierten Boote zur Anzahl aller Boote, die zu detektieren gewesen wären, wieder. In der Spalte *Precision* befindet sich der Anteil der richtig detektierten Boote gegenüber den gesamten Booten, die der Detektor vorhergesagt hat. Zuletzt findet man in der Spalte F_2 den auf den Recall gewichteten harmonischen Mittelwert von Recall und Precision.

Wie schnell erkennbar ist, funktionieren alle unsere Algorithmen auf dem Bodensee Datensatz ausgesprochen gut, jedoch auf den Moonbird und Seagull Datensätzen lediglich mittelmäßig. Unser QPHOT-basierter Detektor liefert im wesentlichen die selben Ergebnisse, wie die PHOT auf Farbbildern, was die Fragen aufwirft, ob die beiden Methodiken durch die Art der Binärisierung bei der QPHOT äquivalent sind und ob es eine QPHOT-basierte Methodik gibt, die wirklich die zusätzliche Farbinformation nutzen kann.

Grundsätzlich liefern die PHOT-basierten Detektoren deutlich bessere Ergebnisse, als eine simple Schwellwert-basierte Binärisierung des Bildes. So werden von den PHOT-basierten Detektoren Boote unabhängig von ihrer Farbe, siehe Abbildung 1, erkannt, wohingegen der Basic-Detektor dunklere Boote übersieht. Zudem lassen sich PHOT-Detektoren nicht von großflächigen andersfarbigen Bereichen (wie der Himmel) irritieren, womit unser Basic-Detektor erhebliche Probleme hat. Dies kann man in Abbildung 2 gut erkennen, wobei die grünen Rechtecke die tatsächlichen Boote umrahmen und die roten, die von unserem Detektor berechneten Begrenzungsboxen zeigen. Auch die Performance eines Kantenerkennungs-basierten Detektors mittels Faltung mit einem Sobel-Filter wird von der eines PHOT-basierten Detektors übertroffen. Zudem scheitert der Basic-Detektor an den Testdaten. So verschlechtern sich die Performance-Ergebnisse auf den Testdaten gegenüber

Detektor	TP	FP	FN	Recall	Precision	F_2
FFT Detector	893	212	107	0.8929	0.8081	0.8746
FFT Quaternion	893	212	107	0.8929	0.8081	0.8746
Sobel Detector	828	463	172	0.8279	0.6414	0.7824
Basic Detector	878	1206	122	0.8779	0.4213	0.7215

(a) Bodensee Datensatz

Detektor	TP	FP	FN	Recall	Precision	F_2
FFT Detector	317	913	311	0.5048	0.2577	0.4236
FFT Quaternion	317	913	311	0.5048	0.2577	0.4236
Sobel Detector	262	719	366	0.4172	0.2671	0.3750
Basic Detector	245	1382	383	0.3901	0.1506	0.2960

(b) Moonbird Datensatz

Detektor	TP	FP	FN	Recall	Precision	F_2
FFT Detector	751	1603	969	0.4366	0.3190	0.4066
FFT Quaternion	751	1603	969	0.4366	0.3190	0.4066
Sobel Detector	258	3533	1462	0.1499	0.0681	0.1209
Basic Detector	554	9820	1166	0.3221	0.0534	0.1605

(c) Seagull Datensatz

Tabelle 1: Vorhersagequalität der untersuchten Methodiken auf den Trainingsdatensätzen. Die Parameter wurden genau für diese Daten optimiert.

