HAW HAMBURG

BACHELORTHESIS
Jonas Kempke

Videobasierte Linienbus- und Fahrtzielerkennung mittels Verfahren des Maschinellen Lernens für einen elektronischen Blindenführhund

FAKULTÄT TECHNIK UND INFORMATIK
Department Informations- und Elektrotechnik

Faculty of Computer Science and Engineering
Department of Information and Electrical Engineering

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Jonas Kempke

Videobasierte Linienbus- und Fahrtzielerkennung mittels Verfahren des Maschinellen Lernens für einen elektronischen Blindenführhund

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Elektro- und Informationstechnik*
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Lutz Leutelt
Zweitgutachter: Prof. Dr. Pawel Buczek

Eingereicht am: 22.04.2021

Jonas Kempke

Thema der Arbeit

Videobasierte Linienbus- und Fahrtzielerkennung mittels Verfahren des Maschinellen Lernens für einen elektronischen Blindenführhund

Stichworte

Maschinelles Lernen, Texterkennung, elektronischer Blindenführhund

Kurzzusammenfassung

In dieser Arbeit wird eine Fahrtrichtungserkennung von Linienbussen über die Bilder einer Kamera entwickelt. Die Erkennung soll in einem elektronischen Blindenhund eingesetzt werden, um Menschen mit eingeschränktem Sichtfeld die tägliche Fortbewegung zu erleichtern. Für die Erkennung der Busnummer und des Fahrtziels wird der Text aus den Anzeigen der Busse softwareseitig ausgewertet.

Jonas Kempke

Title of Thesis

Video based bus detection and route classification by means of machine learning for an electronic guide dog

Keywords

Machine learning, text classification, electronic guide dog

Abstract

This paper involves the development of a bus destination detection from scene images of a camera. The detection is a part of an electronic guide dog that should help visually impaired people to gain independence in their daily life. The text from the display of the bus is used to detect the bus number an route.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application programming interface
BGR	Blau, Grün, Rot
EAST	An Efficient and Accurate Scene Text Detector
FN	False Negative
FP	False Positive
GPS	Global Positioning System
GPU	Graphics processing unit
HAW	Hochschule für Angewandte Wissenschaften
HVV	Hamburger Verkehrsverbund
LED	Light-emitting diode
MNIST	Modified National Institute of Standards and Technology
MSE	Mean squared error
OCR	Optical character recognition
OpenCV	Open Computer Vision
ReLU	Rectified Linear Unit
TP	True Positive
YOLO	You only look once

1 Einleitung

1.1 Problemstellung und Motivation

Ende 2015 lebten in Deutschland laut dem Statistischen Bundesamt etwa 380.000 Blinde beziehungsweise sehbehinderte Menschen [37], welche täglich mit vielen aus ihrer Beeinträchtigung resultierenden Problemen im Alltag konfrontiert werden. Um den Menschen auch unter diesen Umständen ein selbstständiges Leben ermöglichen zu können, wird an vielen Stellen Inklusion betrieben.
Bei der Fortbewegung sind derzeit jedoch noch Grenzen gesetzt, da vollständig selbstfahrende Autos auf den deutschen Straßen noch nicht zugelassen sind [3]. Deshalb sind sehbehinderte Menschen oft auf die öffentlichen Verkehrsmittel angewiesen.
Bei der Bahn gibt es schon einige Angebote, um Personen mit eingeschränkter Sehkraft das Reisen zu ermöglichen. So sind an Bahnhöfen die Wege zu den Gleisen auf dem Boden markiert. Bei einem einfahrenden Zug wird zudem die Linie und das Fahrtziel laut angesagt, sodass auch blinde Fahrgäste wissen, welche Bahn vor ihnen steht.
Anders ist dies beim Linienbusverkehr. In der Regel stehen hier die Fahrtziele nur auf einer großen Anzeige und werden an den Haltestellen nicht akustisch angesagt. Dies stellt ein Problem für sehbehinderte Fahrgäste dar, da sie beim Busfahrer oder bei anderen Fahrgästen die Informationen erfragen müssen und sich dies oft als schwierig erweist [26].
Um Menschen mit eingeschränktem Sichtfeld ein eigenständiges Leben zu ermöglichen, wird in dieser Arbeit als Teilprojekt eines elektronischen Blindenhundes die Erkennung von Linienbussen aus Kamerabildern entwickelt.

1.2 Zielsetzung

Um Menschen mit eingeschränktem Sichtfeld in alltäglichen Situationen helfen zu können, wird an der HAW Hamburg ein elektronischer Blindenhund entwickelt, welcher verschie-

dene Aufgaben übernehmen kann. Eine dieser Aufgaben ist die Erkennung der Fahrtrichtung von Linienbussen, welche in dieser Ausarbeitung betrachtet wird. Diese funktioniert über die Aufnahmen einer Kamera. Dabei sollen aus den Bildern die Informationen der Fahrtrichtungsanzeigen der Busse ausgelesen werden. Hierfür müssen zunächst Busse in den Bildern lokalisiert werden und anschließend ihre Anzeigen ausgelesen werden. Zuletzt soll das Ergebnis ausgegeben werden.

