Universität Rostock

Traditio et Innovatio

Entwicklung eines Smart Labeling-Tools für automatische Objekt-Segmentierung

Name:	Pfeil
Vorname:	Lennard
Matrikel-Nr.:	216203764
Studiengang:	Master Informatik
E-Mail:	lennard.pfeil@uni-rostock.de
Lehrstuhl:	Mobile Multimedia Information Systems
Institut:	Institut für Visual and Analytic Computing
Fakultät:	Fakultät für Informatik und Elektrotechnik
Gutachter:	Prof. Dr.-Ing. Thomas Kirste
Zweitgutachter	Priv.-Doz. Dr.-Ing. habil. Dirk Joachim Lehmann
Betreuer:	Dr. rer. nat. Sebastian Bader
Fachsemester:	3
Abgabedatum:	29.11.2021

Masterarbeit

Universität Rostock

Zusammenfassung

Im Rahmen dieser Masterarbeit wurden unterschiedliche Unterstützungshilfen für das Labeling-Tool der *IAV GmbH* untersucht. Dafür wurden Segmentierungsverfahren umgesetzt und evaluiert, die bei möglichst gleichbleibender Qualität, Zeit während des Labelvorgangs sparen sollen. Zuerst wurden Ansätze untersucht, welche den Nutzer nur mithilfe der Bildinformationen und möglichen Nutzerinteraktionen unterstützen soll. Dabei konnte mithilfe aller Verfahren ein Zeitgewinn gegenüber dem manuellen Labeln feststellen. Außerdem wurden die entstandenen Labels mit der IoU mit manuell erstellten Labels verglichen. Hierbei konnte der Watershed-Algorithmus, neben dem höchsten Zeitgewinn von 68%, zusätzlich auch die höchste Übereinstimmung mit den manuellen Labels erreichen. Aufgrund von Ausreißern zeigte sich bei der visuellen Betrachtung, dass der Nutzer je nach Anwendungsfall entscheiden muss, ob die Ergebnisse mithilfe der Wasserscheidentransformation zufriedenstellend sind.
Des Weiteren wurden Segmentierungsverfahren untersucht, bei denen ein Modell durch bereits gelabelte Daten trainiert wird und anschließend dem Nutzer Labels auf bisher ungesehenen Daten vorschlägt. Im Anschluss kann das Label außerdem mithilfe von Korrekturtools verbessert werden. Bei der Evaluation zeigte sich, dass die CNNs Mask R-CNN und Yolact gute Label vorhersagten. So reichten bereits unter 100 Trainingsbilder aus, um ausreichende Masken vorherzusagen, allerdings zeigte sich auch, dass die Genauigkeit der Label mit wachsender Trainingsdatensatzgröße zunimmt. Als weiteren wichtigen Faktor erwiesen sich die benutzten vortrainierten Gewichte. Es zeigte sich, dass das Training mit kleinen Datensätzen nur erfolgreich war, wenn die gesuchte Klasse bereits mit vortrainiert wurde.

Inhaltsverzeichnis

Abkürzungsverzeichnis

RGB steht für die Farben Rot, Grün und Blau

IoU Intersection over Union (Jaccard index)

SVM Support Vector Machine

CNN Convolutional Neural Network

ReLU Rectified Linear unit

RoI Region of Interest

FPS frames per second (Bilder pro Sekunde)

Abbildungsverzeichnis

Tabellenverzeichnis

I Einleitung

Die Objekterkennung ist ein stetig wachsendes Themengebiet, welches ein Verfahren beschreibt, bei dem diverse Objekte aus visuellen, akustischen oder anderen physikalischen Signalen erkannt werden. Die zunehmenden Anwendungsbereiche erfordern eine immer höhere Genauigkeit. In der Bilderkennung konnte dies unter anderem durch die Weiterentwicklung bisheriger Ansätze erreicht werden. So ist in Abbildung 1 zu erkennen, wie mit der Zeit immer bessere Ergebnisse an unterschiedlichen Benchmark-Datensätzen erreicht werden konnten.

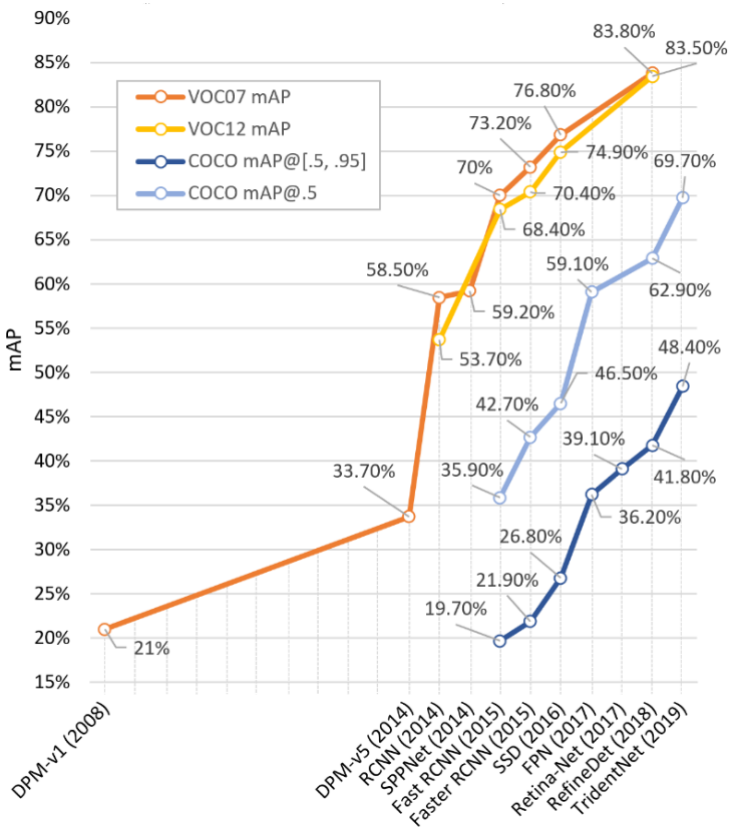

Abbildung 1: Genauigkeit von diversen Ansätzen über die Zeit [1]

Neben verbesserter Software und höherer Rechenleistung durch leistungsstärkere Hardware, sorgt auch die steigende Anzahl an verfügbaren Daten für immer besser werdende Vision-Recognition-Systeme. Für das Training der visuellen Objekterkennung müssen im Regelfall vordefinierte Merkmale der Bilddaten bereitgestellt werden. Dabei handelt es sich um Objektmarkierung auf einem Bild, die meistens als Koordinaten angegeben werden.

Ziel der Arbeit

Das Ziel dieser Arbeit ist es, unterschiedliche Ansätze zur Unterstützung der Objekt-Segmentierung zu untersuchen, mit denen das vorhandene Framework, welches in Abschnitt III näher erläutert wird, erweitert werden kann, um den Grundaufwand des Labeling-Prozesses, bei gleichbleibender Qualität, zu verringern.

II Stand der Technik

In diesem Abschnitt werden die aktuellen Kenntnisse und Ansätze beschrieben, wie Bilddaten segmentiert werden.

II.1 Literaturübersicht

Im folgenden Abschnitt wird die verwendete Literatur dargestellt. Bei der Recherche wurden anfangs 63 Paper aufgrund ihres Titels herausgesucht. Anschließend wurden sie durch das Lesen des *abstracts* und des gesamten Dokumentes schrittweise gefiltert, um relevante und replizierbare Ansätze herauszufinden. So war das primäre Ziel, möglichst allgemeingültige Ansätze für automatisches, bzw. halbautomatisches Labeln zu finden. Dafür wurden mehrere Ansätze aussortiert, die nur ein spezielles Problem thematisierten. Beispielsweise beschreiben [12] und [13] sehr hilfreiche Labeling-Tools, die aber nur visuelle Daten aus Fahrzeugen benutzen und somit aufgrund des eingeschränkten Anwendungsbereiches für diese Arbeit eher uninteressant sind. Weiterhin zeigte sich, dass viele Paper zwar öffentliche Datensätze, aber keinen Code veröffentlicht haben, was eine Reproduktion schwierig gestaltete.

A Color Image Segmentation algorithm Based on Region Growing [14]

Tang stellt in seinem Paper vor, wie Bilder mithilfe des Region Growing-Algorithmus in Kombination mit der Wasserscheidentranformation segmentiert werden können. Für den Region Growing-Algorithmus stellt er dabei die folgenden Kernpunkte heraus:

- Auswählen der Start-Pixel für den Algorithmus
- Bestimmung, wann ein benachbarter Pixel zur Region gehört
- Regeln oder Bedingung, wann der Wachstum der Region stoppt

Bei der Auswahl der Start-Pixel wird die Wasserscheidentransformation genutzt, um initiale Segmentierungen zu erstellen. Daraufhin werden Regionen automatisch ausgewählt, um einen angepassten Region Growing-Algorithmus von diesen Regionen zu starten. Zwar gibt Tang an, dass dieses Verfahren bessere und schnellere Ergebnisse liefert, allerdings wurden weder reproduzierbarer Code noch nachvollziehbare Ergebnisse veröffentlicht.

A new marker-based watersched algorithm [15]

In diesem Paper beschreiben die Autoren, wie sie Bilder mithilfe der Wasserscheidentransformation segmentieren. Als Vorraussetzung wird ein Graustufenbild gefordert, bei dem jeder Pixelwert eine Steigerung oder eine Distanz zu benachbarten Pixeln repräsentiert. Weiterhin sollen Markierungen für den Algorithmus durch Nutzerinteraktionen entstehen. Anschließend wird eine abgeänderte, nicht veröffentlichte Version der Wasserscheidentransformation verwendet, um das Bild zu segmentieren. Als Vorteile ihrer abgeänderten Version nennen die Autoren weniger

Speicherbedarf und geringe Zeitkomplexität. Für diese Arbeit kann zwar nicht die in C angepasste Version genutzt, allerdings kann hierfür trotzdem die konventionelle Version der Wasserscheide getestet und gegebenenfalls angepasst werden.

Snakes on the watershed [16]

Jaesang Park und Keller stellen in ihrem Paper einen Ansatz zum Segmentieren von weißen Blutzellen vor. Der „watersnake“ genannte Algorithmus ist eine Kombination aus dem Snake-Algorithmus und der Wasserscheidentransformation. Diese Kombination lässt sich auf das Problem zurückführen, dass der Snake-Algorithmus anfällig für lokale Minima ist. Deshalb sollen möglichst unnötige Punkte eliminiert werden, damit nur globale Minima erreicht werden. Als Technik soll hierfür die Wasserscheidentranformation dienen. In der Auswertung des Ansatzes zeigt sich, dass „watersnake“ zwar besonders in wenig verrauschten Bereichen dichter an der Ground Truth ist, während in stark verrauschten Bereichen die Wasserscheidentransformation von „watersnake“ keine zufriedenstellenden Ergebnisse erreichen konnte. Als weiterer Nachteil stellt sich die Laufzeit des Ansatzes heraus, da diese mehr als doppelt so hoch, wie von den verglichenen Ansätzen war.

A Semi-Automatic Video Labeling Tool for Autonomous Driving Based on Multi-Object Detector and Tracker [12]

Wang et al. beschreiben in ihrem Paper die Entwicklung eines halbautomatischen Labeling-Tools für autonome Fahrzeuge. Dabei sollen in einer Videosequenz mithilfe eines vortrainierten Faster-RCNN[17] Modells Objekte erkannt und durch einen MDP Tracker nachbearbeitet werden. Anschließend kann jemand in dem Labeling-Tool die Markierungen überprüfen und mögliche Fehler ausbessern. Dadurch soll der Label-Prozess effizienter gestaltet werden, was in dem Paper durch einen Zeitgewinn von 32% anhand des KITTI Datensatzes bekräftigt wurde.

Deep Learning Based Automatic Video Annotation Tool for Self-Driving Car [13]

In diesem Paper beschreiben die Autoren, wie man aus Videodaten Labels für Fahrzeuge, Zweiräder und Fußgänger generiert, um diese für die Entwicklung von selbstfahrenden Fahrzeugen zu benutzen. Für die Erkennung wurden YOLO[18] und Retinanet-50 benutzt. Dabei sollen neben dem eigentlichen Label auch diverse Eigenschaften zur Bewegung, Richtung und Sicht mit erkannt und aufgenommen werden. Dafür wurden in dem Paper die CNN-Architekturen Resnet-50 und VGG-19 mit einander verglichen. Retinanet-50 erreichte mit einer Genauigkeit von 82% das beste Ergebnis für die Objekterkennung und Resnet-50 wurde mit 97% Genauigkeit für die Klassifizierung gewählt. Weiterhin wurde der Deep Sort Algorithmus für die Objektverfolgung benutzt. Das gesamte System war in einem Test 1200-mal schneller als manuelles Labeln.

Improved techniques for automatic image segmentation [19]

In dem Artikel von Gao et al. wird beschrieben, wie man Bilder vereinfachen kann, um anschließend Objekte zu segmentieren. Für Die Vereinfachung behandeln sie mehrere Ansätze, wie die Umwandlung des Farbraumes von RGB zu Lab für eine bessere Farbunterscheidung oder die Anwendung von morphologischen Operationen, um das Bild zu vereinfachen. Das Ziel der unterschiedlichen Techniken ist ein schwarz-weiß Bild, welches nur noch die Konturen von Objekten enthält, die anschließend durch eine Wasserscheidentransformation nachverarbeitet werden können.

A review on automatic image annotation techniques [20]

Zhang et al. stellen in ihrem Paper den Stand von unterschiedlichen Techniken zur Objektsegmentierung dar. Dafür behandeln sie zuerst diverse, nutzbare Features, wie unterschiedliche Farbräume, Texturen oder Formen. Anschließend folgen Techniken zum automatischen Labeln von Bildern. Als erstes wird ein Klassifikator vorgestellt, welcher aus mehreren SVMs besteht, die jeweils die Wahrscheinlichkeit einer Klasse vorhersagen soll. Somit kann das Bild der Klasse mit der höchsten Wahrscheinlichkeit zugeordnet werden. Weiterhin beschreiben die Autoren, wie Objekte mit einem bestehenden neuronalen Netz erkannt werden können. Als letzte Technik wird die Klassifikation mit einem Entscheidungsbaum beschrieben, der Objekte beispielsweise aufgrund der Farbe, Form oder Textur unterscheiden kann.

Road Extraction Using SVM and Image Segmentation [21]

Das Paper von M. Song beschreibt die Anwendung einer support vector machine (SVM) zur Segmentierung von Straßen aus Luftaufnahmen. Nach der Klassifizierung der Straßen wurde als zweiter Schritt eine Nachbearbeitung mithilfe des Region Growing-Algorithmus durchgeführt, um Falsch-Positive Segmente zu entfernen. Dabei wurden Informationen über die Form mit in die Bedingung des Verfahrens eingeführt, da Straßen immer als lange dünne Objekte interpretiert wurden. Dieser Ansatz konnte eine Genauigkeit von 99 % erreichen. Dagegen wurde eine Klassifikation mit Gaussian maximum likelihood(GML) verglichen, welche allerdings nur eine Genauigkeit von 96 % erzielte.

Mask R-CNN [5]

He et al. beschreiben in dem gleichnamigen Paper das Framework Mask R-CNN, welches eine Erweiterung von Faster R-CNN darstellt. Dabei wird die Erkennung von interessanten Bereichen durch Faster R-CNN genutzt und um einen weiteren Ausgang erweitert, der eine Objektmaske zurückgeben soll. Im Vergleich konnte das Framework bessere Ergebnisse erzielen als die Gewinner der COCO segmentation challange 2015 und 2016[10].

Pointly-Supervised Instance Segmentation [22]

Cheng et al. erklären, wie der Prozess der Objektsegmentierung beschleunigt werden kann. Dafür werden in einer Bounding Box zehn zufällige Punkte ausgewählt und von einer Person dem Objekt oder dem Hintergrund zugeordnet. Mithilfe von einem Mask R-CNN Modell, welches auf Resnet-50 basiert, kann aus den Punkten das Objekt deutlich schneller segmentiert werden, als es durch eine manuelle Maske möglich wäre.
Bei einer tieferen Analyse sorgte allerdings die Anforderung des Betriebssystems (Linux oder MacOS) für Komplikationen, sodass es für diese Arbeit aufgrund des erheblichen Mehraufwandes und der vorhandenen Betriebssysteme der *IAV GmbH* nicht berücksichtigt wird.

Non-local Neural Networks [23]

Eine weitere Veröffentlichung zu Mask R-CNN ist das Paper von Wang et al., indem sie erklären, wie Mask R-CNN um nicht-lokale Operationen erweitert werden kann. Während bei convolutional Operationen nur die lokale Nachbarschaft mit in die Berechnung einfließt, gehen bei nicht-lokalen Operationen alle Features gewichtet mit ein. Diese Erweiterung hat zwar einen höheren Berechnungsaufwand, allerdings weisen die Autoren auch eine höhere Genauigkeit, im Vergleich zu anderen Varianten von Mask R-CNN, vor.