Detektor	TP	FP	FN	Recall	Precision	F_2
FFT Detector	476	240	26	0.9482	0.6648	0.8737
FFT Quaternion	476	241	26	0.9482	0.6639	0.8734
Sobel Detector	446	207	56	0.8884	0.6830	0.8380
Basic Detector	450	2977	52	0.8964	0.1313	0.4139

(a) Bodensee Datensatz

Detektor	TP	FP	FN	Recall	Precision	F_2
FFT Detector	69	400	59	0.5391	0.1471	0.3517
FFT Quaternion	69	400	59	0.5391	0.1471	0.3517
Sobel Detector	59	286	69	0.4609	0.1710	0.3442
Basic Detector	57	455	71	0.4453	0.1113	0.2783

(b) Moonbird Datensatz

Detektor	TP	FP	FN	Recall	Precision	F_2
FFT Detector	186	314	270	0.4079	0.3719	0.4002
FFT Quaternion	186	314	270	0.4079	0.3719	0.4002
Sobel Detector	77	827	379	0.0852	0.1689	0.1411
Basic Detector	128	2452	328	0.2807	0.0496	0.1453

(c) Seagull Datensatz

Tabelle 2: Vorhersagequalität der untersuchten Methodiken auf den Testdatensätzen. Die Parameter wurden auf dem jeweiligen Trainingsdatensatz optimiert.

den Trainingsdaten erheblich. Ein Kantenerkennungs-basierter oder PHOT-basierter Detektor zeigt hingegen nur kleine Abweichungen zwischen der Performance auf den Traningsdaten und den Testdaten. Offen bleibt, ob die Kombination der verschiedenen Methodiken noch einmal den F_2-Wert erhöhen kann.

Die Parameteroptimierung ist, da sie nicht gradientenbasiert ist, eine äußerst mühselige Aufgabe. Insbesondere konvergiert der Nelder-Mead-Algorithmus nur dann zu einem Optimum, wenn der initiale Parameter-Simplex bereits in der Nähe dieses liegt. Insbesondere beim Seagull-Datensatz besteht die Möglichkeit, das hier andere Parameterkombinationen noch bessere Ergebnisse liefern. Bezüglich des Moonbird Datensatzen ist anzumerken, das die Bilder in diesem oft nicht realen

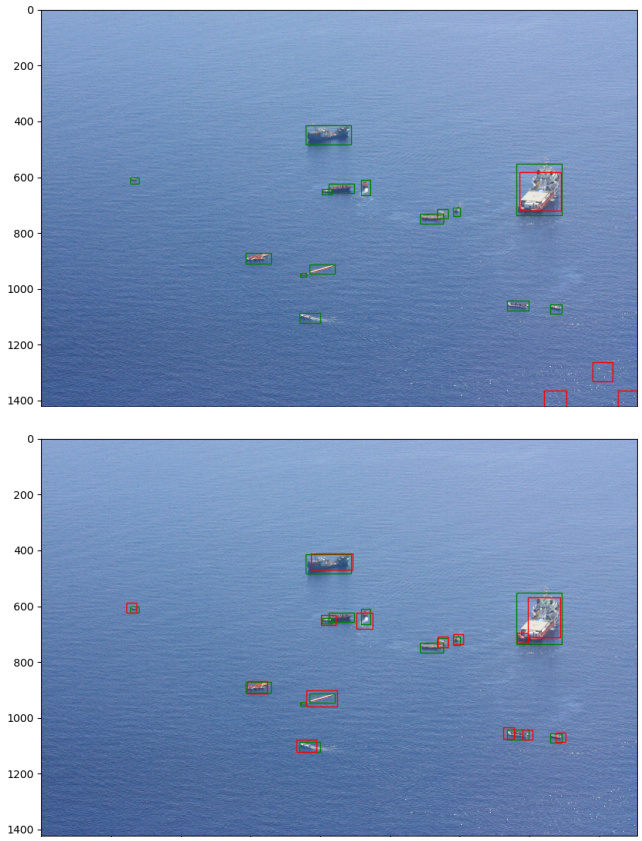


Abbildung 1: Vergleich von Basic (oben) und PHOT auf Moonbird

Dronenfotos entsprechen, da sie oft nicht parallel zur Wasseroberfläche aufgenommen wurden. Der Seagull Datensatz beinhaltet viele Sonnenreflektionen, diese werden oft mit zahlreichen Begrenzungsrahmen erkannt, hier könnte vielleicht eine Methode aufbauend auf einem Schwellwert einer maximalen Anzahl benachbarter Begrenzungsrahmen die Anzahl an False Positives deutlich senken.