Abbildung 1.1: Idee der Buserkennung und Textklassifikation mit anschließender Ausgabe [34]

Der Schritt 1 in Abbildung 1.1 wurde bereits in [2] übernommen, wodurch in dieser Arbeit der Fokus auf das Auslesen der Anzeigen und somit auf die Schritte 2 und 3 fällt.

1.3 Aufbau der Arbeit

Diese Arbeit ist in mehrere Abschnitte unterteilt. Zunächst werden im Kapitel 2 die benötigten theoretischen Grundlagen für die weitere Ausarbeitung erörtert. Anschließend werden in Kapitel 3 die Anforderungen an das System definiert. Hierfür werden die Einsatzszenarien analysiert und bisherige Entwicklungen in diesen Bereich beleuchtet. Anschließend wird ein Konzept für die Entwicklung in Kapitel 4 aufgestellt. Anhand dieses Konzeptes wird in Kapitel 5 die Fahrtrichtungserkennung entwickelt. Dabei werden unter anderem verschiedene Ansätze zur Lösung diverser Herausforderungen untersucht und verglichen. Die Ergebnisse aus der Entwicklung werden im Kapitel 6 ausgewertet und mit den aufgestellten Anforderungen verglichen. Zuletzt werden im Kapitel 7 die Ergebnisse der Arbeit zusammengefasst und ein Ausblick für weitere Arbeiten aufgestellt.

2 Grundlagen

In diesem Kapitel werden zunächst die benötigten Grundlagen für die Entwicklung der Fahrtrichtungserkennung dargestellt. Insbesondere wird die Funktionsweise von verschiedenen Möglichkeiten zu Erkennung von Objekten beziehungsweise Buchstaben erklärt.

2.1 Hamburger Verkehrsverbund

Der HVV ist ein Zusammenschluss aus 29 Verkehrsunternehmen, die in und um Hamburg täglich 2,6 Millionen Fahrgäste transportieren [16]. Mit 4385 Fahrzeugen werden insgesamt 10021 Haltestellen durch 739 Linien miteinander Verbunden. Zu den Fahrzeugen gehören Busse, Bahnen und auch Fähren.

2.2 OpenCV

OpenCV [30] ist eine in C++ geschriebene, frei benutzbare Bibliothek für verschiedene Algorithmen der Bildverarbeitung. Diese sind aus vielen relevanten Forschungsarbeiten zusammengestellt und für die Nutzung optimiert. Dadurch können unterschiedliche wichtige Bildverarbeitungsschritte in hoher Geschwindigkeit gelöst werden. Des Weiteren sind auch viele Algorithmen zur Klassifizierung von Objekten enthalten.
Mit OpenCV können Bilder im BGR-Farbraum eingelesen und verarbeitet werden. Die einzelnen Farben können dabei Werten von 0 bis 255 annehmen, wobei 255 der vollen Helligkeit des jeweiligen Kanals entspricht.

2.3 Klassische Bildverarbeitung

2.3.1 Allgemein

Eine Möglichkeit zur Klassifizierung von Objekten in Bildern ist die klassische Bildverarbeitung. Hierbei müssen zuvor festgelegte Merkmale der zu klassifizierenden Objekte extrahiert werden. Mithilfe dieser kann ein Merkmalsraum aufgespannt werden, über welchen ein geeigneter Klassifikator die Zuordnung bestimmen kann. Wichtig ist eine gute Auswahl der Merkmale, damit im besten Fall mehrere überlappungsfreie Cluster der Klassen entstehen (siehe Abbildung 2.1).

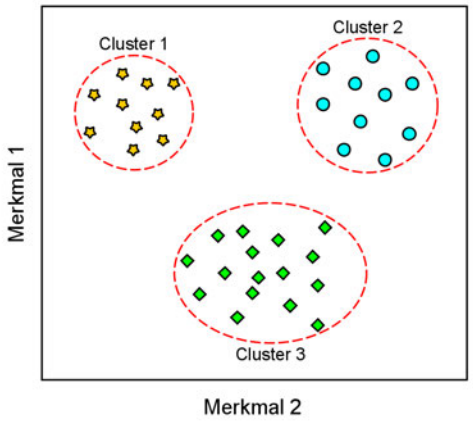
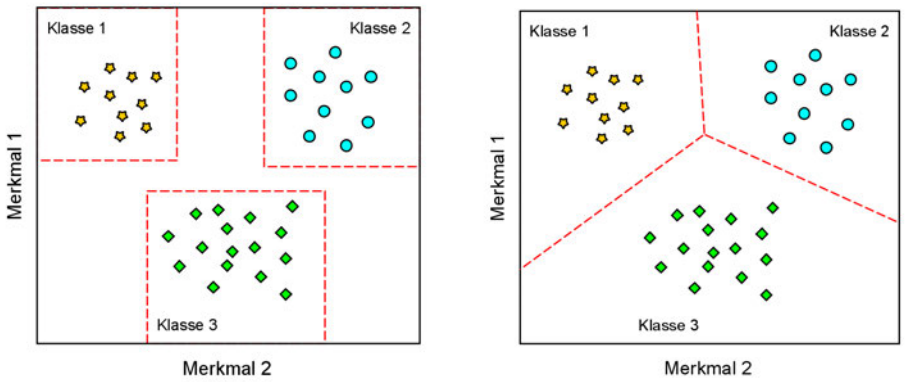