Object Class Segmentation using Random Forests [24]

Das Paper von Schroff et al. behandelt den Random Forest Klassifikator zur Objektsegmentierung in Bildern. Dabei wird, neben der effizienten Berechnung bei Training und Vorhersage, die Variabilität der Features, wie Farbe, Textur oder Form, als wichtigen Vorteil des Klassifikators genannt. Weiterhin werden unterschiedliche Modelle, wie *single-histogram class models(SHCM)* oder *Conditional Random Field (CRF) model*, erläutert und beschrieben, wie man diese in einen Random Forest Klassifikator einbauen kann. Für den Datensatz mit neun unterschiedlichen Klassen konnte so eine Genauigkeit von 87,2 % mit dem CRF-Modell erreicht werden.

YOLACT Real-time Instance Segmentation[6]

Das gleichnamige Paper von Bolya et al. beschreibt das Framework Yolact. Dies ist eine Weiterentwicklung von Yolo[18], bei dem es sich um ein neuronales Netz zur Objekterkennung mithilfe von Bouding Boxen handelt. Dieses CNN wurde so erweitern, dass es zusätzlich eine Vorhersage einer Maske des gefundenen Objektes ermöglicht. Als Hauptargument nennen die Autoren die Performance des Frameworks, da es die Möglichkeit bietet, Objekte in Echtzeit[1] zu segmentieren.

[1] Bearbeitung von mindestens 30 Bildern pro Sekunde

II.2 Übersicht alternativer Labeling-Tools

In der Bildverarbeitung sind Labeling-Tools bereits ein weit verbreitetes Hilfsmittel, um möglichst schnell Bilder zu labeln. Aufgrund von unterschiedlichen Labelarten (Unterabschnitt IV.2), diversen Speicherformaten und zusätzlichen Erweiterungen haben sich zahlreiche Tools entwickelt, wovon einige im folgenden Abschnitt beschrieben werden.
Viele Tools, wie Labelme[25] oder LabelImg[26] sind einfache Labeling-Tools, welche als Weboberfläche oder lokales Tool nur die Möglichkeit bieten, mit unterschiedlichen Labelarten Objekte in Bildern zu markieren und in diversen Formaten abzuspeichern. Die bekanntesten Speicherformate sind *COCO JSON*, bei dem alle möglichen Label eines Datensatzes in einer einzelnen *json*-Datei gespeichert werden, und *PASCAL VOC XML*. Bei diesem Format wird für jedes gelabelte Bild jeweils eine *xml*-Datei erstellt, welche die Labelinformationen enthalten.
Weiterhin gibt es mit beispielsweise DeepLabel[27] oder OpenLabeling[28] Labeling-Tools, welche zusätzliche Erweiterungen zum automatischen Labeln bieten, indem sie Bounding Boxen durch CNNs, wie Faster R-CNN oder Yolo, vorhersagen können.
Außerdem existieren mit SuperAnnotate[29] und V7[30] komplexere Labeling-Tools, welche selbst segmentierte Label mithilfe von CNNs vorhersagen können. Zusätzlich bieten beide Tools die Möglichkeit eine Label-Pipeline zu erstellen, um einen kundenspezifischen Ablauf zu erstellen. Dieser kann manuelle Labeling-Prozesse bestimmten Personen zuweisen oder auch mithilfe bereits gelabelter Bilder ein CNN trainieren und anschließend anwenden. Hierfür stehen bei V7 Grafikkarten-Server für das Training und die Vorhersage bereit. Für die Segmentierung benutzen sie ein eigenes CNN mit dem Namen V7 Neurons, welches nach eigenen Angaben genauer als Mask R-CNN und Yolact sein sollen[2] und stark vortrainierte Gewichte nutzt.

II.3 Fazit der Literaturanalyse

Bei der Literaturanalyse zeigten sich diverse Ansätze zur Objektsegmentierung. Grundlegend kann dabei zwischen Ansätzen mit und ohne Vorwissen unterschieden werden.
Bei Techniken ohne Vorwissen, wie Region Growing[14], Watershed[15] oder Snake[16], werden Objekte nur mithilfe von Bildinformationen und Nutzerinteraktionen segmentiert. Da bei dem typischen Labelprozess anfangs nur die Bilddaten ohne weitere Informationen oder bereits gelabelten Bildern vorhanden sind, wird der erste Teil dieser Arbeit sich mit den Ansätzen ohne Vorwissen beschäftigen. Hierbei muss untersucht werden, wie gut die Verfahren aus der Literatur auf unterschiedliche Daten mit welchem Zeitgewinn anwendbar sind.
Anschließend wird der zweite Teil Techniken mit Vorwissen behandeln. Hierbei wird sich mit Modellen beschäftigt, die durch bereits gelabelte Bilder Wissen erlangen, um neue Label auf bisher ungesehenen Daten zu erzeugen. Im Gegensatz zu den Verfahren ohne Vorwissen, sollten Segmentierungsverfahren, wie Mask R-CNN[5, 22, 23], Yolact[6], SVM[21, 31] oder Random Forest[24, 32, 33] zusätzlich danach evaluiert werden, wie groß der Trainingsdatensatz sein muss. So ist davon auszugehen, dass die neuronalen Netze Mask-RCNN und Yolact einen sehr langen Trainingszeitraum in Anspruch nehmen und deshalb voraussichtlich nicht in Frage kommen, solange das Training nicht ausgelagert wird. Allerdings lassen [24, 32, 33] darauf schließen, dass es gute Ergebnisse liefern wird und somit einen guten Vergleich erlaubt. Somit muss hier evaluiert

[2] Benchmarks auf dem COCO-Datensatz

werden, ob es neuronale Netze gibt, welche zufriedenstellende Ergebnisse liefern ohne so viele Daten für ein Training zu benötigen, dass der Datensatz für den eigentlichen Anwendungszweck bereits groß genug ist.
Währenddessen ist bei Random Forest zwar eine deutlich niedrigere Trainingszeit zu erwarten, allerdings nutzen viele Paper[32, 33], die gute Ergebnisse erreichten, Kameras mit Tiefeninformationen oder hatten einen sehr speziellen Anwendungszweck. Für diese Arbeit werden allgemeingültige Ansätze untersucht, wodurch spezielle Anwendungszwecke oder Tiefeninformationen keinen wirklichen Mehrwert haben.
Das Ziel hierbei ist mindestens genauso gute Label erzeugen zu können, wie es mit den Segmentierungsverfahren ohne Vorwissen der Fall ist. Im Optimalfall wird hierdurch die benötigte Zeit und die Nutzerinteraktion des Labelprozesses minimiert.

III Anforderungsanalyse

Für diese Arbeit steht als Grundlage ein Labeling-Tool der *IAV GmbH* zur Verfügung, welches Möglichkeiten zur Erzeugung von manuellen Label in Bilddaten bereitstellt.
Hierfür können Bilddatensätze als Verzeichnis geladen werden und anschließend einzelne Bilder geöffnet werden. Zum Labeln stehen mit *Bounding Box*, *Lasso* und *Polygonal Line* unterschiedliche, manuelle Methoden zur Verfügung, die je nach Anwendungszweck benutzt werden können. Weiterhin kann mithilfe von inneren und äußeren Linien eine Verdeckung oder ein mehrteiliges Objekt eingezeichnet werden. Das Tool bietet nach dem Labeln die Möglichkeit, die Labels in einem *IAV*-eigenen Format zu speichern, welche beim nächsten Öffnen erneut geladen werden können.

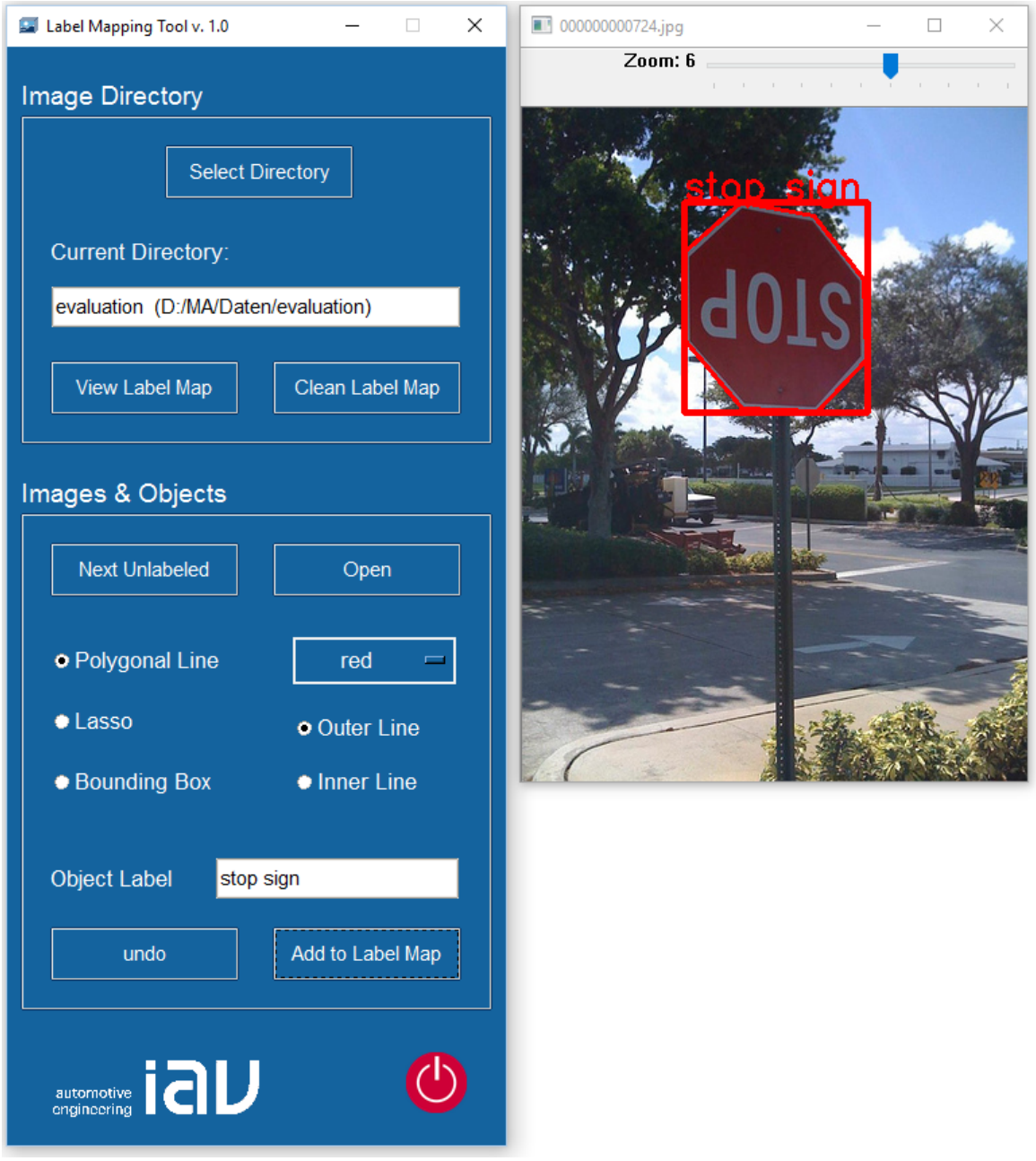

Abbildung 2: Screenshot des Labeling-Tools der *IAV GmbH*

In dieser Arbeit sollen zur Erweiterung des Labeling-Tools unterschiedliche Ansätze zur Unterstützung der Objekt-Segmentierung untersucht werden. Das Ziel dabei ist es, den Zeitaufwand des stupiden Labeling-Prozesses bei gleichbleibender Qualität zu verringern. Hierfür sind instantane, allgemeingültige Verfahren zu evaluieren, die nur mithilfe der Bildinformationen und möglichen Nutzerinteraktionen die Labels der gesuchten Objekte vorschlagen können. Diese Verfahren werden im folgenden Teil der Arbeit als Segmentierungsverfahren ohne Vorwissen bezeichnet, da hierbei nur die Bildinformationen und Nutzerinteraktionen bereit stehen.
Anschließend soll untersucht werden, ob bei größeren Datensätzen ein parallel oder vorher trainiertes Modell die nötige Labelunterstützung liefern kann, indem es Label vorhersagt. Dabei ist zu beachten, dass es sich um einen Datensatz einer oder mehreren gleichbleibenden Klassen handelt, wodurch das Modell auf bereits gelabelte Bilder trainiert werden kann, um anschließend mit diesem Wissen weitere Label auf bisher ungesehenen Bildern vorherzusagen. Bei diesen, als Segmentierungsverfahren mit Vorwissen bezeichneten, Methoden soll evaluiert werden ob und ab welcher Datensatzgröße sich eine Nutzung dieser Verfahren als sinnvoll erweist.
Die hierfür benutzten und von der *IAV GmbH* bereitgestellten Datensätze werden in Unterabschnitt IV.6 näher erläutert. Das gegebene Labeling-Tool bietet eine Grundlage für diese Arbeit, da bereits unterschiedliche Interaktionen, wie Einzeichnen oder Zoomen, und das Speichern von Labels implementiert wird und somit diese Schnittstelle genutzt werden kann.

IV Grundlagen

In dem folgenden Teil der Arbeit werden die grundlegenden Verfahren erläutert und die benutzten Datensätze dargestellt.

IV.1 Bilddarstellung

In der Bildverarbeitung werden Bilder üblicherweise als Matrix von einzelnen Pixeln interpretiert. Graustufenbilder können mithilfe einer Matrix dargestellt werden, bei der jeder Pixel einen einzelnen Zahlenwert besitzt. Dieser Wert liegt zwischen 0 und 255 [3] und bestimmt die Intensität des Grautons.

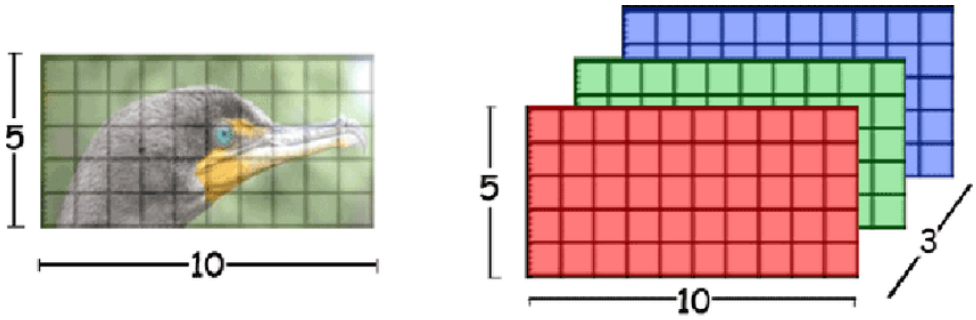

Abbildung 3: Bildrepräsentation nach [2]

Währenddessen werden, wie es in Abbildung 3 erkennbar ist, für Farbbilder drei Matrizen für die digitale Repräsentation benötigt. Der bekannteste Farbraum ist RGB, bei dem die Matrizen die Zusammensetzung der Pixel aus rot, grün und blau zeigen, wobei jeder Wert zwischen 0 und 255 liegt und dadurch die Intensität der jeweiligen Farbe bestimmt. Weitere bekannte Farbräume sind CMY (*Cyan, Magenta* und *Yellow*) oder HSV(Farbwert, Farbsättigung und Helligkeit)[34].

IV.2 Objektlabel

In der Bildverarbeitung gibt es unterschiedliche Labelarten, die im folgenden Abschnitt näher erläutert werden.

Bounding Boxen

Bounding Boxen (zu deutsch Begrenzungsrahmen) ist eine der meist genutzten Labelarten. Hierbei wird ein Objekt durch eine einfache, minimale Box auf dem Bild lokalisiert, welche das gesamte Objekt beinhaltet. Sie kann mithilfe von zwei Eckpunkten und gegebenenfalls einer Objektbezeichnung angegeben werden.

Polygone

Polygone können genutzt werden, wenn präzisere Label benötigt werden, als es durch Bounding Boxen möglich ist. Hierbei werden die Linien an die Objektgrenzen angepasst, sodass die

[3] 255 lässt sich mithilfe von einem Byte (8 Bit) darstellen

Polygone nur noch das Objekt beinhalten. Das Label kann anschließend als Liste von Konturpunkten angegeben werden. Polygone haben einen hohen Stellenwert für die diese Arbeit, da sie neben Masken sich für die Beschreibung der Objektsegmentierung eignen. Dabei werden ein oder mehrere Objekte aus einem Bild anhand der Objektkontur „ausgeschnitten“ und können mithilfe von Polygonen und einer Klassenbezeichnung angegeben werden. Eine Objektsegmentierung kann zwar ebenfalls als binäre Maske (Objekt und Hintergrund) angegeben werden, jedoch werden Polygone aufgrund des geringeren Speicherplatzbedarfs öfter benutzt, da hier nur wenige Koordinaten und keine ganzen Masken in der Größe des Originalbildes gespeichert werden müssen.