In der praktischen Anwendung an Bord eines zivilen Seenotrettungsschiffes wird jede Bootdetektion eines Algorithmuses noch einmal von einem Menschen überprüft. Von daher ist eine Betrachtungsweise der Erkennungsqualität der Algorithmen anhand der Bilder, die sich hinterher ein Mensch anguckt, sinnvoll. Eine niedrige Präzision könnte dazu führen, dass die Person, die sich hinterher die ausgewerteten Bilder anschaut, unnötig viele Bilder betrachten muss. Wie die Abbildung 3 zeigt, werden jedoch größere Boote auch mal durch mehrere Bounding-Boxes, statt durch nur eine, detektiert, was eine Ursache niedriger Präzision sein könnte. Um die Performance noch besser hinsichtlich eines semi-automatischen Anwendungsfalles zu messen, formulieren wir die folgenden Fragen:

1. *Auf wie vielen Bildern wurden Boote vorhergesagt, auf denen sich am Ende kein einziges Boot befunden hat?*
2. *Wie viele Bilder existieren, auf denen der Detektor kein Boot vorhersagt, obwohl sich welche auf dem Bild befanden?*

Beide Zahlen sollten möglichst klein sein, damit auf der einen Seite nicht unnötig viele Bilder per Hand bewertet werden müssen, auf denen sich kein einziges Boot befindet. Auf der anderen Seite soll auch kein Bild übersehen werden, auf dem sich ein Boot befindet. Die mit den auf den Trainingsdaten optimierten Detektoren erzielten Ergebnisse zu diesen Fragen sind in Tabelle 3 festgehalten. Auch hier bestätigen sich die Ergebnisse von zuvor: Die PHOT-basierte Methodik funktioniert am Besten.

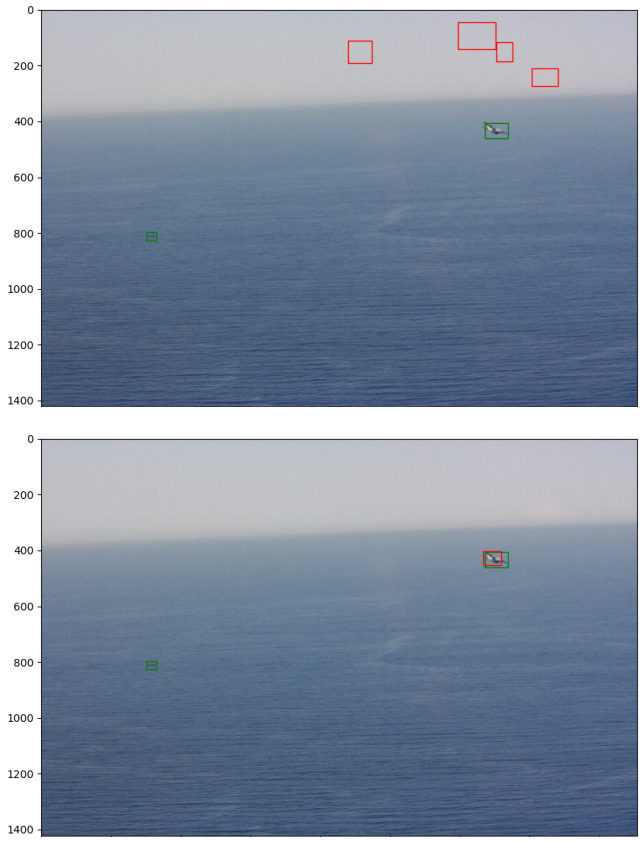


Abbildung 2: Vergleich von Basic (oben) und PHOT auf Moonbird

Abbildung 3: große Schiffe sorgen für mehrfache Detektionen von Phot auf Moonbird

Nur auf den Moonbird Trainingsdaten führt erstaunlicherweise der simple Basic Detector dazu, dass am wenigsten Bilder mit Booten übersehen werden.