Abbildung 2.1: Zweidimensionaler Merkmalsraum mit optimaler Clusterbildung

2.3.2 Geometrische Klassifikatoren

Als Klassifikatoren können zum Beispiel der Quader-Klassifikator oder die Support Vector Machine [6] eingesetzt werden (vergleiche Abbildung 2.2). Allerdings haben diese das Problem, dass bei einer Vermischung der Klassen im Merkmalsraum feste Entscheidungsgrenzen schwer zu finden sind.



(a) (b)

Abbildung 2.2: Beispiel der Merkmalsraumaufteilung mit einem Quader Klassifikator (a) und einer Support Vector Machine (b)

2.3.3 k-Nearest-Neighbour Klassifikator

Der k-Nearest-Neighbour Klassifikator funktioniert über die Messung der Abstände des zu erkennenden Objektes zu bekannten Stichproben im Merkmalsraum (siehe Abbildung 2.3). Dabei wird das Objekt der Klasse zugewiesen, welche unter den k nächsten Stichproben am häufigsten vorkommt. So ist auch eine gewisse Robustheit gegenüber einzelnen Ausreißern gegeben [23].

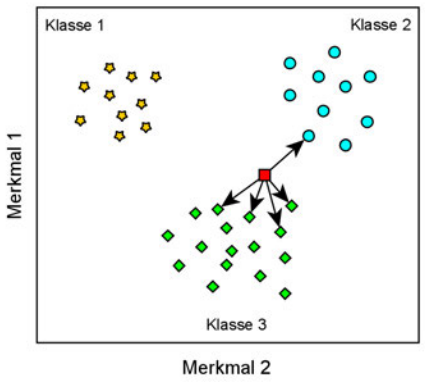


Abbildung 2.3: k-Nearest-Neighbour Algorithmus für die Klassifizierung eines neuen Objektes mit $k = 5$

Dieser Klassifikator ist leicht zu implementieren, indem die Merkmale von bekannten Objekten eintrainiert werden. Dies ist zum Beispiel über OpenCV möglich.

2.4 Neuronale Netze

2.4.1 Allgemein

Ein weiterer Ansatz zur Klassifizierung von Objekten in Bildern ist über künstliche neuronale Netze, welche Nervensystemen wie im menschlichen Gehirn nachempfunden sind [12]. Durch Training können sie an die gewünschte Aufgabe angepasst werden.
Ein künstliches neuronales Netz besteht dabei aus Neuronen, welche mehrere gewichtete Eingänge besitzen können und diese über eine zuvor bestimmte Aktivierungsfunktion auswerten und ausgeben. Die Neuronen sind wie in Abbildung 2.4 in verschiedenen Schichten aufgebaut. So gibt es immer eine Eingangsschicht, welche zum Beispiel den Pixeln eines Bildes entsprechen kann und es gibt eine Ausgabeschicht, welche die Ergebnisse des Netzes widerspiegelt. Dazwischen können je nach Komplexität der Aufgabe viele weitere Schichten liegen, welche verborgene Schichten (englisch: "hidden layer") genannt werden. Die Ausgänge der Neuronen einer Schicht werden gewichtet als Eingänge der Neuronen der nächsten Schicht verwendet. Bei zwei oder mehr verborgenen Schichten wird von Deep Learning gesprochen [8].