3D Boxen

3D Boxen sind eine Erweiterung der Bounding Boxen, wobei durch die dreidimensionale Box um das Objekt zusätzliche Tiefeninformationen genutzt und gespeichert werden. Das Speichern kann durch die Angabe von drei Koordinaten auf dem Bild erfolgen.

Semantische Segmentierung

Bei der Semantischen Segmentierung wird jedem Pixel des Bildes einer Klasse zugeordnet, sodass das Ergebnis eine Maske als Label des gesamten Bildes ist, bei der die Objekte einer Klasse nur durch einen bestimmten Farbwert dargestellt werden. Hierbei werden die Objekte, wie bei dem Labeln mit Polygonen, anhand ihrer Konturen gelabelt.

weitere Formen

Weiterhin existieren noch mehrere einfache Formen, wie Punkte, Linien, Kreise oder Ellipsen, die als Label verwendet werden können. Allerdings sind diese weniger verbreitet und haben spezielle Anwendungszwecke. So eignen sich Kreise beispielsweise besonders um runde Objekte zu labeln. Eine andere speziellere Labelart sind Linien, welche zum Beispiel für die Markierungen für Straßenspuren oder Roboterwege genutzt werden. Linien können, wie Polygone, als Liste von Punkten angegeben werden.[35, 36]

IV.3 Segmentierungsverfahren

Bei der Segmentierung im Zweidimensionellen wird eine größere Struktur in sinnvolle, dem Anwendungszweck entsprechende, miteinander verbundene kleinere Elemente aufgeteilt [37]. Dieses Zuordnungsproblem ist nicht hart definiert und somit vom Anwendungsbereich sowie von dem Nutzerinteresse abhängig. Der Label-Prozess beinhaltet meistens eine Segmentierung eines gesuchten Objektes, während der restliche Bereich als Hintergrund definiert wird.
Dies sorgt dafür, dass eine vollautomatische Segmentierung ohne zusätzliches Wissen über die gewünschten Segmente unwahrscheinlich ist. Allerdings gibt es Verfahren, die versuchen das zeitaufwendige, manuelle Segmentieren auf ein Minimum an Benutzerinteraktionen zu reduzieren.
Bei diesen Ansätzen aus der Bildanalyse kann dabei in pixelorientierte, regionenbasierte und kantenbasierte Verfahren unterschieden werden.

IV.3.1 Pixelorientierte Segmentierungsverfahren

Bei pixelorientierten Verfahren, wie dem globalen Schwellwertverfahren, handelt es sich um die einfachsten Methoden zur Einteilung eines Bildes. Bei diesem Beispiel werden alle Pixel mit einem Schwellwert verglichen und demnach zu einer Region zugeordnet.
Dieses Verfahren kann bei speziellen Anwendungsbereichen oder als Vorverarbeitungsschritt hilfreich sein. Bei realen Bildern führt es allerdings oft zu einer Übersegmentierung, wobei neben dem eigentlichen Objekt noch viele weitere Segmente aufgrund der homogenen Farbwerte der Region zugeordnet werden.

IV.3.2 Regionenorientierte Segmentierungsverfahren

Regionenorientierte Segmentierungsverfahren basieren auf der Annahme, dass ein Bild in Regionen unterteilt werden kann, welche homogene Eigenschaften besitzen. Dabei wird die Zuordnung von Pixeln zu einem bestimmten Segment durch die Homogenität in der Nachbarschaft entschieden.
Im Folgenden werden zwei Verfahren dieser Kategorie vorgestellt. Diese haben sich in der Literaturanalyse als interessant herausgestellt und werden in dieser Arbeit angewendet und verglichen werden.

Region growing (Regionenwachstum)

Das regionenorientierte Verfahren fasst homogene Bildelemente anhand der Differenz zwischen benachbarten Pixelwerten zu Regionen zusammen. Ausgehend von ein oder mehreren Anfangspunkten wird die Aufnahme benachbarten Punkte der Region solange überprüft bis kein Wachstum der Region mehr möglich ist. Wenn der Betrag der Differenz kleiner als ein festgelegter Grenzwert t ist, wird der benachbarte Punkt in die Region aufgenommen.

$$d(p,q) < t \tag{1a}$$

Für ein Graubild lässt sich d wie folgt berechnen:

$$d(p,q) = |p - q| \tag{1b}$$

Für diese Arbeit wurden Pixel aus dem RGB-Farbraum mithilfe der Euklidischen Distanz verglichen, wobei die Pixel in ihre drei Farbwerte rot, grün und blau zerlegt werden.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2} \tag{1c}$$

Watershed (Wasserscheidentransformation)

Die Wasserscheidentransformation ist ein Verfahren, bei dem ein Graustufenbild segmentiert wird. Üblicherweise ist ein Vorverarbeitungsschritt nötig, um Farbwerte in Grautöne umzuwandeln, wobei meistens ein Gradientenbild verwendet wird, welches die Variation der Farbwerte in der Umgebung eines Punktes repräsentiert.
Für den Watershed-Algorithmus wird das Graustufenbild als topografische Fläche interpretiert

bei dem die Pixelwerte die jeweilige Höhe widerspiegeln.
Im eigentlichen Algorithmus werden die Becken der Fläche von vorher definierten Punkten aus „geflutet“. Dabei bilden sich kleine Seen, die mit steigendem Wasserstand irgendwann den Zeitpunkt erreichen, an dem sie sich mit einem bislang getrennten See verbinden. An solchen Stellen werden die sogenannten Wasserscheiden errichtet, welche nach vollständiger Flutung die endgültige Segmentierung widerspiegeln.

IV.3.3 Kantenorientierte Segmentierungsverfahren

Kantenorientententierte Segmentierungsverfahren basieren ebenfalls auf der Annahme von homogenen Eigenschaften von Regionen. Dabei gehen sie davon aus, dass sich die Objektkanten sprunghaft verändern. Diese Verfahren durchsuchen die Nachbarschaft des jeweiligen Pixels nach lokalen Intensitätsunterschieden [38].
Ein Nachteil dieser Verfahren ist, dass die gefundenen Kanten meist nicht geschlossen sind und somit keine geschlossene Kontur bilden. Allerdings gibt es die „aktiven Konturen“, welche verbundene Kanten garantieren können.

Snake (Aktive Konturen)

Bei den Modellen der aktiven Konturen wird das Problem behandelt, dass eine offene oder geschlossene Kontur in der Nähe des zu betrachtenden Objektes im Bildbereich existiert. Diese muss möglichst so verformt werden, dass ihre endgültige Gestalt mit den Objektgrenzen übereinstimmt. Die Abhängigkeit der Kontur werden in äußere und innere Bedingungen unterteilt. Die äußeren Bedingungen zeigen den Einfluss vom Bildinhalt, wie Grauwert oder Gradient. Die inneren Bedingungen beschreiben die Form und den Verlauf der Kontur, wie Ableitungen oder Krümmungen.
Die bekannteste Variante, die auch in dieser Arbeit angewendet wurde, ist das Snake-Modell, welches sich durch die Veröffentlichung von Kass et al.[39] entwickelte.
Der Name „Snake“(zu deutsch: Schlange) ist daraus entstanden, dass sich die Kontur versucht in ein lokales Minimum „hineinzuschlängeln“[38].
Eine Snake lässt sich als $v(s) = (x(s), y(s))^T$ mit den Koordinatenfunktionen x und y und dem Parameterbereich $s \in [0, 1]$ darstellen. Die Kontur wird hierbei, wie folgt vom Minimum der Energiefunktion bestimmt [38]:

$$E_{snake}(v) = \int_0^1 E_{int}(v) + E_{ext}(v)\, ds. \tag{2}$$

Die interne Energie, welche die Spannung und Starrheit der Kontur beschreibt, lässt sich durch ihre Parameter α und β bestimmen.

$$E_{int}(v) = \alpha(s)\left|\frac{\partial v}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2 v}{\partial s^2}\right|^2 \tag{3}$$

Eine Vergrößerung des Wertes $\alpha(s)$ hat zur Folge, dass die Spannung der Kurve erhöht wird und somit weniger Schleifen und Wellen entstehen. Währenddessen sorgt eine Vergrößerung von $\beta(s)$ für eine Glättung und weniger Flexibilität zur Anpassung an gekrümmte Objektkanten.

Die externe Energiefunktion E_{ext} wird mittlerweile als einziges Energiepotential zusammengefasst, welches Kanten mit starker Intensität in dem Gradientenbild erkennt. Zusätzlich bestimmt ein Wichtungsfaktor den Einfluss dieser externen Energie.

IV.3.4 Segmentierung mithilfe von einfachen Machine Learning-Ansätzen

Machine Learning (zu deutsch: Maschinelles Lernen) ist ein Teilgebiet der künstlichen Intelligenz, bei dem die unterschiedlichen statistischen Modelle Wissen aus bekannten Informationen generieren und anschließend anwenden können.
In dem Anwendungsfall dieser Arbeit müssen aus Bilddaten Objekte segmentiert werden. Hierfür erhalten die Modelle die Bilddaten und die entsprechenden Segmentierungen als binäre Objektmaske, welche anschließend auf bisher ungesehenen Daten, vorhergesagt werden soll.

Random Forest

Ein weit verbreiteter Machine Learning-Ansatz ist *Random Forest*. Dies ist ein Lernalgorithmus, der aus einer Vielzahl von Decision Trees (zu deutsch: Entscheidungsbaum) besteht. Diese Entscheidungsbäume weisen eine meist binäre Baumstruktur auf, welche aus einer möglichst kleinen Zahl von Entscheidungsfragen aufgebaut ist. Das Ziel ist die Trennung der Daten durch Abarbeiten der Entscheidungsfragen bis ein Blatt erreicht wird. [40]
Bei einer Klassifikation mit Random Forest wird jeder, der leicht unterschiedlichen Bäume ausgewertet und die Klasse mit den meisten Stimmen gewählt[41]. Allerdings zeigt sich, dass Random Forest ebenfalls Anwendung in der Regression findet und beispielsweise auch für die Bildsegmentierung genutzt wurde[24, 33, 41, 42].

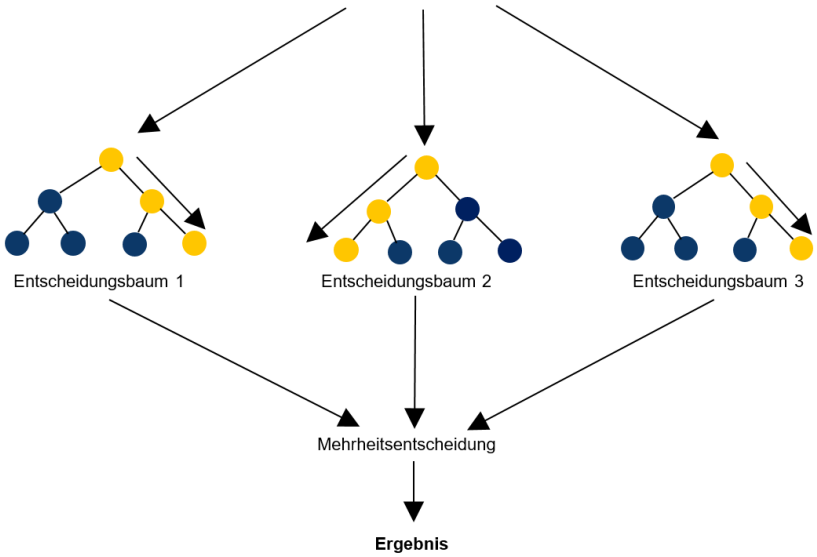

Abbildung 4: Funktionsweise von Random Forest

Support Vector Machine

Support Vector Machine (kurz SVM) ist ein Verfahren aus dem Bereich des maschinellen Lernens. Sie erlaubt das Klassifizieren von Objekten und kommt üblicherweise bei Bild-, Text- oder Handschrifterkennungen zum Einsatz. SVM kann bei linearen und nicht-linearen Klassifizierungen eingesetzt werden, wobei der Objektbereich bei der nicht-linearen Klassifizierung um eine weitere Dimension erweitert wird. Diese sogenannte Hyperebene ermöglicht das Finden einer einfacheren Trennungsfläche zwischen den Klassen. Hierbei wird jedes Objekt als Vektor in einem Vektorraum behandelt und versucht eine möglichst große Distanz zwischen der Hyperebene und den Objektvektoren zu erreichen [3, 43].

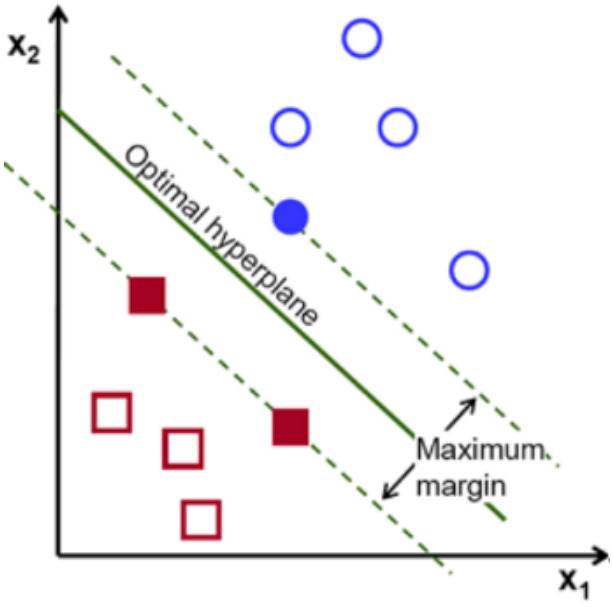

Abbildung 5: Beispiel einer Hyperebene von einer SVM [3]

In Abbildung 5 ist ein einfaches Beispielproblem dargestellt, bei dem eine Trennung zwischen zwei Objektklassen gesucht wird. Hierbei hat die SVM eine Hyperebene, anhand der vorliegenden Daten, so gelernt, dass die einzelnen Objekte einen möglichst großen Abstand zur Ebene haben. Den größten Einfluss auf die Trennebene haben die Objekte, welche den geringsten Abstand besitzen. Sie werden Stützvektoren oder Support Vectors genannt, wovon der Algorithmus seinen Namen hat.

IV.3.5 Convolutional Neural Network

Ein Convolutional Neural Network (kurz CNN) kommt aus dem Bereich der Künstlichen Intelligenz und findet primär Anwendung im Bereich der Bild- und Spracherkennung. Dieses neuronale Netz besteht aus mehreren verbundenen Schichten von Neuronen, welche durch das Anpassen von Gewichten und Schwellwerten die Informationen verarbeiten und weitergeben.
Die Besonderheit dieses Netzes ist, dass es mehrere sogenannte Faltungsschichten beinhaltet. Bei einer Faltung wird schrittweise eine, zur Eingabe vergleichsweise kleine Matrix, auch Kernel genannt, über die Eingabe bewegt und das Produkt aus dem Eingabebereich und der Matrix berechnet. Die anschließende Ergebnismatrix dient als Eingabe der entsprechenden Neuronen,

wodurch benachbarte Neuronen auf überlappende Bereiche reagieren. Dies können sich die genannten Anwendungsbereiche bei der Verarbeitung von ähnlichen Frequenzen oder lokalen Bildpunkten zu nutzen machen. Der Output der Neuronen wird anschließend mithilfe einer Aktivierungsfunktion[4] berechnet.
Weiterhin kann ein CNN sogenannte Pooling-Schichten beinhalten, welche zur Datenreduktion beitragen sollen, indem sie überflüssige Informationen verwerfen. Beispielsweise leitet die verbreitetste Variante *Max*-Pooling aus einem Fenster von Neuronen nur die Aktivitäten des „aktivsten“ Neurons weiter. Diese Technik sorgt für einen geringeren Speicherbedarf und einer erhöhten Berechnungsgeschwindigkeit.
Das Ende des CNNs bildet eine oder mehrere vollständig verknüpfte Schichten von Neuronen. Die Neuronen sind mit allen Elementen der vorgelagerten Schicht verknüpft, sodass für die Ausgabe alle möglichen Informationen mit einfließen. Die Größe der letzten Schicht ist dabei von der Anzahl der zu unterscheidenden Klassen oder Objekte abhängig [44].

ResNet

ResNet steht für *Residual Network* und ist eine der bekanntesten CNN-Architekturen. Bei der Forschung nach neuen CNN-Architekturen wurde festgestellt, dass durch immer tiefer werdende Netze irgendwann keine Verbesserung mehr stattfindet. Das als *Vanishing Grandient* bekannte Problem führt darauf zurück, dass bei der Aktualisierung der Gewichte der gebildete Gradient durch die Vielzahl an Multiplikationen gegen Null geht.