Laufzeitanalyse. Ein klarer Vorteil klassischer Bildverarbeitungsmethodiken gegenüber Deep-Learning-Modellen ist meist die Laufzeit der Berechnungen. Dies ist für den Anwendungsfall Bootserkennung von einem zivilen Seenotrettungsboot aus oder eventuell gar direkt an Bord der Drone von hoher Wichtigkeit. Wir vergleichen die Laufzeit, die unsere Algorithmen benötigen, um

Detektor	1. Frage	2. Frage
FFT Detector	20	11
Sobel Detector	191	23
Basic Detector	362	17

Detektor	1. Frage	2. Frage
FFT Detector	72	33
Sobel Detector	91	36
Basic Detector	77	25

(a) Trainingsdatensatz, links Bodensee, rechts Moonbird

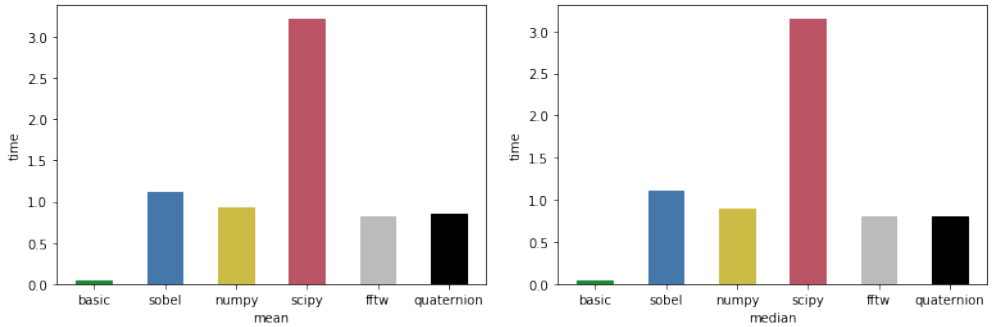
Detektor	1. Frage	2. Frage
FFT Detector	36	3
Sobel Detector	125	3
Basic Detector	601	4

Detektor	1. Frage	2. Frage
FFT Detector	21	8
Sobel Detector	36	19
Basic Detector	25	9

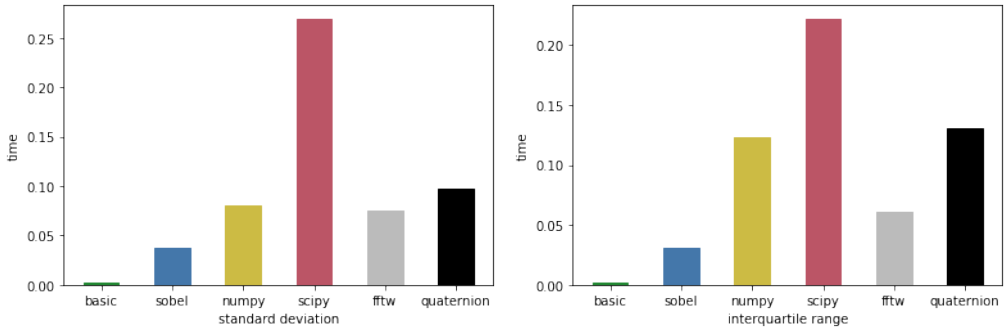
(b) Testdatensatz, links Bodensee, rechts Moonbird

Tabelle 3: Performance der Detektoren auf Basis der Bilder.

ein Bild auf einem mittelmäßigen Laptop zu verarbeiten. Um für etwaige zufällige Einflüsse zu kontrollieren, führen wir mehrere Durchläufe auf kleinen Datensätzen durch. Die Ergebnisse sind in Abbildung 4 dargestellt.
Die Laufzeitanalysen zeigen deutlich, wie schnell der Basic-Detektor, im Vergleich zu den Detektoren arbeitet, die zusätzlich zur Binärisierung eine Transformation des Bildes durchführen müssen.



(a) Lagemaße



(b) Streuungsmaße

Abbildung 4: Statistiken zur Laufzeit verschiedener Methodiken.