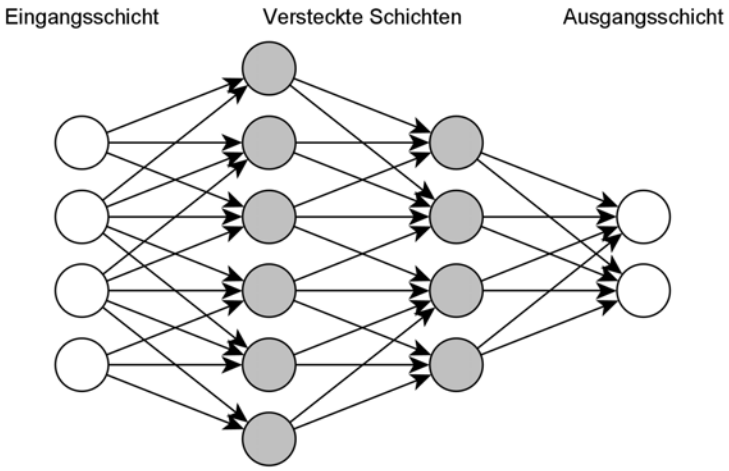


Abbildung 2.4: Aufbau eines sehr einfachen neuronalen Netzes mit zwei versteckten Schichten

Die einzelnen Neuronen lassen sich durch die Formel

$$a = f(\mathbf{w} \cdot \mathbf{x} + b) \tag{2.1}$$

beschreiben, wobei $\mathbf{w}$ die Gewichte sind, $\mathbf{x}$ die Eingänge und b das Bias, welches einem Offset entspricht. Das Ergebnis aus den gewichteten Eingängen mit dem Bias wird in die Aktivierungsfunktion gegeben und berechnet so den Ausgang des Neurons. Beim sogenannten feedforward werden von der Eingangsschicht ausgehend die Ausgänge der Neuronen der nachfolgenden Schichten wie folgt berechnet:

$$\mathbf{a}^{Ln} = f(W^{Ln} \cdot \mathbf{a}^{L(n-1)} + \mathbf{b}^{Ln}) \tag{2.2}$$

Durch geschickte Vektortransformationen kann die Berechnung des Ausgangs einer Schicht für beliebig viele Bilder gleichzeitig erfolgen.
Damit das neuronale Netz bei der Eingabe eines Bildes auch das richtige Ergebnis liefert, muss es trainiert werden. Das Training basiert auf dem Gradientenabstieg. Hierfür muss zunächst nach einer Vorhersage des neuronalen Netzes überprüft werden, wie gut das berechnete Ergebnis ist. Als Maß für diesen Soll-Ist-Vergleich werden Kostenfunktionen benutzt. Diese vergleichen die Ergebnisse der letzten Schicht mit dem gewünschten Ergebnis. Häufig wird die MSE Kostenfunktion verwendet, welche über

$$C = \frac{1}{n} \sum_{x=1}^{n} (a_x^L - y_x(x))^2 \tag{2.3}$$

definiert ist. Dabei entspricht n der Anzahl der Neuronen der letzten Schicht, a_x^L deren Ausgängen und y_x dem gewünschten Ergebnis dieser Schicht.
Eine alternative Kostenfunktion, um Lernblockaden zu überwinden, ist die Cross-Entropy [27]

$$C = -\frac{1}{n} \sum_{x=1}^{n} y_x \cdot ln(a_x^L) + (1 - y_x) \cdot ln(1 - a_x^L), \tag{2.4}$$

welche abweichenden Ergebnissen höhere Kosten zuweist als die MSE-Kostenfunktion. Sie wird verwendet, wenn ein binärer Ausgang einer Schicht erwartet wird und das Ergebnis einer prozentualen Sicherheit entspricht. Diese Funktion arbeitet über eine Art Fallunterscheidung, bei der die Kosten an den Neuronen, die 1 entsprechen sollen, durch

$$C_x = -ln(1 - a_x^L) \tag{2.5}$$

berechnet werden, während bei den Neuronen, die 0 ergeben sollen,

$$C_x \;=\; -ln(a_x^L) \tag{2.6}$$

verwendet wird.

Beim Training wird über den Gradientenabstieg versucht, die Gewichte und Bias so zu verändern, dass die Kosten minimal werden. Die Änderung der Gewichte lässt sich aus

$$\Delta w \;=\; -\eta \cdot \nabla_w C \tag{2.7}$$

berechnen, wobei ∇C der nach den einzelnen Gewichten abgeleiteten Kostenfunktion entspricht und η der Lernrate, welche die Geschwindigkeit des Abstiegs bestimmt. Gleiches gilt für die Änderungen des Bias

$$\Delta b \;=\; -\eta \cdot \nabla_b C. \tag{2.8}$$

Bei komplexen, tiefen Netzwerken ist die Berechnung unter Berücksichtigung aller Gewichte zu rechenaufwendig, sodass stattdessen die Änderungen schichtenweise berechnet werden. Angefangen bei der Ausgabeschicht wird das Netz so unter Berücksichtigung der Ergebnisse der hinteren Schichten und der jeweiligen Aktivierungsfunktion bis zur Eingangsschicht durchgelaufen. Dieser Vorgang nennt sich Backpropagation.
Da die Kostenfunktion für sehr viele Trainingsdaten aufwendig zu berechnen ist, werden diese in zufällige Batches einer zuvor festgelegten Größe eingeteilt und der Gradientenabstieg für diese einzeln berechnet. Dieser Vorgang wird für den gesamten Datensatz durchgeführt und wird als Epoche bezeichnet.
Durch den Gradientenabstieg können so über mehrere Epochen die Gewichte und das Bias angepasst werden, sodass die Kosten minimal werden und somit das gewünschte Ergebnis geliefert wird. Die Lernrate muss jedoch auf das Netz angepasst werden, da Oszillation entstehen kann oder sogar globale Minima übersprungen werden, wenn diese zu groß gewählt ist. Andersherum kann bei einer zu kleinen Lernrate das Training wesentlich länger dauern oder der Gradientenabstieg in lokalen Minima enden. Abbildung 2.5 zeigt die unterschiedlichen Verläufe.