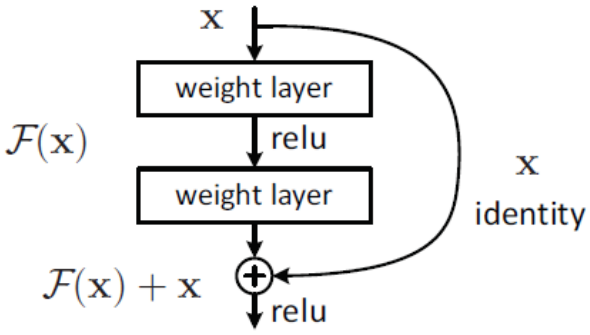

Abbildung 6: Aufbau eines Residual Blocks [4]

Deshalb stellten He et al.[4] 2015 eine neue Technologie für CNN vor und nannten sie *Deep Residual Learning*. Hierbei fließt nicht nur die vorherige Schicht in die Berechnung der Gradienten ein, sondern außerdem wird, wie in Abbildung 6 erkennbar, der Eingang einer Schicht zusätzlich zur Ausgabe addiert, wodurch diese sogenannten *identity shortcut connections* auch in die Berechnung der Gradienten Einfluss haben. Diese Technik ermöglicht deutlich größere Netze, welche nach He et al.[4] auch besser abschneiden als andere CNN.

[4] bei CNN meistens ReLU($f(x) = max(0, x)$)

Mask R-CNN

Das CNN Mask R-CNN ist eine Erweiterung von Faster R-CNN mit dem Zusatz, dass, neben der Vorhersage einer Bounding Box und deren Klassifizierung, noch eine Maske des Objektes vorhersagt und somit speziell für die Objektsegmentierung geeignet ist. Beide Ansätze sind CNNs die in der Bildverarbeitung in ihrem jeweiligen Anwendungszweck weit verbreitet sind [1, 5, 23, 17].

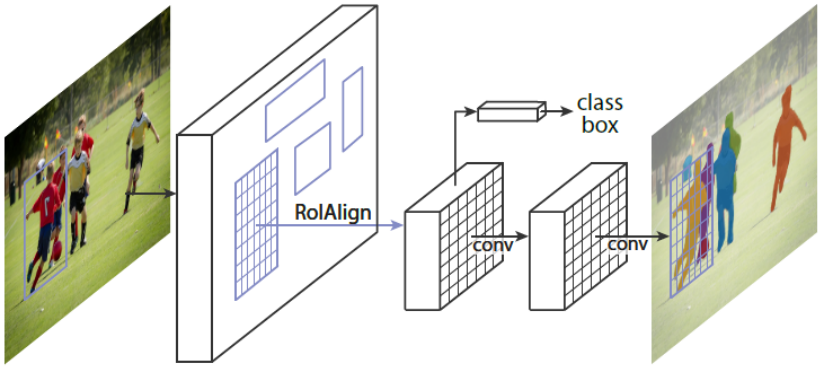

Abbildung 7: Aufbau des Mask R-CNN Frameworks [5]

Mask R-CNN besteht, wie auch Faster R-CNN, aus 2 Schritten, welche in Abbildung 7 sichtbar sind. Im ersten Teil werden im sogenannten Region Proposal Network bis zu 200000 Regionen (RoI) vorhergesagt, in denen sich interessante Objekte befinden können.
Anschließend folgt im zweiten Schritt die Klassifizierung der Regionen und Generierung der Masken. Als grundlegende CNN-Architektur wird für diese Arbeit ResNet[4] benutzt, da es nach He et al.[5] üblicherweise angewendet wird.
In Abbildung 8 ist zu erkennen, wie Faster R-CNN mit ResNet zu Mask R-CNN erweitert wurde. Hierbei wurde ein weiterer Zweig angelegt, welcher mithilfe von zusätzlich Convolutional Schichten die Masken generiert.

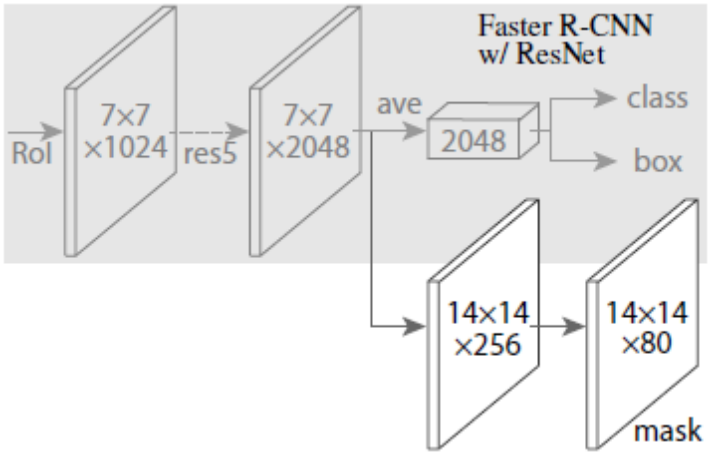

Abbildung 8: Erweiterung von Faster R-CNN zu Mask R-CNN [5]

Yolact

Yolact[5] ist, ähnlich wie Mask R-CNN zu Faster R-CNN, eine Erweiterung des CNN Yolo bei der, neben der Erkennung von klassifizierten Bounding Boxen, zusätzlich Masken für die Objekte vorhergesagt werden.
Nach Bolya et al.[6] ist ein großer Vorteil gegenüber anderen Ansätzen, wie Mask R-CNN, dass es in Echtzeit[6] laufen kann.

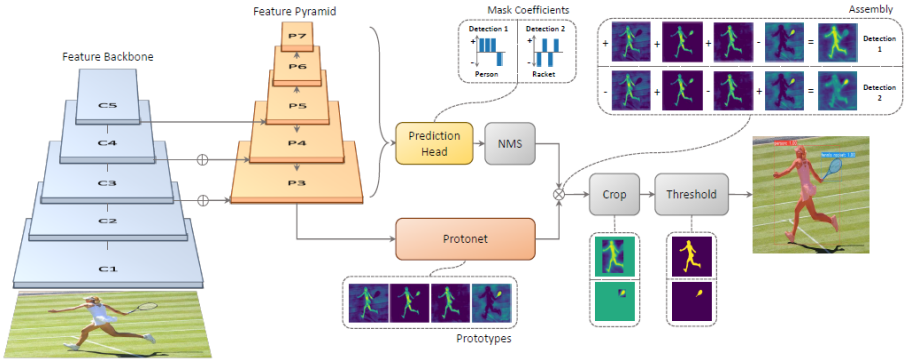

Abbildung 9: Aufbau des Yolact Frameworks [6]

Yolact hat keinen expliziten Lokalisierungsschritt, sondern einen Zweig (hier: *Protonet*) indem prototypische Masken erstellt werden und einen Zweig (hier: *Prediction Head*), welcher jeweils diverse Vektoren für die Klassifizierung vorhersagt. Anschließend werden beide Zweige verknüpft, um die endgültigen, klassifizierten Masken zu generieren.

IV.4 Morphologische Operationen

Morphologische Operationen sind Methoden zur Verarbeitung von binären Bildern, bei denen die Pixel als Mengen interpretiert werden. Für die Verarbeitung werden sogenannte Strukturelemente benutzt, welche strukturierte Mengen sind. Bei Binärbildern können Strukturelemente maximal zweidimensional sein, während sie bei Grauwertbildern maximal dreidimensional sind.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & X & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & X & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Vierer-Nachbarschaft | Achter-Nachbarschaft

Abbildung 10: Beispiele für zweidimensionale Strukturelemente

Die eigentliche Operation findet statt, während das Strukturelement pixelweise über das Gesamtbild verschoben wird. Dabei gibt es die Basisoperationen Erosion und Dilation, welche im folgenden Abschnitt näher erläutert werden.

[5] Yolact steht für **Y**ou **O**nly **L**ook **A**t **C**oefficien**T**s
[6] Nach [6] ab 30 FPS

Erosion

Bei der Erosion wird an jeder Stelle x überprüft ob das Strukturelement S vollständig in die Menge X passt. Wenn dies der Fall ist, wird der Bezugspunkt x zur erodierten Menge hinzugefügt.

$$E_S(X) = \{x | S_x \subseteq X\} \tag{4}$$

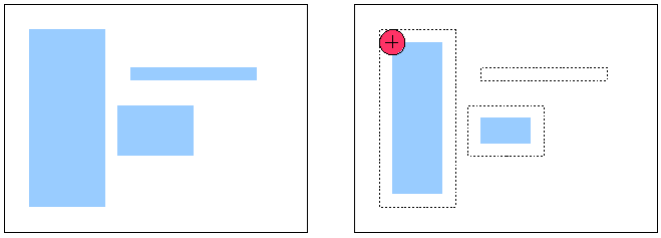

Abbildung 11: Illustrierung der morphologischen Erosion von Binärbildern [7]

Dilatation

Bei der Dilatation wird mithilfe eines Strukturelementes S an jedem Pixel x überprüft, ob S die Menge X berührt.

$$D_S(X) = \{x | S_x \cap X \neq 0\} \tag{5}$$

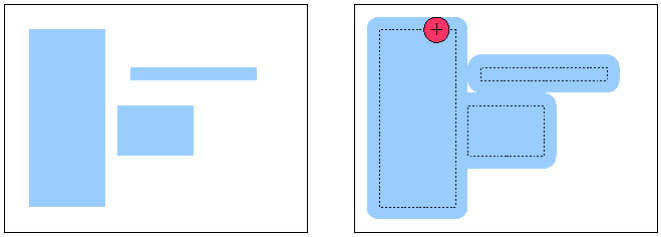

Abbildung 12: Illustrierung der morphologischen Dilatation von Binärbildern [8]

Schliessen

Das morphologische Schließen (*engl.: closing*) ist die Verkettung von der Dilatation D und der Erosion E mit dem selben Strukturelement S. Durch die Ausbreitung der Dilatation können Löcher in der Menge geschlossen werden. Die anschließende Erosion reduziert die Menge, bis auf die vollständig geschlossenen Löcher, wieder zurück.

$$C_S(X) = E_S(D_S(X)) \tag{6}$$

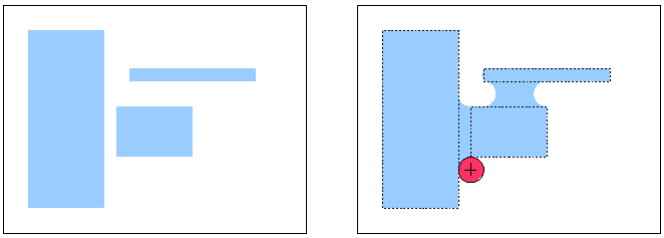

Abbildung 13: Illustrierung des morphologischen Schliessens von Binärbildern [9]

IV.5 Bewertungsmass

Da in dieser Arbeit die Erstellung von Labels eine zentrale Rolle spielt, besteht die Notwendigkeit unterschiedliche Labels miteinander vergleichen zu können. Dafür werden einerseits die Zeiten aufgezeichnet, die benötigt sind, um ein zufriedenstellendes Label aus einem Bild zu erstellen. Weiterhin werden die entstandenen Labels als Teilflächen des Bildes interpretiert und mithilfe des Jaccard-Index verglichen.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{7}$$

Das auch als *Intersection over Union*(IoU) bekannte Bewertungsmaß wird, wie in Gleichung 7 zu sehen, berechnet. Dabei wird die Schnittmenge beider Bereiche durch die Vereinigungsmenge geteilt.

IV.6 Daten

Daten für die Segmentierung ohne Vorwissen

Für das Testen und die Evaluation der Segmentierungsverfahren ohne Vorwissen wurde ein Datensatz mit unterschiedlichen Bildern benötigt. Hierfür wurden Bilder aus dem COCO-Datensatz (2017)[10] mithilfe des Python-Paketes *FiftyOne*[45] herausgefiltert.
Der genutzte Datensatz ist einer der bekanntesten Grundlagen in dem Bereich um Bilderkennung und Segmentierung. Die Entwickler aktualisieren den Datensatz und bieten jährliche Herausforderungen, um beispielsweise unterschiedliche Bilderkennnungs- und Segmentierungsmodelle vergleichen zu können. Der benutzte Datensatz besteht aus circa 330000 Bildern, wovon über 200000 gelabelt sind. *FiftyOne* bietet eine Schnittstelle auf diesem Datensatz mit den insgesamt 1,5 Millionen gelabelten Objekten.
Für die Evaluation sollen die Bilder zufällig anhand von vordefinierten Kategorien aus dem Datensatz gefiltert werden. Für eine breite Variation der Klassen wurden die folgenden Kategorien definiert und anhand der Klassen aus dem COCO-Datensatz gleichmäßig aufgefüllt.

Kategorie	**übersetzte Objektbezeichnungen nach [10]**	**Anzahl an Objekten**
Verkehr	Auto, Zug, Bus, Stoppschild	7
Menschen	Person	7
Essen	Pizza, Banane, Apfel	7
Tiere	Katze, Hund, Vogel	5
Objekte	Skateboard, Tastatur, Baseballschläger, Tennisschläger, Surfboard	6

Tabelle 1: Objektkategorien für die Evaluation

Die Schnittstelle von *FiftyOne* bietet die Möglichkeit, durch Angabe von Klassennamen eine festgelegte Menge an Bildern zurückzubekommen, in denen die Klassen auftreten.
Dadurch ist mithilfe der Aufteilung aus Tabelle 1 ein Datensatz von 20 Bildern entstanden, welcher unter anderem die folgenden Beispiele enthält.

Abbildung 14: Beispielbild Person und Baseballschläger aus [10]

Abbildung 15: Beispielbild Banane und Apfel aus [10]

Daten für die Segmentierung mit Vorwissen

Die Segmentierung mit Vorwissen benötigt eine gleichbleibende Domain, um Vorwissen aus bereits gelabelten Daten zu erlangen. Deshalb wurde zum Testen und Evaluieren dieser Verfahren der DAVIS-Datensatz[11] benutzt. Dieser bietet zusätzlich den Vorteil gegenüber dem COCO-Datensatz, dass hier deutlich genauere Label als Ground Truth vorhanden sind. Somit ist für das Training von möglichen Modellen, manuelles Labeln nicht notwendig.
Der DAVIS-Datensatz besteht aus Bildern und segmentierten Labels aus 50 Videosequenzen in den Auflösungen 480p und 1080p[7].

[7] 480p ist eine Abkürzung für eine Bildauflösung mit einer Höhe von 480 Pixel und meistens einer Breite von 640 Pixel. 1080p steht für 1080 Pixel in der Höhe und 1920 Pixel in der Breite

Abbildung 16: Beispielbild aus [11]

Abbildung 17: Beispielabel aus [11]

Da der DAVIS-Datensatz immer nur kurze Videosequenzen beinhaltet, wird für die die Analyse der Größe ein weiterer Datensatz benötigt. Hierfür wurde ein Erdbeer-Datensatz von der *IAV GmbH* bereitgestellt. Dieser enthält circa 2000 hochauflösende Nahaufnahmen von Erdbeeren, welche bereits segmentierte Labels im COCO JSON-Format enthält. Die Bilder können mehrere Objekte enthalten, welche für diese Arbeit der Einfachheit nur der Klasse „Erdbeere“zugeordnet werden.

Abbildung 18: Beispielbild aus dem Erdbeerdatensatz

V Segmentierung ohne Vorwissen

Für die Segmentierung ohne Vorwissen wurden drei unterschiedliche Segmentierungsverfahren genutzt, welche in diesem Abschnitt näher beschrieben werden.

V.1 Pipeline

Für den Segmentierungsprozess wurde die folgende Pipeline konstruiert. Der Ablaufplan erlaubt einen problemlosen Austausch der Verfahren und einen einfacheren Vergleich der entstandenen Segmentierungen. Nach der Bildauswahl kann die erste Nutzerinteraktion getätigt werden, welche von dem ausgewählten Segmentierungsverfahren abhängig ist. Diese werden in dem jeweiligen Abschnitt des Segmentierungsverfahrens näher erläutert.
Mithilfe der Interaktion wird das ausgewählte Verfahren auf dem Bild angewendet. Alle Ansätze wurden so implementiert, dass der temporäre Stand der Segmentierung durch ein binäres Bild repräsentiert wird. Dabei steht in dem Bild eine *1* für den Objektbereich und eine *0* für den Hintergrund.
Anschließend findet eine Nachverarbeitung statt, um eine einheitliche Kontur zu generieren, welche anschließend mithilfe einer *OpenCV*-Funktion [46] ausgelesen und in einem vordefinierten Format als Liste von Konturpunkten abgespeichert werden kann.