Limitierungen der Methode. Wie bereits angedeutet funktioniert die PHOT insbesondere dann nicht besonders gut, wenn noch weitere Effekte im Bild nur lokal auftreten. Dies können etwa die Reflektionen der Sonne sein, wie sie in Abbildung 5 zu sehen sind. Generell erzielen die verschiedenen Methodiken nur dann eine gute Performance, wenn sie zuvor auf den jeweiligen Datensatz optimiert wurden. Während des Entwicklungsprozesses fanden wir eine sehr niedrige inter-Datensatz Generalisierbarkeit der optimalen Parameterkombinationen. Dies, sowie die nach wie vor nicht zufriedenstellende qualitative Performance, sind sehr drastische Limitierungen hinsichtlich eines Einsatzes außerhalb von Laborbedingungen.

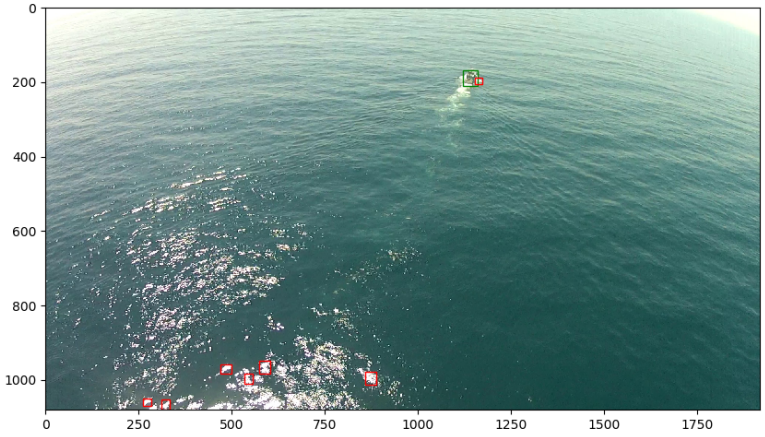


Abbildung 5: Sonnenreflektion bei Seagull sorgt für falschen Positivklassifizierungen des Phot-basierten Detektors

6 Diskussion

Die Erkennung von Booten in Seenot auf Dronenaufnahmen ist keine triviale Aufgabe. Wir untersuchen im Wesentlichen, ob die Phase-Only Transform als Methodik der Anomalieerkennung in diesem Kontext geeignet ist. In der Tat erzielen wir auf unterschiedlichen Datensätzen, die die Aufgabenstellung approximieren sollen gute Ergebnisse. Generell liefert unser PHOT-Detektor sowohl bessere Ergebnisse als unser naiver Basic-Detektor als auch der Sobel-Detektor. Je nach Datensatz unterscheiden sich Recall und Precision sehr stark. Wir erzielen aber stets mindestens 40% Recall und knapp 15% Precision, mit Ausnahme vom Basic- und Sobel-Detektor auf dem Seagull-Datensatz. Insbesondere limitiert wird die Precision hierbei durch andere lokale Anomalien auf den Bildern, wie etwa Sonnenreflektionen oder durch die Schiffe aufgeschäumte Wellen. Dies könnte ein Ansatzpunkt sein, mittels anderer Methodiken *False Positives* herauszufiltern.

Gleichzeitig zeigt unsere Analyse auch auf, dass die PHOT nicht geeignet ist, als hauptsächliches System zur Booterkennung verwendet zu werden. Es werden Boote übersehen, was, da es um Menschenleben geht, nicht zulässig ist. Möglicherweise wird diese Schwäche abgemildert durch die Tatsache, dass das selbe Boot oft auf mehreren Dronenaufnahmen zu sehen ist, übersieht man es auf einer, so besteht die Möglichkeit, dass es auf einem anderen Bild vorhergesagt wird. Dies haben wir jedoch nicht untersucht.