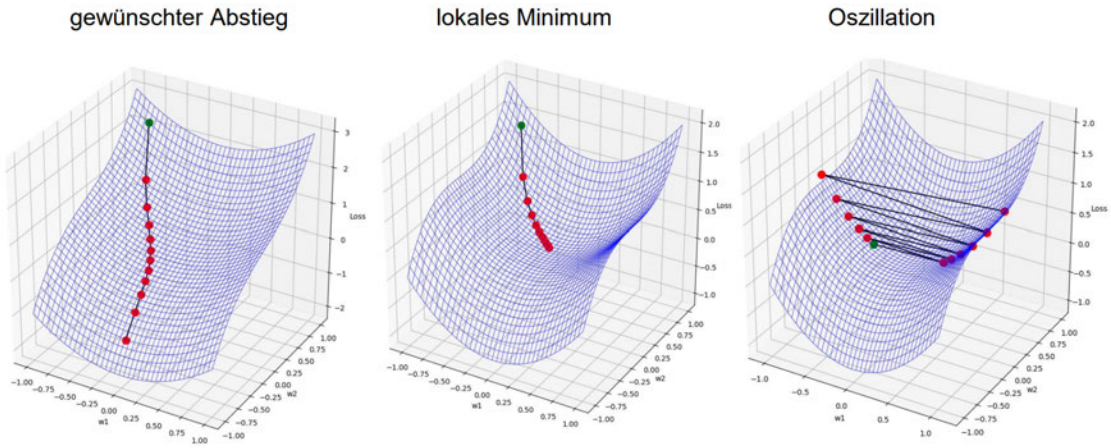


Abbildung 2.5: Gradientenabstieg mit gewünschtem Verlauf, mit zu kleiner und mit zu großer Lernrate [10]

Um den Gradientenabstieg schneller zu gestalten und kleine lokalen Minima zu umgehen, kann ein Momentum eingesetzt werden, welches die Bewegung des Gradientenabstiegs beibehält. Dabei wird ein Geschwindigkeitsvektor

$$m_{neu} = \beta \cdot m + \eta \cdot \nabla C \tag{2.9}$$

errechnet, welcher den Gradienten als Beschleunigung interpretiert. Die Gewichte werden dann mit

$$w_{neu} = w + m_{neu} \tag{2.10}$$

berechnet. Das Moment β liegt dabei zwischen 0, was dem normalen Gradientenabstieg entspricht, und 1, wodurch eine reibungsfreie Bewegung simuliert wird.

Als Aktivierungsfunktion kommen mehrere Typen in Frage. Der einfachste Ansatz ist eine Sprungfunktion, welche einem binären Ausgang entspricht. Für die Bildverarbeitung ist diese jedoch nicht optimal, da sich der Ausgang des Neurons bei kleinen Änderungen am Eingang schlagartig ändern kann und so ein training nur schwer möglich ist.

In der Bildverarbeitung wird deshalb häufig mit der Sigmoid-Funktion

$$\sigma(z) \;=\; \frac{1}{1+e^{-z}} \qquad mit \;\; z \;=\; \mathbf{w}\cdot\mathbf{x}+b \tag{2.11}$$

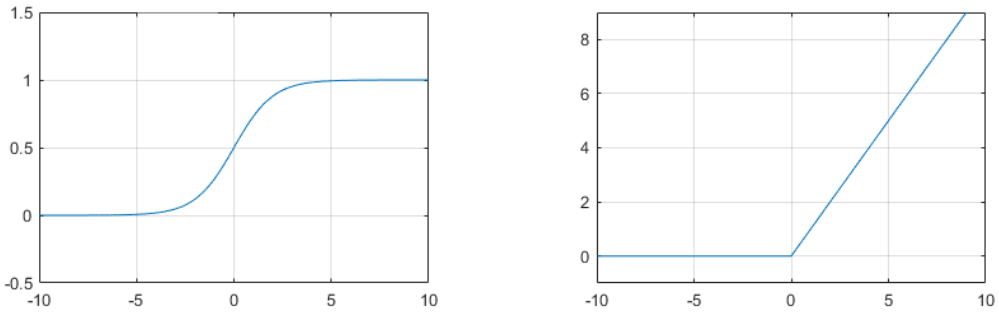
gearbeitet. Diese hat einen ähnlichen Verlauf wie die Sprungfunktion (vergleiche Abbildung 2.6), ist jedoch stetig und ermöglicht so beim Training einen funktionierenden Gradientenabstieg. Die Ableitung der Sigmoid-Funktion