V.2 Nutzerinteraktion

Die vorgestellten Segmentierungsverfahren benötigen neben den Bildinformationen zusätzlich Nutzerinteraktionen zur Segmentierung der Objekte. Hierfür besteht die Möglichkeit in das Bild, welches im Tool dargestellt ist, Punkte und Linien mit der Maus einzuzeichnen, wie es beispielsweise in Abbildung 19 gezeigt wird. Dabei wertet das Tool die Linien als Liste von Punkten aus, wobei ein Punkt als zweidimensionale Koordinaten im Bild interpretiert werden. Die Art der Interaktion hängt von dem jeweiligen Verfahren ab. So benötigen Region Growing und Watershed Markierungen in einzelnen Segmenten des Bildes, während Snake eine ungefähre Kontur des Objektes benötigt.
Zusätzlich wurden „Pinsel“ und „Radierer“ als Korrekturtools implementiert, um den Nutzer weitere Optionen zu geben, die Maske anzupassen.

Abbildung 19: Beispiel einer eingezeichneten Nutzerinteraktion(rot) auf einem Bild aus [10]

V.3 Nachverarbeitung

Bei der Analyse des binären Bildes hat sich ein Nachverarbeitungsschritt als sinnvoll erwiesen. Der Grund hierfür ist, dass das binäre Bild, wie in dem Beispiel aus Abbildung 21, oft Lücken oder Löcher aufweist, die zur ungewollten Spaltung der Kontur führen können. Deshalb wird vor dem Speichern und Anzeigen des Labels das morphologische Schließen auf die binäre Maske ausgeführt.

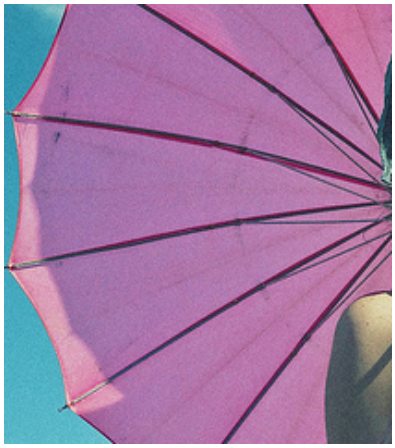

Abbildung 20: Originalausschnitt aus einem Bild aus dem COCO-Datensatz [10]

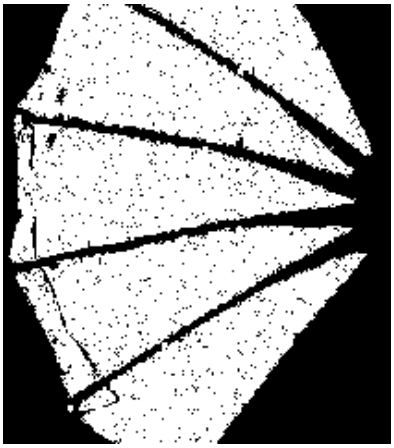

Abbildung 21: Ausschnitt aus dem Ergebnis des Region Growing-Algorithmus

Abbildung 22: Ausschnitt aus dem Ergebnis des morphologischen Schliessens

V.4 Region Growing

Der Region Growing-Algorithmus wurde, wie in Unterunterabschnitt IV.3.2 beschrieben implementiert. Dabei wurden durch den Nutzer gesetzte initiale Punkte als Ausgangspunkte für die Region benutzt. Als mögliche Nutzerinteraktion können hier durch klicken oder das Ziehen von Linien auf dem Objekt Punkte markiert werden, wobei gezeichnete Linien als Liste von Punkten interpretiert werden.
Anschließend wurde die Nachbarschaft der äußeren Pixel der Region überprüft, wofür die Achter-Nachbarschaft (Abbildung 10) diente.
Dabei wird die Differenz der Farbwerte zwischen den äußeren Pixeln und den Nachbarpixeln durch die euklidische Distanz (siehe Gleichung 1c) berechnet. Als weitere Kontrolle werden die Werte zwischen dem Nachbarpixel und dem initialen Punkt verglichen, um weiche Farbübergänge zu vermeiden und somit bei einer zu großen Differenz das Wachstum abzubrechen.
Nutzerinteraktionen sind wiederholbar, sodass die Segmentierung solange angepasst wird, bis der Nutzer zufrieden ist und das Ergebnis zusammen mit einem Labelnamen abspeichern kann. Durch das Abspeichern wird das binäre Segmentierungsbild zurückgesetzt, weshalb die Segmentierung von mehreren Objekten je Bild ermöglicht wird.

V.5 Snake

Für den Snake-Algorithmus, welcher in Unterunterabschnitt IV.3.3 beschrieben wurde, dient eine vorgefertigte Implementierung von *scikit-image* [47] als Grundlage.
Als Nutzerinteraktion muss eine ungefähre Kontur um das Objekt in dem Bild gezeichnet werden. Mithilfe der Kontur und dem Bild entsteht durch den Algorithmus eine neue Kontur, welche durch ausfüllen in eine binäre Objektsegmentierungsmaske umgewandelt werden kann. Diese Maske kann durch weitere Interaktion erweitert werden oder mithilfe der Korrekturtools bearbeitet werden.

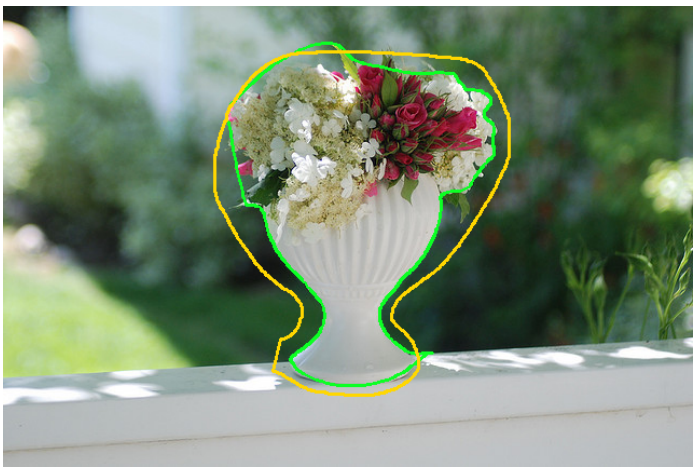

Abbildung 23: Anwendung des Snake-Algorithmus auf ein Beispielbild aus dem COCO-Datensatz [10] (Nutzerinteraktion=gelb und Ergebniskontur des Snake-Algorithmus=grün)

V.6 Watershed

Für die Umsetzung der Wasserscheidentransformation wurde auch hier eine Implementierung von *scikit-image* [47] genutzt.
Wie in Gleichung IV.3.2 beschrieben, werden für dieses Verfahren ein Gradientenbild und initiale Punkte benötigt. Hierfür wird aus der Differenz zwischen der Dilation und der Erosion das morphologische Gradientenbild erzeugt, wie es in Abbildung 25 zu erkennen ist.

Abbildung 24: Beispielbild aus dem COCO-Datensatz [10]

Abbildung 25: Gradientenbild

Für die Generierung der Ausgangspunkte des Verfahrens wurden zwei unterschiedliche Ansätze untersucht. Zuerst wurde eine Menge von Initialpunkten zufällig verteilt, sodass die Wasserscheidentransformation ohne Nutzerinteraktion auf das Bild angewendet werden konnte. In Abbildung 26 ist ein zufriedenstellendes Resultat erkennbar, da viele Objektgrenzen des Busses richtig erkannt wurden. Trotzdem kann die zufällige Verteilung durch einen Mangel an Punkten in einer Region zu fehlender Segmentierung, wie es in der Abbildung auf der linken Seite des Busses erkennbar ist, oder ein Überfluss zu einer Übersegmentierung führen. Somit müsste der Nutzer im Anschluss alle Segmente auswählen, die zum Objekt gehören und zusätzliche weitere Initialpunkte vorgeben oder manuelle Grenzen einzeichnen.

Abbildung 26: Segmentierung der Wasserscheidentransformation mit zufälligen Initialpunkten

Abbildung 27: Segmentierung der Wasserscheidentransformation mit manuell gesetzten Initialpunkten

Bei dem zweiten Ansatz werden die Initialpunkte vom Nutzer, durch das Einzeichnen von Linien und Punkten auf dem Bild, gesetzt. Dies hat den Vorteil, dass der Nutzer gleichzeitig bestimmen kann, ob die Initialpunkte zum Objekt oder zum Hintergrund gehören, wodurch die Auswahl der Segmente wegfällt.
So ist in Abbildung 27 zu erkennen, wie die Objektgrenzen des Busses mit deutlich weniger Teilsegmente durch gezielte Nutzerinteraktionen gefunden wurden. Deshalb wird für den weiteren Teil der Arbeit die Wasserscheidentransformation mit manuell gesetzten Initialpunkten verwendet.

V.7 Vergleichsgrössen

Um eine Aussage über die Ergebnisse der Segmentierungsverfahren treffen zu können, wird eine Vergleichsgröße benötigt. Hierfür besteht die Möglichkeit die Labels mit manuell erstellten Labels zu vergleichen oder Datensätze mit vordefinierten Ground Truth Label zu benutzten. Der verwendete COCO-Datensatz[10] hat den Vorteil, dass für einen Großteil der enthaltenen Bilder sowohl Bounding Boxes als auch segmentierte Labels bereitstehen.
Zuerst wurden manuell erstellte Labels mit diesen Ground Truth Labels verglichen, um die Genauigkeit zu überprüfen. Hierfür wurden für alle 32 Objekte die IoU zwischen den beiden Labels berechnet, woraus anschließend der Durchschnitt von 0,87 und eine Varianz von 0,005 ermittelt werden konnte. Allerdings wurde bei der visuellen Betrachtung der Labels deutlich, dass

die Ground Truth Label sehr einfach gehalten sind und keine gute Vergleichsgröße repräsentieren. In Abbildung 28 wird ein Vergleich zwischen den gegebenen Ground Truth Label und einem manuell erstellten Label dargestellt. Auf dem Bild ist eine starke Übersegmentierung mit wenig Konturpunkten zu erkennen, welche zu einer IoU mit dem manuellen Labels von 0,8 führt.

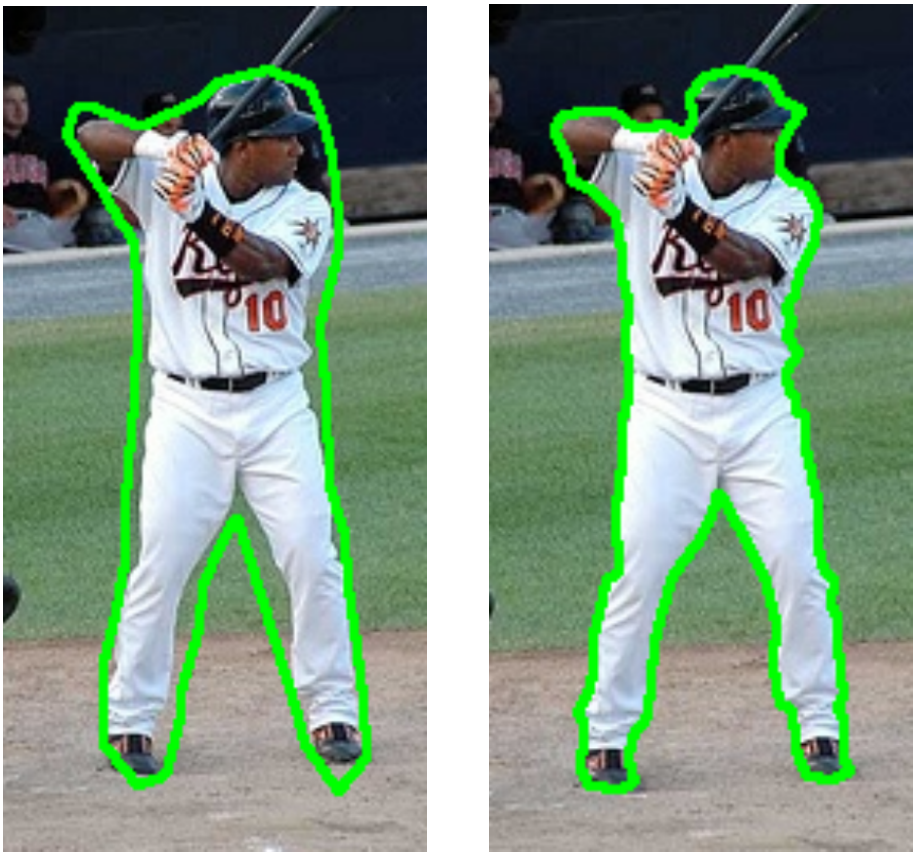

Abbildung 28: Vergleich zwischen Ground Truth Label (links) und manuellem Label (rechts)

Deshalb soll für diese Arbeit das manuelle Labeln als Vergleich zu den Labels dienen, die mithilfe der Segmentierungsverfahren erstellt wurden. Hierbei sollte die interpersonelle Varianz überprüft werden, um personenabhängige Label zu vermeiden.
Dafür wurden die manuell erstellten Label (hier: Person A) mit manuell erstellten Label von zwei weiteren Personen (B und C) verglichen. Die weiteren Testpersonen hatten bereits Erfahrungen mit dem Labeln von Objekten und haben das Tool im Evaluationsmodus zusammen mit dem vorgestellten Datensatz erhalten. Dieser Modus hatte die zusätzliche Funktion, alle erstellten Label als binäre Maske abzuspeichern und alle benötigten Zeiten zu messen und abzuspeichern, um sie im Anschluss vergleichen zu können. Dabei haben die Probanden die Aufgabe, die vorliegenden Bilder zu labeln, zusammen mit einer Übersicht mit den zu labelnden Objekten bekommen.

IoU zwischen 2 Personen		**Person A**	**Person B**	**Person C**
Person A	**Durchschnitt**	-	0,92	0,93
	Varianz	-	0,003	0,003
Person B	**Durchschnitt**	0,92	-	0,93
	Varianz	0,003	-	0,002
Person C	**Durchschnitt**	0,93	0,93	-
	Varianz	0,003	0,002	-

Tabelle 2: Vergleich der Label von unterschiedlichen Personen anhand dem Durchschnitt und der Varianz der IoU

In der Tabelle 2 ist, anhand von hohen und ähnlichen IoU-Werten, zu erkennen, dass es zwischen den Personen eine vergleichbare Objektinterpretation zum Labeln besteht. Die niedrige Varianz und eine manuelle Inspektion der einzelnen Werte lassen zusätzlich auf geringe Unterschiede schließen.

Person	**durchschnittlich benötigte Zeit pro Objekt**
Person A	55,8 s
Person B	52,5 s
Person C	47,0 s

Tabelle 3: Vergleich der benötigten Zeit von unterschiedlichen Personen

Weiterhin wurden in Tabelle 3 die Ergebnisse des zeitlichen Vergleichs dargestellt. Es ist aufgefallen, dass alle Personen ähnlich lange benötigt haben und auch hier keine Besonderheiten erkennbar sind. Zusätzlich ist zu beachten, dass alle Personen bereits Erfahrungen im Labeln hatten, wodurch unerfahrenere Probanden mit hoher Wahrscheinlichkeit mehr Zeit benötigt hätten.
Die Zeiten werden außerdem in Abbildung 29 dargestellt. Dabei wird deutlich, dass die Zeiten, bis auf einen Ausreißer von Person B, ähnlich verteilt sind, wobei Person C im Durchschnitt am schnellsten und Person A am langsamsten war.

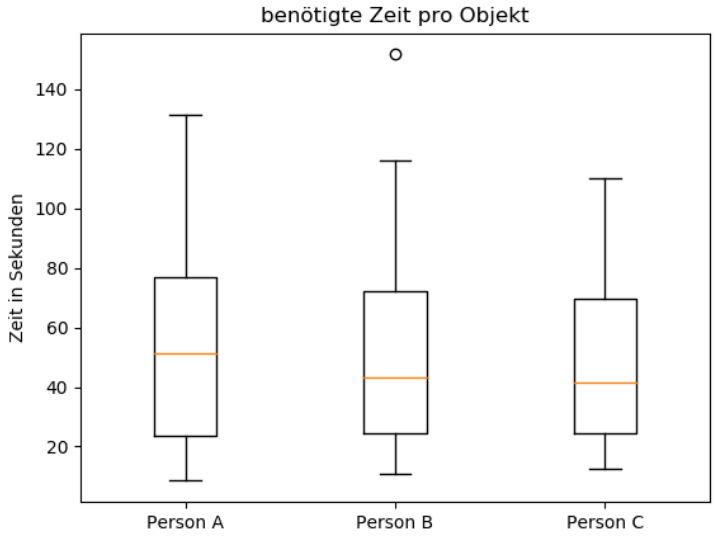

Abbildung 29: Zeitvergleich der manuellen Label als Boxplot

Zusammenfassend kann aus dem Vergleich gesagt werden, dass es bei den drei Personen durch die hohe Übereinstimmung der Label und den kleinen Varianzen nur geringfügige Differenzen gibt.
Für die weitere Evaluation werden die manuellen Label von Person A als Vergleichsgröße genutzt. Für den zeitlichen Vergleich wird der Durchschnitt der drei Personen genutzt, welcher 51,8 Sekunden entspricht.
In den folgenden Abbildungen ist der visuelle Vergleich der Labels an einem Objekt möglich. Zwischen den Labels der unterschiedlichen Personen sind nur geringe Unterschiede erkennbar, während in Abbildung 30 erneut die stark vereinfachten Labels sichtbar sind.