Nichtsdestotrotz bietet sich die PHOT für zwei Anwendungsfälle im Bereich der zivilen Seenotrettung an. Zum Einen kann diese zur Vorsortierung des Bildmaterials vor einer manuellen Sichtung genutzt werden. So wird der Prozess, Boote in Seenot zu entdecken, womöglich deutlich beschleunigt. Zum Anderen bietet sich die PHOT an, direkt auf einem eingebetteten System auf der Drone, etwa einem Raspberry Pi, zu laufen,da sie einen vergleichsweise niedrigen Rechenaufwand hat. Unsere Analyse zeigt eine Performance von etwa einem verarbeiteten Bild pro Sekunde (1 FPS) auf einem etwas älteren handelsüblichen Laptop, ohne Multiprocessing zu verwenden. Eventuell wird diese Geschwindigkeit auf einem eingebetteten System etwas langsamer sein. Sollte sie zu niedrig fallen, so könnte Multiprocessing eventuell aushelfen. So liefern etwa vier Kerne mit je 0,25 FPS effektiv auch 1 FPS Leistung.

Am rechenintensivsten sind die FFT und die inverse FFT während der PHOT. Betrachtet man verschiedene Implementationen, so empfiehlt sich nach unserer Analyse die FFTW als bewährter Generalist unter den FFTs. Eventuell ist ihre Performance noch zu schlagen. Eine optimale Implementation für den Anwendungsfall PHOT im Rahmen der Bootserkennung zu finden, ist

jedoch schwierig. Am Beispiel eines Algorithmus für dünnbesetzte Algorithmen zeigen wir dies. Unsere sparse FFT führt leider nicht zu Laufzeitverringerungen, da der essentielle Bestandteil des Zerschneiden und Zusammenfügens viel Zeit kostet und die Bilder schlicht nicht wirklich dünnbesetzt sind.

Eine Limitierung unserer Ergebnisse ist, dass die betrachteten Datensätze die Fragestellung nur approximieren. Wir können keine Aussagen darüber treffen, wie gut die PHOT in der Realität funktioniert. Insbesondere stellen wir fest, das je nach Datensatz unterschiedliche Parameter nötig sind, um brauchbare Ergebnisse zu erhalten. Demnach ist es essentiell, die PHOT stets auf den jeweiligen Einsatzfall zu kalibrieren. Hier stellt sich im Wesentlichen die Frage, wie viel Aufwand dieser Prozess benötigt. Es kann sein, dass nur einmal zu Beginn einer Mission auf dem Mittelmeer kalibriert werden muss, wahrscheinlich ist aber, dass je nach Wetter und Tageszeit unterschiedliche Parameter nötig sind, was die PHOT praktisch nicht nutzbar machen würde.

Wir experimentieren mit der QPHOT als Variante, die die Farbinformation der Bilder mit verwendet. Allerdings liefert unsere Implementation der QPHOT, bis auf eine zusätzliche falsche Positivklassifierzung auf den Testdaten des Bodensee Datensatzes, genau die gleichen Ergebnisse wie die graustufenbasierte PHOT. Es wäre spannend, mathematisch zu untersuchen, ob unser QPHOT-Algorithmus äquivalent zu unserem PHOT-Algorithmus ist oder tatsächlich zu anderen Ergebnissen führen könnte.

Anomalieerkennung im Allgemeinen und die PHOT im Spezifischen sind vielversprechende Ansätze für die Erkennung vom Booten in Seenot auf Dronenaufnahmen. Dies können wir nach ersten Experimenten sagen. Um Aussagen über die Praxistauglichkeit treffen zu können, ist jedoch noch viel Arbeit nötig. Insbesondere sind andere Methodiken, etwa Deep-Learning-Modelle, mit der PHOT zu vergleichen. Ebenfalls sollte eine graphische Oberfläche sowie eine Implementierung für eingebettete Systeme entwickelt werden. Schließlich kann nur ein Praxistest endgültige Aussagen liefern. Die NRO SearchWing startet im März 2020 einen Pilotversuch während einer Mission des zivilen Seenotrettungsschiffes Alan Kurdi der NRO Sea-Eye e.V.. Dieser könnte Ansatzpunkt für weitere Studien sein.