$$\sigma'(z) \;=\; \sigma(z)\cdot(1-\sigma(z)) \tag{2.12}$$

kann zudem leicht mithilfe der Funktion selber berechnet werden.
Ein Problem der Sigmoid-Funktion ist jedoch, dass bei größeren absoluten Werten die Funktion in Sättigung geht, sodass sich das Training verlangsamen kann. Die Lösung hierfür ist eine Funktion ohne Sättigung wie die ReLU (Rectified Linear Unit) [21]

$$f(z) = \max(0, z). \tag{2.13}$$

Auch diese Funktion lässt sich mit geringem Aufwand differenzieren und noch schneller als die Sigmoid-Funktion berechnen. Allerdings ist die ReLU-Funktion bei negativen Werten in der Sättigung, weshalb es noch weitere Aktivierungsfunktionen gibt, die dieses Problem lösen können, wie die Leaky ReLU oder ELU. Allerdings bringen auch diese Probleme mit sich, da sie nicht stetig differenzierbar oder nur aufwändig zu berechnen sind. Deshalb wird häufig die ReLU-Aktivierungsfunktion verwendet [7].



(a) (b)

Abbildung 2.6: Häufig verwendete Aktivierungsfunktionen: (a) Sigmoid; (b) ReLU

Eine weitere wichtige Aktivierungsfunktion ist die Softmax-Funktion

$$\hat{p}_k = \frac{e^{x_k}}{\sum_i^K e^{x_i}}, \tag{2.14}$$

welche oft in der Ausgangsschicht verwendet wird, um das Ergebnis in einer prozentualen Sicherheit zu erhalten. Dabei entspricht K der Anzahl der Neuronen in der entsprechenden Schicht und x_k dem berechneten Eingang des Neurons k, mit welchem die Wahrscheinlichkeit der Zugehörigkeit zur Klasse k $\hat{p}_k$ berechnet wird. Hohe Werte werden aufgrund der exponentiellen Berechnung stärker gewichtet.

2.4.2 Faltende Netze

Beim Aufbau der Netze gibt es unterschiedliche Ansätze. Eine mögliche Struktur stellen sogenannte ”dense layer” dar, welche alle Neuronen der vorherigen Schicht mit allen Neuronen der nächsten Schicht verbinden (ähnlich wie in Abbildung 2.4). Dieser Ansatz ist bei großen und komplexen Netzen jedoch nicht optimal, da

$$n_{w^L} = n_{N^L} \cdot n_{N^{L-1}} \tag{2.15}$$

Gewichte sowie n_{N^L} Bias für eine Schicht L benötigt werden. Des Weiteren korrelieren nicht alle Pixel eines Bildes miteinander. So ist der gegenseitige Einfluss von zwei Pixeln, die sehr weit auseinander liegen, meist gering.
Eine Lösung hierfür sind faltende Schichten (englisch: ”convolutional layer”), die mit einem kleinen Filterkern (häufig 3x3, 5x5 oder 7x7) die vorige Schicht durchlaufen. Dabei können die Filterkerne auch eine dritte Dimension besitzen, wodurch zum Beispiel die drei Farbkanäle berücksichtigt werden können. Dieser Vorgang kann mit verschiedenen Filtern durchgeführt werden, sodass eine Vielzahl von sogenannten ”features maps” entstehen, die zusammen eine Schicht im Netz ergeben. Auch die Anzahl der Bias kann so verringert werden, da lediglich eines pro feature map benötigt wird. Die Größe einer feature map ist abhängig von verschiedenen Einstellungen. Die Schrittweite der Filterkerne dient als Dividend der Dimension. Die Ränder sind abhängig vom sogenannten ”padding”, ohne welches sich die Dimensionen um maximal der Größe des Filterkerns minus 1 verringern kann. Um bei faltenden Netzen die Daten zu reduzieren, werden zusätzlich sogenannte ”pooling layer” eingesetzt, welche zum Beispiel über einen 2x2 Filterkern und einer Schrittweite von 2 die vorige Schicht in ein Viertel ihrer Größe reduzieren können.

Dabei können die Werte im Filterkern gemittelt oder auch nur der maximale Wert verwendet werden.
Um am Ende eines Netzes aus den zweidimensionalen Schichten, in denen sich die extrahierten Merkmale befinden, eine Vorhersage für die verschiedenen Klassen zu treffen, werden über ein sogenanntes ”flatten layer” diese Schichten in eine eindimensionale Schicht umgewandelt. Abschließend werden meist wenige ”dense layer” eingesetzt, um die Klassifizierung zu übernehmen. Um dabei robust gegenüber kleinen Änderungen zu sein, werden hier ”dropout layer” eingebaut, welche einen zuvor festgelegten Anteil ihrer Neuronen zufällig ausschalten. Dadurch werden dieselben Funktionen von mehreren Neuronen übernommen. Bei der Klassifikation werden dann alle Neuronen verwendet. Die Struktur eines solchen Netzes ist in Abbildung 2.7 festgehalten.