Abbildung 30: Ground Truth Label von COCO[10]

Abbildung 31: Label von Person A

Abbildung 32: Label von Person B

Abbildung 33: Label von Person C

V.8 Evaluation

Für die Evaluation wurden die unterschiedlichen Verfahren unter sich und mit den manuellen Labeln verglichen. Der genutzte Datensatz, welcher in Unterabschnitt IV.6 näher beschrieben ist, wird hierfür nach den vorgegeben Objektklassen aus Tabelle 1 mithilfe der drei vorgestellten Segmentierungsverfahren, sowie komplett manuell gelabelt. Das Endergebnis der jeweiligen Segmentierung wird zusammen mit der benötigten Zeit abgespeichert.
Hierbei wird die benötigte Zeit als Zeitraum zwischen der ersten und letzten Änderung des Labels definiert. Zusätzlich wird für die Segmentierungsverfahren die Berechnungszeit und die Interaktionszeit mit aufgezeichnet. Die Berechnungszeit umfasst die Zeitspanne, die der jeweilige Algorithmus braucht, um die Interaktionen zu verarbeiten. Zur Interaktionszeit gehört die Zeit, die zum Zeichnen benötigt wird, die Korrekturzeit für die Benutzung von Korrekturtools und die sonstige Zeit zur Anpassung, wie das Zoomen, Bewerten, Bewegen der Maus oder das Rückgängig machen.

Verfahren	**Berechnungs-zeit**[8]	**Interaktionszeit**			**benötigte Zeit**
		Zeichnen	Korrekturen	Anpassungen	
Manuell	-	51,8 s			51,8 s
Region Growing	2,3 s	10,6 s	1,2 s	21,8 s	37,1 s
Snake	18,2 s	7,1 s	3,5 s	3,3 s	32,1 s
Watershed	1,8 s	6,2 s	0 s	8,6 s	16,6 s

Tabelle 4: Vergleich der unterschiedlichen Verfahren anhand der durchschnittlichen Zeiten pro Objekt

Verfahren	**Anzahl an Interaktionen**	**Anzahl an Korrekturen**
manuell	60,7	0
Region Growing	20,4	1,2
Snake	1,2	2,2
Watershed	9,3	0

Tabelle 5: Vergleich der durchschnittlichen Anzahl an Interaktionen und Korrekturen pro Objekt

In Tabelle 4 wird der zeitliche Vergleich der Verfahren dargestellt. Für das manuelle Labeln konnte aufgrund der Interaktionsart nur die Gesamtzeit gemessen werden, da hier einzelne Konturpunkte gesetzt werden und keine messbaren Berechnungen stattfinden.
In der Tabelle werden die durchschnittlichen Zeiten pro Objekt dargestellt, wobei in der letzten Spalte sichtbar ist, dass alle Verfahren im Durchschnitt weniger Zeit benötigen und somit alle einen Zeitgewinn erreichten. Bei der Nutzung der Wasserscheidentransformation konnte mit einer durchschnittlichen Zeit von 16,6 Sekunden am schnellsten gelabelt werden. Dies entspricht einem durchschnittlichen Zeitgewinn von 68% im Vergleich zu dem manuellen Labeln.

[8] Berechnungszeit auf einem HP EliteBook 850 G4 (i5-7300U, 16 GB Arbeitsspeicher)

In der Evaluation wird außerdem deutlich, dass der Snake-Algorithmus die höchste Berechnungszeit hat. Dies lässt darauf schließen, dass bei Snake das größte Potential besteht, die Berechnungszeit und dadurch auch die Gesamtzeit durch bessere Hardware zu minimieren. Bei der Betrachtung der Berechnungszeit von Region Growing fällt auf, dass viele Werte unter einer Sekunde liegen, während Ausreißer, wie 11,6 oder 10,7 Sekunden existieren. Diese sind darauf zurückzuführen, dass bei sehr großen Flächen und weichen Farbübergängen die Region ungewollt zu groß werden kann und dabei sehr viele Berechnungen stattfinden.
Weiterhin ist auffällig, dass bei der Wasserscheidentransformation in Tabelle 4 und Tabelle 5 keine Korrekturen aufgezeichnet wurden. Bei der Implementierung der Wasserscheidentransformation hat sich die Möglichkeit geboten, mit der rechten Maustaste Segmente des Hintergrundes zu markieren und somit die Maske aktiv zu verkleinern, wodurch keine Korrekturwerkzeuge notwendig sind. Währenddessen waren die Korrekturtools bei Region Growing und Snake hilfreich, um ungewollte Konturen auszubessern. Bei dem manuellen Labeln wurden die Korrekturtools auch nicht benötigt, da hier die Konturpunkte gezielt gesetzt werden konnten.
In Tabelle 5 ist weiterhin erkennbar, dass der Snake-Algorithmus, trotz der langen Berechnungszeit, die wenigsten Interaktionen benötigt hat. Aber auch Region Growing und Wasserscheide haben, im Gegensatz zu den manuellen Labels, deutlich weniger Interaktionen benötigt. Dies lässt sich darauf zurückführen, dass die Verfahren einen Teil der Arbeit abnehmen und durch das Zeichnen von Linien eine größere Fläche einbezogen werden kann, als es mit einzeln gesetzten Punkten der Fall ist.
Zusammenfassend lässt sich aus der zeitlichen Evaluation schließen, dass aufgrund des Zeitgewinns aller Verfahren eine Nutzung sinnvoll wäre, wenn die erstellten Labels eine zufriedenstellende Genauigkeit erreichen würden, welche im folgenden Abschnitt ausgewertet werden. Bei einer ähnlichen Genauigkeit, sollte allerdings die Wasserscheidentransformation bevorzugt werden, da hier die meiste Zeit gespart wurde.

IoU zwischen 2 Labelvorgängen		Person A	Person B	Person C	Region Growing	Snake	Wasserscheide
Person A	**Durchschnitt**	-	0,92	0,93	0,88	0,87	0,90
	Varianz	-	0,003	0,003	0,007	0,007	0,004
Person B	**Durchschnitt**	0,92	-	0,93	0,84	0,86	0,88
	Varianz	0,003	-	0,002	0,009	0,008	0,005
Person C	**Durchschnitt**	0,93	0,93	-	0,85	0,86	0,89
	Varianz	0,003	0,002	-	0,008	0,008	0,005
Region Growing	**Durchschnitt**	0,88	0,84	0,85	-	0,84	0,88
	Varianz	0,007	0,009	0,008	-	0,010	0,007
Snake	**Durchschnitt**	0,87	0,86	0,86	0,84	-	0,87
	Varianz	0,007	0,008	0,008	0,010	-	0,007
Watershed	**Durchschnitt**	0,90	0,88	0,89	0,88	0,87	-
	Varianz	0,004	0,005	0,005	0,007	0,007	-

Tabelle 6: Vergleich der Label von unterschiedlichen Labelvorgängen anhand des Durchschnittes und der Varianz der IoU

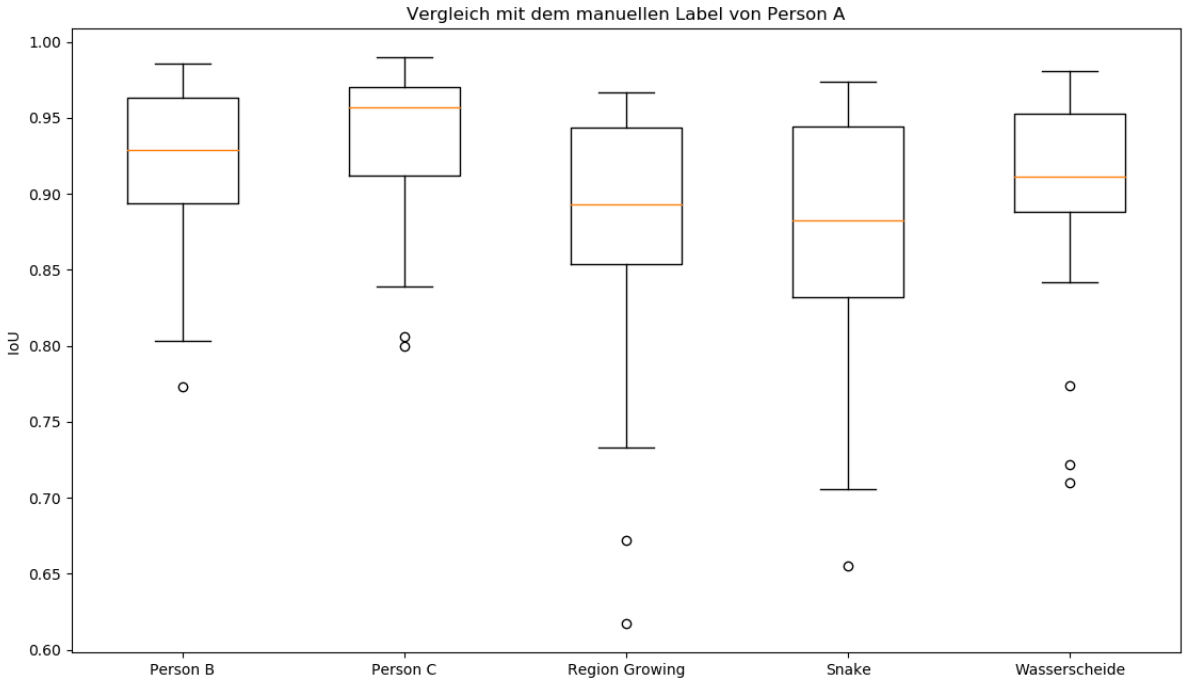

Abbildung 34: Vergleich der Label mit den manuellen Labels von Person A

In Tabelle 6 ist das Gesamtergebnis der Evaluation zu sehen, welches in Abbildung 34 grafisch dargestellt wird. Dort wird sichtbar, dass die Verfahren mit einer durchschnittlichen IoU von 0,84 bis 0,90 hohe Übereinstimmungen mit den manuellen Labels haben. Die höchste Übereinstimmung erreichte dabei die Wasserscheidentransformation mit einer durchschnittlichen IoU von 0,88; 0,89 und 0,90. Bei der visuellen Betrachtung der Boxplot-Diagramme zeigte sich außerdem, dass, bis auf 3 Ausreißer, hohe Übereinstimmungen mit den manuellen Labels von Person A vorhanden sind. Allerdings kam das Verfahren von der Übereinstimmung her im Durchschnitt nicht an die manuellen Labels heran, wodurch eine Nutzung nicht direkt empfohlen werden kann.
Anhand der höheren Varianz und der Ausreißer aus dem Diagramm wird deutlich, dass die Verfahren bei bestimmten Objekten schlechter abgeschnitten haben. Deshalb werden im folgenden Teil der Evaluation diese Schwachstellen untersucht, um zu überprüfen, ob diese Ausreißer bestimmte Merkmale aufweisen oder unabhängig auftreten.

Abbildung 35: Vergleich der entstandenen Label auf einem Bild

Verfahren	**IoU mit manuellem Label**	**benötigte Zeit**
manuell (Person A)	-	8,5 s
Region Growing	0,67	18,9 s
Snake	0,71	24,9 s
Watershed	0,72	7,5 s

Tabelle 7: Evaluation der Segmentierungsverfahren aus Abbildung 35

In Abbildung 35 und Tabelle 7 werden die Ergebnisse der Evaluation von einem Objekt dargestellt, welches niedrige IoU-Werte aufweist. Dabei zeigt sich, dass alle drei Verfahren Probleme mit der Erstellung der Maske des Baseballschlägers hatten. Bei einer genaueren Betrachtung des Objektes fällt auf, dass, aufgrund von schwachen Farbunterschieden, keine klaren Kanten zum Hintergrund existieren. Zusätzlich handelt es sich um ein sehr kleines Objekt, welches aus wenig Bildpunkten besteht. Somit können wenige Punkte einen großen Unterschied auf die IoU bewirken. Zusätzlich zeigt sich in den Zeiten der Tabelle 7, dass bei Region Growing und Snake es zu deutlichen Zeitverlusten gekommen ist, während das Labeln mithilfe der Wasserscheidentransformation mit einer Sekunde zu einem geringen Zeitgewinn führte. Da der Watershed-Algorithmus trotzdem insgesamt eine hohe Übereinstimmung erreichte, kann eine Nutzung je nach Anwendungsfall sinnvoll sein.

VI Segmentierung mit Vorwissen

Bei der Segmentierung mit Vorwissen handelt es sich um einen Segmentierungsprozess, bei dem ein Datensatz einer Klasse existiert und bereits teilweise gelabelt ist. Hierbei geht es um die Evaluation, ob Modelle mit diesen bereits gelabelten Daten trainiert werden können, um anschließend auf ungesehenen Daten einen schnelleren oder genaueren Labelprozess zu ermöglichen.

VI.1 Pipeline

Dieser Ansatz wird aufgrund von fehlender Zeit und einer fehlenden Architektur nicht direkt in das Tool implementiert. Für einige, der im folgenden Absatz behandelten, Ansätze wird eine Grafikkarte empfohlen [48], um es in annehmbarer Zeit zu nutzen. Allerdings darf für die Nutzung des Tools, der Besitz einer Grafikkarte keine Voraussetzung sein. Deshalb wäre eine mögliche Umsetzung eine Client-Server-Architektur, bei dem die Segmentierungsverfahren extern auf einem Grafikkartenserver laufen würden.
Bei einer möglichen Implementierung, müsste zuerst ein Teil des Datensatzes manuell gelabelt werden, um eine Grundlage für das Training der Modelle zu gewährleisten. Dies kann beispielsweise mithilfe der Segmentierungsverfahren ohne Vorwissen gemacht werden, wie es in der folgenden Pipeline skizziert ist.

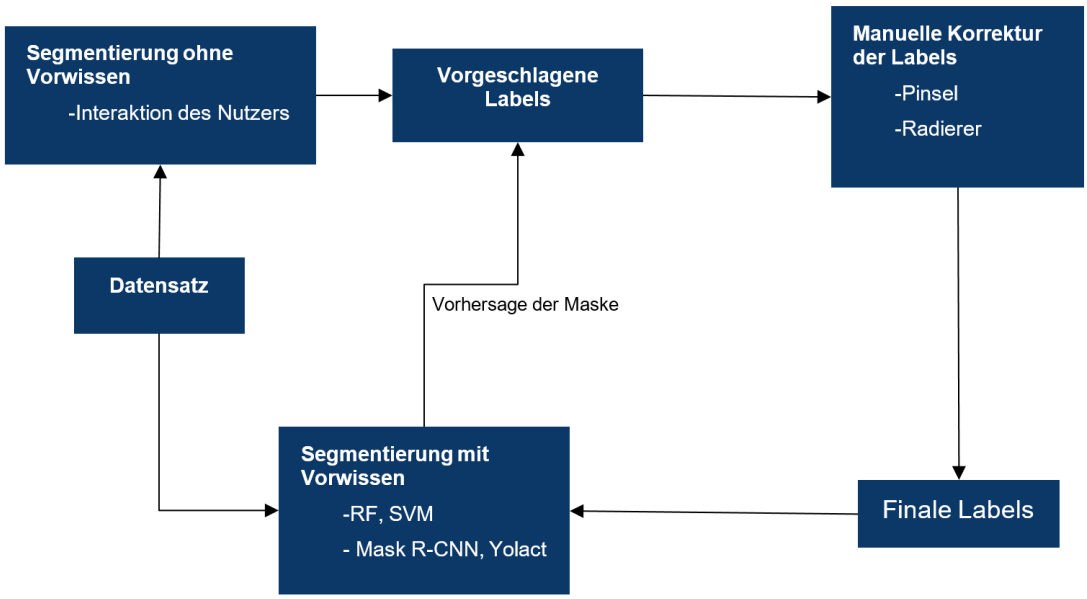

Abbildung 36: Pipeline des Segmentierungsprozesses dieser Arbeit

Anschließend kann das entsprechende Modell auf die Vorhersage von Masken trainiert werden. Zum Schluss kann das trainierte Modell auf weiteren Bildern angewendet werden, um neue Masken vorherzusagen. Diese können anschließend im Tool angezeigt werden und mithilfe der Korrekturtools weiter bearbeitet werden.

Die, im kommenden Absatz beschriebenen, Verfahren werden wie folgt evaluiert. Zuerst werden alle Ansätze anhand von einem Datensatz aus DAVIS[11] miteinander verglichen. Hierbei handelt es sich um eine Videosequenz mit der Bezeichnung „car-turn“, bei der ein einzelnes Auto auf einer Straße fährt.