DANKSAGUNGEN

Wir danken Gerlind Plonka-Hoch und Jochen Schulz für hilfsbereite Betreuung des Projektes und Anna Katharina Schuldt für ihre tatkräftige Unterstützung in einer frühen Phase des Projektes. Diese Arbeit wäre nicht möglich gewesen ohne das großartige Engagement zahlreicher Ehrenamtlicher bei der NRO SearchWing, die etwa den Bodensee Datensatz bereitgestellt hat.

LITERATUR

Dror Aiger and Hugues Talbot. The phase only transform for unsupervised surface defect detection. In *Emerging Topics In Computer Vision And Its Applications*, pp. 215–232. World Scientific, 2012. URL `https://perso.esiee.fr/~aigerd/phot.pdf`.

Xiaolong Bai, Yuming Fang, Weisi Lin, Lipo Wang, and Bing-Feng Ju. Saliency-based defect detection in industrial images by using phase spectrum. *IEEE Transactions on Industrial Informatics*, 10 (4):2135–2145, 2014. URL `https://ieeexplore.ieee.org/abstract/document/6906292/`.

Michael Boyle. *quaternion documentation*, 2019. URL `https://quaternion.readthedocs.io/`.

Todd A Ell and Stephen J Sangwine. Hypercomplex fourier transforms of color images. *IEEE Transactions on image processing*, 16(1):22–35, 2006. URL `https://ieeexplore.ieee.org/abstract/document/4032812`.

Matteo Frigo and Steven G Johnson. Fftw: An adaptive software architecture for the fft. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Pro-*

cessing, ICASSP'98 (Cat. No. 98CH36181), volume 3, pp. 1381–1384. IEEE, 1998. URL https://ieeexplore.ieee.org/abstract/document/681704/.

Henry Gomersall. *pyFFTW documentation*, 2016. URL https://pyfftw.readthedocs.io/.

Chenlei Guo, Qi Ma, and Liming Zhang. Spatio-temporal saliency detection using phase spectrum of quaternion fourier transform. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE, 2008. URL https://ieeexplore.ieee.org/abstract/document/4587715.

John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007. URL https://aip.scitation.org/doi/abs/10.1109/MCSE.2007.55.

Mario Monteiro Marques, Pedro Dias, Nuno Pessanha Santos, Vitor Lobo, Ricardo Batista, D Salgueiro, A Aguiar, M Costa, J Estrela da Silva, A Sergio Ferreira, et al. Unmanned aircraft systems in maritime operations: Challenges addressed in the scope of the seagull project. In *OCEANS 2015-Genova*, pp. 1–6. IEEE, 2015. URL http://vislab.isr.ist.utl.pt/wp-content/uploads/2016/02/2015_oceans.pdf.

John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965. URL https://academic.oup.com/comjnl/article/7/4/308/354237.

Gerlind Plonka-Hoch, Daniel Potts, Gabriele Steidl, and Manfred Tasche. *Numerical Fourier Analysis*. Springer, 2018a.

Gerlind Plonka-Hoch, Katrin Wannenwetsch, Annie Cuyt, and Wen-shin Lee. Deterministic sparse FFT for m-sparse vectors. *Numerical Algorithms*, 78(1):133–159, 2018b. URL https://link.springer.com/article/10.1007/s11075-017-0370-5.

Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, and Victor Eruhimov. Real-time computer vision with opencv. *Communications of the ACM*, 55(6):61–69, 2012. URL https://dl.acm.org/doi/10.1145/2184319.2184337.

Jean Serra. *Image analysis and mathematical morphology*. Academic Press, Inc., 1983.

Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. A talk at the Stanford Artificial Intelligence Project (SAIL) Project, 1968. URL https://www.researchgate.net/publication/239398674_An_Isotropic_3x3_Image_Gradient_Operator.

Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985. URL https://www.sciencedirect.com/science/article/abs/pii/0734189X85900167.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, pp. 1–12, 2020. URL https://www.nature.com/articles/s41592-019-0686-2.

Therese von Wulffen. *Ein deterministischer FFT-Algorithmus für dünn besetzte Vektoren*. Bachelorarbeit, Georg-August-Universität Göttingen, 2019.

Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011. URL https://aip.scitation.org/doi/abs/10.1109/mcse.2011.37.