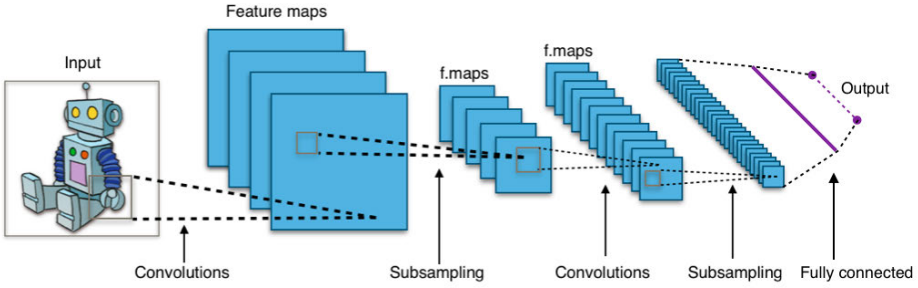


Abbildung 2.7: Aufbau eines faltenden Netzes [51]

2.4.3 Haar Cascade Classifier

Ein Haar Cascade Classifier [41] ist eine ältere Methode zur Erkennung von Gesichtern in Bildern. Dabei werden die Merkmale aus den Bildern über festgelegte Filter wie in Abbildung 2.8 extrahiert. Zu diesen gehören Kanten- und Linienfilter in verschiedenen Ausrichtungen. Aus jedem der Filter wird ein Wert durch Subtraktion der Summe der Pixel im weißen Feld von der Summe der Pixel im schwarzen Feld berechnet.

Abbildung 2.8: Filter der Haar Cascade Classifier [49]: (1) vertikaler Kantenfilter; (2) horizontaler Kantenfilter; (3) horizontaler Linienfilter; (4) diagonaler Linienfilter

Für die so berechneten Werte kann beim Training ein Schwellwert gefunden werden, welcher entscheidet, ob das Objekt im Bild vorhanden ist. Um eine möglichst gute Klassifizierung zu erhalten, werden die Filter mit den geringsten Fehlerraten beibehalten.
Ein Problem dieser Methode ist die Robustheit, da zum Beispiel leicht seitliche Aufnahmen von Gesichtern häufig nicht erkannt werden [9].
Zwar wurde diese Methode ursprünglich für Erkennung von Gesichtern entwickelt, jedoch können auch andere Objekte so klassifiziert und in Bildern lokalisiert werden.

2.4.4 YOLO

YOLO [34] ist eine neuronale Netzwerk-Architektur, welche die Lokalisierung und Klassifizierung von trainierten Objekten in Bildern zulässt (vergleiche Abbildung 2.9).

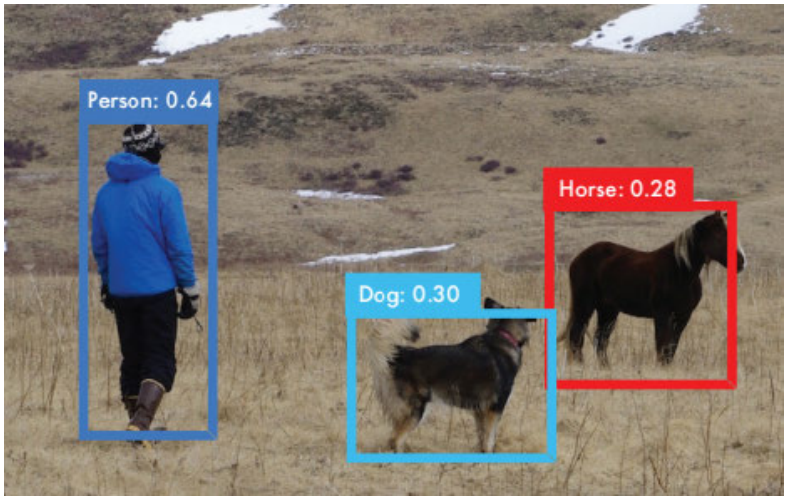


Abbildung 2.9: Detektion verschiedener Objekte innerhalb eines Bildes [34]