Abbildung 37: Beispielbild aus dem Evaluationsdatensatz [11]

Von dem insgesamt 70 Bildern wurden zehn Bilder für diese eigentliche Evaluation reserviert. Die restlichen Bilder wurden für das Training der einzelnen Modelle benutzt. Wie in Abbildung 17 gezeigt wurde, enthalten die Datensätze bereits sehr genaue Masken, wodurch kein manueller, zeitintensiver Labeling-Prozess notwendig ist.
Im Anschluss konnten die Ergebnisse und die Ground-Truth-Masken mithilfe der IoU verglichen werden.

VI.2 Random Forest

Der Random Forest-Klassifikator wurde mithilfe des Python-Paket *scikit-image*[47] umgesetzt. Die Anzahl der Entscheidungsbäume von 250 und die maximale Tiefe der Bäume von 12 wurde aus [49] übernommen.
Für die Untersuchung des Ansatzes wurden unterschiedliche Eingaben getestet, welche in Tabellenform mithilfe des Paketes *pandas*[50] dem Model übergeben wurde. Zuerst wurde das einfache Graustufenbild und dann im Anschluss das originale Farbbild im RGB-Farbraum getestet. Bei der visuellen Inspektion von Abbildung 39 zeigt sich, dass zwar die Konturen des Fahrzeugs erkennbar sind, allerdings ist das Bild stark übersegmentiert, da ein Großteil des Hintergrundes als Objekt segmentiert wurde.
Im nächsten Schritt wurde das Modell mit den RGB-Farbwerten trainiert, wodurch eine klare Verbesserung erkennbar wurde. In Abbildung 40 wird dargestellt, dass der Hintergrund deutlich weniger segmentiert wurde, allerdings weißt auch das Fahrzeuge eine schlechtere Kontur auf.
Im nächsten Schritt wurden für den Eingang des Modells neben den Farbwerten noch weitere berechnete Features hinzugefügt. Das Ziel hierfür ist, dass durch zusätzliche Informationen die Objekte besser segmentiert wurden. Dabei handelt es sich um eine Vielzahl von unterschiedlichen Features[9] zur Glättung und Kantenerkennung, welche aus dem Farbbild berechnet wurden.

[9] Übernahme von Features aus [24, 49, 51]

Beispiele hierfür sind der Gauß-Filter, welcher ein Weichzeichner mit unterschiedlichen Parametern ist oder der Sobel-Filter, bei dem mithilfe eines horizontalen und vertikalen Gradientenbildes Informationen zur Kantenerkennung bereitgestellt werden.
Bei der Betrachtung der Ergebnisse, wie zum Beispiel in Abbildung 41, wird deutlich, dass im Hintergrund nur noch einige Punkte zur Objektmaske zugeordnet wurden. Allerdings ist das eigentliche Objekt deutlich untersegmentiert.

Abbildung 38: Originalbild aus DAVIS [11]

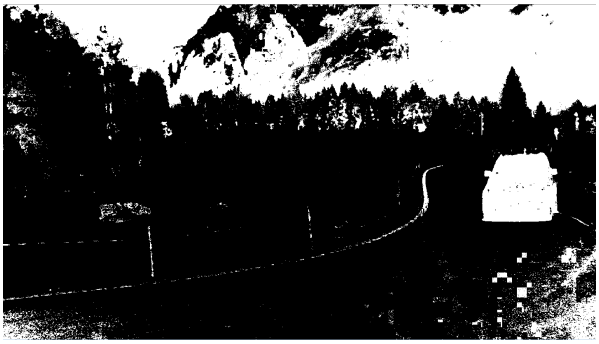

Abbildung 39: Beispielmaske des Random Forest-Algorithmus mit dem Graustufenbild als Eingang

Abbildung 40: Beispielmaske des Random Forest-Algorithmus mit dem RGB-Bild als Eingang

Abbildung 41: Beispielmaske des Random Forest-Algorithmus mit weiteren Features

VI.3 Support Vector Machine

Die Support Vector Machine (kurz: SVM) wurde, wie in Unterunterabschnitt IV.3.4 beschrieben, mithilfe einer vorgefertigten Implementierung von *scikit-image*[47] umgesetzt. Hierfür wurden die Standardparameter des Python-Paketes übernommen. Die SVM wurde, wie auch Random Forest, sowohl mit dem Graustufenbild, als auch mit dem RGB-Farbbild trainiert wurde. Dabei wurde deutlich, dass die vorhergesagten Masken ebenfalls übersegmentiert sind.

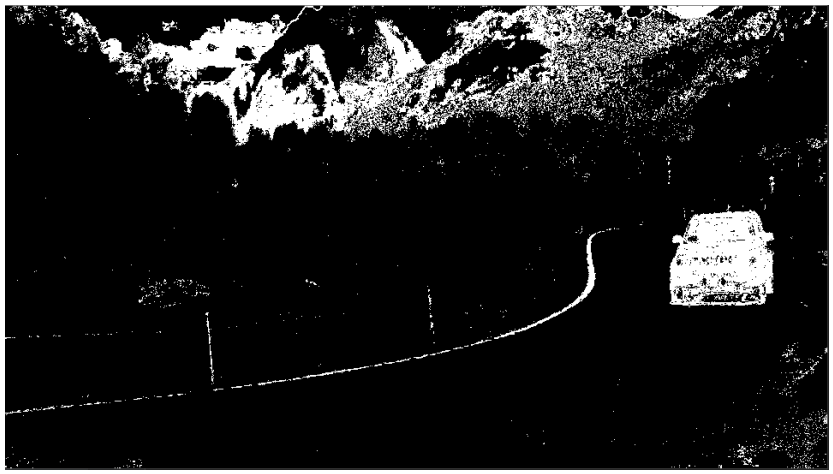

Abbildung 42: Beispielmaske der SVM mit dem RGB-Farbbild als Eingang

Weiterhin wurde auch bei der Support Vector Machine die Vorhersage der Maske mithilfe von weiteren Features getestet. Allerdings sorgte hierbei eine erheblich längere Trainingszeit und die Vorhersage von leeren Masken dafür, dass dieser Ansatz nicht für die Anwendung genutzt werden kann und somit nicht weiter berücksichtigt wird.

VI.4 Mask R-CNN

Die Implementierung von Mask R-CNN wurde mithilfe von [5] umgesetzt. Hierfür war es notwendig den Trainingsdatensatz von 70 Bildern in 50 Trainings- und 20 Validierungsbilder aufzuteilen. Als CNN-Architektur wurde dabei das vorgeschlagene ResNet-101 benutzt und dazu auf dem COCO-Datensatz[10] vortrainierte Gewichte geladen. Dabei wurde auf 80 unterschiedlichen Klassen vortrainiert, die unter anderem „car“ enthalten.
In der folgenden Abbildung ist ein Ergebnis aus der Evaluation von Mask R-CNN sichtbar. Dabei zeigt sich, dass das Modell eine gute Maske vom Fahrzeug vorhersagen konnte.

Abbildung 43: Beispielmaske von Mask R-CNN

VI.5 Yolact

Für die Umsetzung von Yolact, welches in Unterunterabschnitt IV.3.5 beschrieben wurde, anhand von [6] war zuerst eine Konvertierung der Masken aus dem DAVIS-Datensatz zu Label im COCO-Format notwendig. Zusätzlich mussten auch hier die Trainingsdaten nochmal in Trainingsdaten und Validierungsdaten(Verhältnis 5:2) aufgeteilt werden. Das neuronale Netz wurde anschließend mit der vorgeschlagenen Grundarchitektur ResNet-50 konfiguriert, welches auf dem ImageNet-Datensatz[10] vortrainiert ist. Es ist ein wichtiger Punkt, dass das Netz bereits auf Klassen, wie „car“ vortrainiert ist und somit davon auszugehen ist, dass ein Training auf diese Klassen schneller und genauer sein wird, als auf bisher unbekannten Klassen.
Wie auch bei Mask R-CNN wurde das Training und die Evaluation auf einem Computer mit einer Grafikkarte ausgeführt, um die Laufzeit zu minimieren. Ein Beispielergebnis der Anwendung des trainierten Modells ist in Abbildung 44 zu sehen. Hier wird deutlich, dass das Fahrzeug mit Yolact gut erkannt werden konnte, aber die Kontur etwas grob ist.

Abbildung 44: Beispielmaske von Yolact

VI.6 Vergleich der Segmentierungsverfahren

Für den Vergleich wurden die vier Segmentierungsverfahren auf den Testdatensatz von zehn Bildern angewendet. Anschließend konnten die vorhergesagten Masken mit den Ground Truth-Masken des Datensatzen verglichen werden. Hierfür dient, wie auch bei der Segmentierung ohne Vorwissen die IoU. Weiterhin ist zu beachten, dass ein zeitlicher Vergleich nicht möglich ist, da die Modelle auf unterschiedlichen Systemen angewendet wurden. Außerdem kann die Trainingszeit vernachlässigt werden, da durch das Auslagern auf ein System mit besserer Hardware die Zeit deutlich sinken sollten. Dazu besteht die Möglichkeit, dass die Person während des Trainingsschrittes anderen Tätigkeiten nachgehen kann, wie beispielsweise dem weiteren Labeln mit Segmentierungsverfahren ohne Vorwissen.

[10] Imagenet ist ein großer Bilddatensatz mit teilweise gelabelten Daten und über 20000 Labelkategorien[52]

Verfahren	**durchschnittlich IoU**	**Varianz der IoU**
Random Forest	0,197	0,0066
SVM	0,110	0,0102
Mask R-CNN	0,901	0,0006
Yolact	0,908	0,0015

Tabelle 8: Vergleich der IoU

In der Tabelle werden die Ergebnisse der vier Segmentierungsverfahren dargestellt. Dabei handelt es sich bei Random Forest um die Version, welche neben dem RGB-Farbbild noch weitere Features als Eingang erhält. Die evaluierte SVM bekommt lediglich das RGB-Farbbild, da Modelle mit weiteren Features dazu führten leere Masken vorherzusagen und somit lediglich eine IoU von 0 erreichten.
In der Tabelle spiegeln sich die Ergebnisse aus den jeweiligen Beispielmasken wider. So erreichten Random Forest und SVM sehr niedrige Durchschnitte und sind somit für eine mögliche Nutzung im Tool nicht zu gebrauchen. Währenddessen erreichten beide CNN gute Ergebnisse und konnten im jedem Testbild das Objekt erkennen. Somit besteht die Möglichkeit, einen der Ansätze im Labeling-Tool zu nutzen, um neue Label bei größeren Datensätzen vorzuschlagen.
Bei weiteren Tests der CNN wurde deutlich, dass mit Mask R-CNN Probleme bei anderen Datensätzen entstanden, bei den die Klasse nicht mit vortrainert wurde. So führten beispielsweise die Datensätze „goldfish“ und „soapbox“[11], welche nicht im COCO-Datensatz enthalten sind, dazu, dass das Modell keine Objekte auf dem Testdatensatz vorhersagen konnte. Währenddessen erreichte Yolact eine durchschnittliche IoU von 0,83 (Varianz: 0,00024) auf dem „soapbox“-Datensatz und lieferte somit gute Ergebnisse für dieses komplexe Objekt.

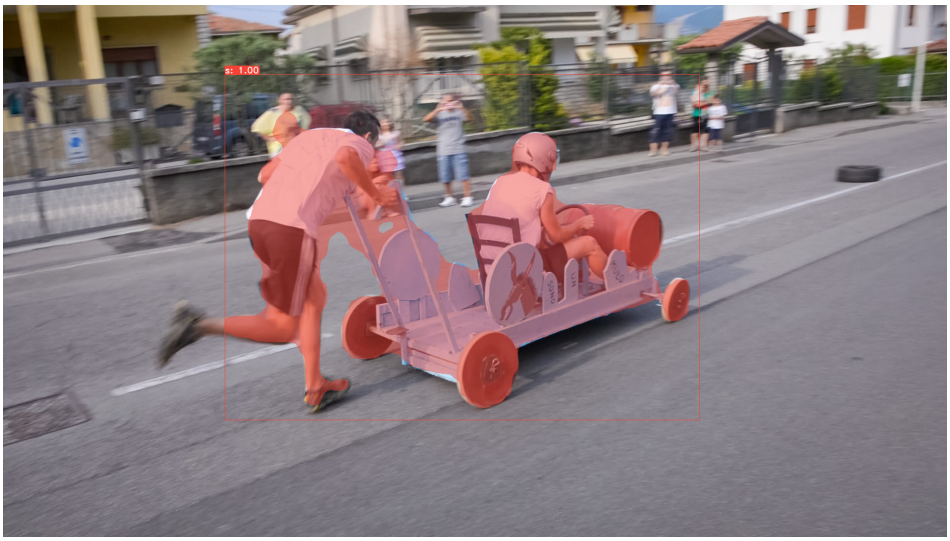

Abbildung 45: Ergebnis der Anwendung von Yolact auf ein Testdatensatzbild aus [11]

[11] Bei dem Datensatz ging es darum eine Seifenkiste inklusive von zwei Personen zu erkennen.

Dies lässt darauf schließen, dass bei Klassen, welche nicht mit vortrainiert wurden, eine Anwendung nur mit einem größeren Datensatz sinnvoll ist. Außerdem zeigt sich hier durch die Vielzahl unterschiedlicher Klassen, dass ein auf ImageNet[52] vortrainiertes Modell einen deutlich größeren Anwendungsbereich hat.

VI.7 Vergleich der Datensatzgrösse für Yolact

Zwar erreichten die CNNs gute Ergebnisse, allerdings muss für eine mögliche Nutzung zusätzlich geklärt werden, ab welcher Datensatzgröße eine Anwendung sinnvoll ist. Hierfür wurde der Erdbeer-Datensatz der *IAV GmbH*, aufgrund der Vielzahl an bereits segmentierten Bilder genutzt. Dabei wird ein Yolact Modell mit unterschiedlicher Anzahl an Trainingsdaten trainiert und anschließend auf einen Testdatensatz von 10 Bildern mit insgesamt 25 gelabelten Erdbeeren angewendet. Dabei werden falsche Labels mit einer IoU von 0 gewertet. Die Anzahl der Validierungsbilder wurde auf 20% der Trainingsdaten festgelegt und die Trainingsdauer wurde auf 30 Epochen festgelegt.

Anzahl Trainingsbilder	**Anzahl Validierungsbilder**	**durchschnittlich IoU**	**Varianz der IoU**
1000	200	0,89	0,05
500	100	0,72	0,18
100	20	0,67	0,11
50	10	0,66	0,12
25	5	0,37	0,17

Tabelle 9: Vergleich von Yolact mit unterschiedlichen Datensatzgrössen

In der Tabelle 9 sind die Ergebnisse der Evaluation dargestellt. Dabei zeigt sich, dass größere Traningsdatensätze deutlich bessere Ergebnisse erzielen konnten. Das beste Ergebnis erreichte somit das Modell mit 1200 Trainingsbildern (davon 200 Validierungsbilder), welches 24 von 25 Erdbeeren erkannt hat. Währenddessen wurden mit sinkender Datensatzgröße immer weniger Erdbeeren richtig vorhergesagt, sodass das kleinste Modell zwar 43 Erdbeeren angezeigt hat, allerdings stimmten davon nur 15 mit den vorgegebenen Labels überein.
Daraus lässt sich schließen, dass größere Datensätze genauere Vorhersagen ermöglichen. Dies ist zusätzlich in den folgenden Abbildungen erkennbar, da die beiden kleinsten Modelle zu viele und ungenaue Masken vorhersagen. Währenddessen erzeugen die restlichen Yolact-Modelle sehr genaue Masken, wobei dem Modell mit 110 Trainingsbildern zwei Vorhersagen fehlen.

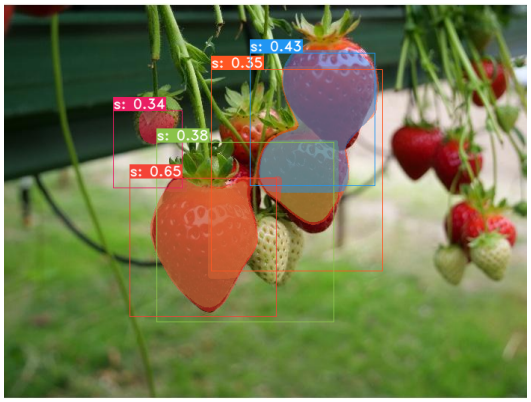

Abbildung 46: Vorhersage des Yolact-Modells mit 30 Trainingsbildern

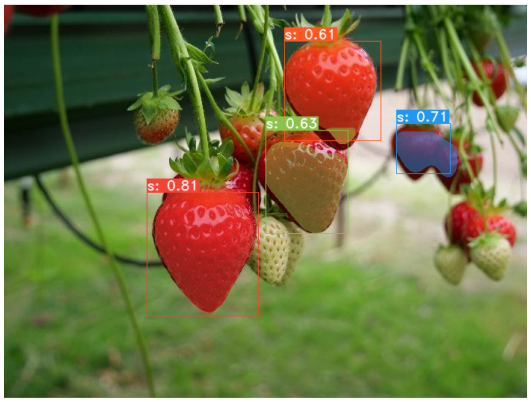

Abbildung 47: Vorhersage des Yolact-Modells mit 60 Trainingsbildern

Abbildung 48: Vorhersage des Yolact-Modells mit 120 Trainingsbildern

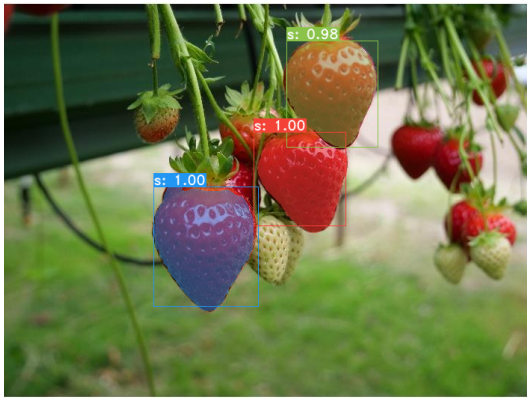

Abbildung 49: Vorhersage des Yolact-Modells mit 600 Trainingsbildern

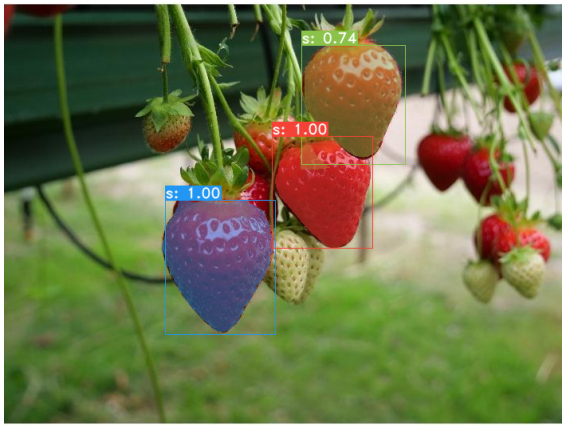

Abbildung 50: Vorhersage des Yolact-Modells mit 1200 Trainingsbildern

Aus der visuellen Inspektion der vorangegangenen Bilder und Tabelle 9 lässt sich schließen, dass für diesen Erdbeerdatensatz 30 Trainingsbilder aufgrund der ungenauen Vorhersagen zu wenig sind und 1200 Trainingsbilder keine signifikante Verbesserung zu dem Modell mit 600 Bildern bringt.

VII Fazit und Ausblick

In dieser Arbeit wurden unterschiedliche Segmentierungsverfahren vorgestellt. Das Ziel dabei war die Erweiterung eines Labeling-Tools der *IAV GmbH* um Zeit beim Labeln, bei gleichbleibender Qualität, zu sparen.
Diese Arbeit ist dabei so aufgebaut, dass zuerst der aktuelle Stand der Technik einen Überblick über die Literatur und bereits vorhandenen Tools gibt. Das zu erweiternde Labeling-Tool der *IAV GmbH* wurde dabei in der Anforderungsanalyse genauer dargestellt. Anschließend werden in Abschnitt IV die benutzten Verfahren und benötigten Grundlagen erklärt. Im Anschluss werden im Hauptteil die Umsetzung und die Evaluation der unterschiedlichen Segmentierungsverfahren beschrieben. Dabei wurden diese so unterteilt, dass Modelle, bei den bereits gelabelte Bilder für ein Training benötigt werden, der Segmentierung mit Vorwissen zugeordnet werden. Die restlichen Verfahren, welche in dieser Arbeit als Segmentierung ohne Vorwissen bezeichnet wurden, wurden mit manuellen Labels verglichen und konnten dabei alle einen Zeitgewinn erzielen. Allerdings wurde bei dem Vergleich deutlich, dass die entstandenen Labels nicht ganz an die Genauigkeit der manuell erstellten Labels kamen. Insgesamt konnte Watershed (Wasserscheidentransformation) die besten Ergebnisse mit einer durchschnittlichen IoU von 0,89 und einem Zeitgewinn von durchschnittlich 68 % der manuellen Labelzeit erzielen und wird somit für die Nutzung empfohlen. Da sich bei der näheren Analyse der Ausreißer auch Schwächen gezeigt haben, muss allerdings bei der Anwendung, insbesondere bei kleinen Objekten, überprüft werden, ob die Labels eine ausreichende Genauigkeit für den jeweiligen Anwendungsfall erzielen.
Bei der Segmentierung mit Vorwissen wurden unterschiedliche statistische Machine Learning Methoden und CNNs zur Segmentierung mit bereits gelabelten Bildern trainiert. Bei der anschließenden Evaluation stellte sich heraus, dass bei einem einfach gehaltenen Beispiel nur die CNNs gute Ergebnisse erzielten. Weiterhin wurde festgestellt, dass bei kleinen Datensätzen die vortrainierten Gewichte einen großen Einfluss auf das Ergebnis haben. So sind gute Ergebnisse bei Klassen entstanden, welche bereits vortrainiert wurden, währende bei neuen Klassen keine sinnvolle Segmentierung stattfand. Besonders positiv ist so Yolact aufgefallen, da es mit ImageNet[52] auf einen der größten Bilddatensätze vortrainiert war.
Im Anschluss zeigte sich an einem Beispiel zur Evaluation der Datensatzgröße, dass die Genauigkeit und die Anzahl richtig erkannter Objekte mit zunehmender Datensatzgröße steigt. Der kleinste Datensatz mit 30 Bildern zeigte dabei durch übergreifende Labels und mehreren falschen Vorhersagen, dass der Trainingsdatensatz vermutlich zu klein war. Deshalb lässt sich daraus schließen, dass für eine Benutzung von Segmentierungsverfahren mit Vorwissen ein CNN empfohlen wird, welches bereits auf die gewünschte Klasse vortrainiert ist und mindestens 50-100 bereits gelabelte Bilder als Training benutzen kann.
Da für ein schnelles Training und einer Vorhersage innerhalb weniger Sekunden eine Grafikkarte benötigt wird, müsste für eine zukünftige Umsetzung die Nutzung nur auf Grafikkartensysteme begrenzt werden. Eine weitere Option wäre die Anpassung zu einer Client-Server-Architektur, sodass die CNNs auf einem Grafikkartenserver ausgeführt werden. Hierbei muss allerdings evaluiert werden, ob sich dieser Umbau für ein einfaches Labeling-Tool lohnt oder ob einfache Segmentierungsverfahren ohne Vorwissen ausreichen.

Für eine weitere Verbesserung des Tools in der Zukunft gibt es mehrere Möglichkeiten. So können beispielsweise weitere Segmentierungsverfahren, wie Adaptive Region Growing, untersucht werden. Somit könnte dem Nutzer die Arbeit abgenommen werden, den Grenzwert von Region Growing manuell anpassen zu müssen. Zusätzlich können die Nutzerinteraktionen im Tool verbessert werden, indem der Nutzer für Region Growing beispielsweise Grenzen definieren kann, um eine Übersegmentierung bei weichen Farbverläufen zu verhindern. Außerdem könnte dem Nutzer die Möglichkeit geboten werden, das Label im Anschluss anhand von verschiebbaren Eckpunkten zu korrigieren. Weiterhin ist es aktuell nur möglich den letzten Schritt rückgängig zu machen. Dies kann durch die Speicherung von weiteren binären Masken der letzten Schritte erweitert werden.
Bei der Segmentierung mit Vorwissen können ebenfalls weitere Ansätze ausprobiert werden. So könnten beispielsweise Segmentierungsverfahren, welche aufgrund von Hardware-Beschränkungen nicht einfach zu reproduzieren waren, angepasst und untersucht werden. Ein möglicher Ansatz wurde von Cheng et al.[22] vorgestellt, bei der das CNN zusätzliche Unterstützung durch Nutzerinterkationen erhält. Weiterhin können hier auch noch weitere Möglichkeiten der Nutzerinteraktionen für die Segmentierung mit Vorwissen untersucht werden. Beispielsweise könnte der Nutzer die Lokalisierung des Objektes übernehmen, sodass ein neuronales Netz nur noch die Segmentierung in einem begrenzten Bereich um das Objekt durchführen muss. Zusätzlich könnte auch die Suche nach einfacheren Netzen für kleinere Datensätze erfolgversprechend sein, da hier weniger Gewichte anzupassen wären.

Literatur

[1] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey, 2019.

[2] Bhupendra Pratap Singh. Imaging applications of charge coupled devices (ccds) for cherenkov telescope. 05 2015.

[3] Rishabh Misra. Support vector machines-soft margin formulation and kernel trick, Jun 2020. URL https://towardsdatascience.com/support-vector-machines-soft-margin-formulation-and-kernel-trick-4c9729dc8efe. (zuletzt besucht 09.11.2021).

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn, 2018.

[6] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. pages 9156–9165, 10 2019. doi: 10.1109/ICCV.2019.00925.

[7] Martin Pfeiffer. Illustrierung der morphologischen Erosion von Binärbildern, 2007. URL https://commons.wikimedia.org/wiki/File:MorphologicalErosion.png. (zuletzt besucht 12.09.2021).

[8] Martin Pfeiffer. Morphologische Dilatation eines Binärbildes, 2008. URL https://de.wikipedia.org/wiki/Datei:MorphologicalDilation.png. (zuletzt besucht 12.09.2021).

[9] Martin Pfeiffer. Illustratierung des morphologischen Schließens von Binärbildern, 2008. URL https://de.wikipedia.org/wiki/Datei:MorphologicalClosing.png. (zuletzt besucht 12.09.2021).

[10] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[11] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[12] Ben-Li Wang, Chung-Ta King, and Hung-Kuo Chu. A semi-automatic video labeling tool for autonomous driving based on multi-object detector and tracker. In *2018 Sixth International Symposium on Computing and Networking (CANDAR)*, pages 201–206, 2018. doi: 10.1109/CANDAR.2018.00035.

[13] N. S. Manikandan and K. Ganesan. Deep learning based automatic video annotation tool for self-driving car, 2019.

[14] Jun Tang. A color image segmentation algorithm based on region growing. In *2010 2nd International Conference on Computer Engineering and Technology*, volume 6, pages V6–634–V6–637, 2010. doi: 10.1109/ICCET.2010.5486012.

[15] Hai Gao, Ping Xue, and Weisi Lin. A new marker-based watershed algorithm. volume 2, pages II – 81, 06 2004. ISBN 0-7803-8251-X. doi: 10.1109/ISCAS.2004.1329213.

[16] J.M. Jaesang Park und Keller. Snakes on the watershed. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1201–1205, 2001. doi: 10.1109/34.954609.

[17] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

[18] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi: 10.1109/CVPR.2016.91.

[19] Hai Gao, Wan-Chi Siu, and Chao-Huan Hou. Improved techniques for automatic image segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(12): 1273–1280, 2001. doi: 10.1109/76.974681.

[20] Dengsheng Zhang, Md. Monirul Islam, and Guojun Lu. A review on automatic image annotation techniques. *Pattern Recognition*, 45(1):346–362, 2012. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2011.05.013. URL https://www.sciencedirect.com/science/article/pii/S0031320311002391.

[21] D. Civco M. Song. Road extraction using svm and image segmentation. *Photogrammetric Engineering and Remote Sensing*, 70:1365–1371, 2004.

[22] Bowen Cheng, Omkar Parkhi, and Alexander Kirillov. Pointly-supervised instance segmentation, 2021.

[23] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks, 2018.

[24] Florian Schroff, Antonio Criminisi, and Andrew Zisserman. Object class segmentation using random forests. 01 2008. doi: 10.5244/C.22.54.

[25] Kentaro Wada. labelme: Image Polygonal Annotation with Python. https://github.com/wkentaro/labelme, 2016. (zuletzt besucht 13.11.2021).

[26] Tzutalin. Labelimg. Free Software: MIT License, 2015. URL https://github.com/tzutalin/labelImg. (zuletzt besucht 13.11.2021).

[27] Josh Veitch-Michaelis. jveitchmichaelis/deeplabel: v0.16.1, October 2021. URL https://doi.org/10.5281/zenodo.5543578.

[28] J. Cartucho, R. Ventura, and M. Veloso. Robust object recognition through symbiotic deep learning in mobile robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2336–2341, 2018.

[29] SuperAnnotate AI. Superannotate. URL https://www.superannotate.com.

[30] V7 Labs. V7. URL https://www.v7labs.com/.

[31] A. Chinnu. Brain tumor classification using svm and histogram based image segmentation. 2015.

[32] Vivek Sharma, Frank Dittrich, Sule Yildirim Yayilgan, Ali Imran, and Heinz Wørn. How to tune a random forest for real-time segmentation in safe human-robot collaboration? volume 528, pages 700–704, 08 2015. ISBN 978-3-319-21379-8. doi: 10.1007/978-3-319-21380-4_118.

[33] Byeongkeun Kang and Truong Q. Nguyen. Random forest with learned representations for semantic segmentation. *IEEE Transactions on Image Processing*, 28(7):3542–3555, Jul 2019. ISSN 1941-0042. doi: 10.1109/tip.2019.2905081. URL http://dx.doi.org/10.1109/TIP.2019.2905081.

[34] Wikipedia. Farbraum — Wikipedia, the free encyclopedia. http://de.wikipedia.org/w/index.php?title=Farbraum&oldid=216152632, 2021. (zuletzt besucht 09.11.2021).

[35] 5 types of image annotation and their use cases, Feb 2015. URL https://www.telusinternational.com/articles/an-introduction-to-5-types-of-image-annotation. (zuletzt besucht 12.11.2021).

[36] Types of image annotation - the ultimate guide for your ai project, Feb 2021. URL https://humansintheloop.org/types-of-image-annotation/. (zuletzt besucht 14.11.2021).

[37] Sandeep Pulla, Anshuman Razdan, and Gerald Farin. Improved curvature estimation for watershed segmentation of 3-dimensional meshes. *IEEE Transactions on Visualization and Computer Graphics - TVCG*, 5, 05 2001.

[38] Michael Kux. Ein Ansatz für die interaktive, semiautomatische Erzeugung von Aktiven Konturen innerhalb triangulierter Oberflächen von3D-Objekten. page 6–38, Apr 2005.

[39] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988. doi: 10.1007/bf00133570.

[40] Benjamin Aunkofer. Maschinelles Lernen mit Entscheidungsbaumverfahren, Dec 2017. URL https://data-science-blog.com/blog/2017/02/13/entscheidungsbaumverfahren-artikelserie/. (zuletzt besucht 28.02.2020).

[41] Wikipedia. Random Forest — Wikipedia, the free encyclopedia. http://de.wikipedia.org/w/index.php?title=Random%20Forest&oldid=190458641, 2020. (zuletzt besucht 02.03.2020).

[42] Philipp Kainz, Martin Urschler, Samuel Schulter, Paul Wohlhart, and Vincent Lepetit. You should use regression to detect cells. volume 9351, 10 2015. ISBN 978-3-319-24573-7. doi: 10.1007/978-3-319-24574-4_33.

[43] Stefan Luber. Was ist eine Support Vector Machine?, Nov 2019. URL https://www.bigdata-insider.de/was-ist-eine-support-vector-machine-a-880134/. (zuletzt besucht 03.01.2020).

[44] Stefan Luber. Was ist ein convolutional neural network?, Mar 2019. URL https://www.bigdata-insider.de/was-ist-ein-convolutional-neural-network-a-801246/. (zuletzt besucht 04.11.2021).

[45] Voxel51 // fiftyone. URL https://voxel51.com/fiftyone/. (zuletzt besucht 03.08.2021).

[46] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[47] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[48] Shachi Shah. Do we really need gpu for deep learning? - cpu vs gpu, Dec 2018. URL https://medium.com/@shachishah.ce/do-we-really-need-gpu-for-deep-learning-47042c02efe2. (zuletzt besucht 17.11.2021).

[49] David Griffiths. Machine learning - image segmentation. https://github.com/dgriffiths3/ml_segmentation, 2018.

[50] The pandas development team. pandas-dev/pandas: Pandas, February 2020. URL https://doi.org/10.5281/zenodo.3509134.

[51] Varshil Gandhi. Svm vs. random forest for image segmentation. https://github.com/varshilgandhi/SVM-vs.-Random-Forest-for-image-segmentation, 2021.

[52] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Eidesstattliche Versicherung

Ich versichere eidesstattlich durch eigenhändige Unterschrift, dass ich die Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht und ist in gleicher oder ähnlicher Weise noch nicht als Studienleistung zur Anerkennung oder Bewertung vorgelegt worden. Ich weiß, dass bei Abgabe einer falschen Versicherung die Prüfung als nicht bestanden zu gelten hat.

Rostock

23.11.2021	Lennard Pfeil
(Abgabedatum)	(Vollständige Unterschrift)