YOLO funktioniert über die gleichzeitige Lokalisierung und Klassifizierung aller im Bild vorhandenen Objekte. Hierfür wird das Bild in ein S x S Raster eingeteilt. Der Mittelpunkt eines Objektes bestimmt die Zugehörigkeit zum jeweiligen Feld, in welchen B Objekte erkannt werden können. Zu jedem detektierten Objekt gehören die Parameter x, y, w, h, welche den Rahmen von diesem bestimmen. Dazu kommt ein Parameter zur Sicherheit des Netzes, ob es sich tatsächlich um ein Objekt handelt und wie passend der gefundene Objektrahmen ist. Unabhängig von der Anzahl der detektierten Rahmen von Objekten in einer Zelle wird die Vorhersage für die einzelnen Klassen nur einmal pro Zelle vorgenommen. So ergibt sich bei der Ausgabe von YOLO, anders als bei den zuvor

besprochenen Netzen, kein Vektor, sondern ein S x S x $(B \cdot 5 + C)$ Tensor, wobei C der Anzahl der trainierten Klassen entspricht. Die Netzwerkarchitektur von YOLO entspricht einer modifizierten Variante vom GoogLeNet [38].

2.4.5 EAST

EAST [54] ist eine Pipeline Struktur zur Erkennung und Lokalisierung von Text in Bildern mit drei Farbkanälen. Die Breite und Höhe der Bilder müssen jeweils ein Vielfaches von 32 betragen. Mithilfe eines trainierten faltenden neuronalen Netzes können Wörter oder ganze Textzeilen erkannt werden. Das Netz gibt die Position, Größe und den Winkel der gefundenen Textfelder zusammen mit einer Sicherheit der Erkennung aus. Anschließend können über ein Schwellwertverfahren erkannte Felder mit geringer Sicherheit aussortiert werden. Durch Non-Maximum Suppression [18] werden zuletzt überlappende Felder mit gleicher Klasse zu einem Feld zusammengeführt. Die Struktur von EAST ist in Abbildung 2.10 zu sehen.

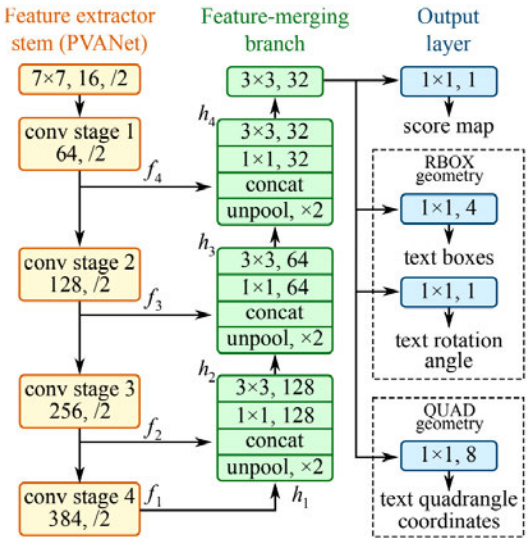


Abbildung 2.10: Aufbau des faltenden neuronalen Netzes von EAST [53]

Durch die Struktur des Netzes ist EAST sehr schnell und kann mit einer Titan X GPU über 13.2 Bilder pro Sekunde mit einer Auflösung von 1280×720 Pixeln bearbeiten [54].

3 Anforderungsanalyse

Das Ziel dieses Kapitels ist es, die Anforderungen für die Linienbuserkennung aus verschiedenen realen Einsatzfällen abzuleiten, mögliche Probleme zu identifizieren und den Einsatzbereich abzugrenzen. Hierfür werden auch bisherige Arbeiten zu diesem Thema herangezogen.

3.1 Bisheriger Stand

3.1.1 Blindenhund

Diese Arbeit ist ein Teilaspekt für die Entwicklung eines elektronischen Blindenhundes. Dieser soll Menschen mit eingeschränktem Sichtfeld helfen, sich selbstständig im Alltag zu bewegen. Hierfür sollen verschiedene Anwendungen, wie zum Beispiel die Buserkennung, auf einem mobilen, handlichen System integriert werden. Die Plattform und die benötigte Hardware werden parallel zu der Linienbuserkennung an der HAW Hamburg entwickelt.

3.1.2 Buserkennung

Im Rahmen der Arbeiten zum elektronischen Blindenhund wurde bereits die Linienbuserkennung in [2] adressiert. Diese Arbeit beschreibt einen Ansatz zur Detektion von Objekten in Bildern über einen Haar Cascade Classifier. Über OpenCV wurde eine solche Klassifizierung mit online verfügbaren Bildern von Busfronten [4] trainiert.
Um die Genauigkeit dieser Busdetektion zu prüfen, werden zwei Testvideos verwendet. In einem dieser Videos sind heranfahrende Busse zu sehen, während das zweite Video eine Kreuzung mit Passanten und verschiedenen Autos zeigt.
In den 336 Bildern des ersten Videos wird der enthaltene Bus 264-mal erkannt, jedoch wird auch 28-mal ein anderes Objekt fälschlicherweise als Bus identifiziert. Bei dem Video ohne Bus wird bei 323 Bildern in 56 Fällen ein Bus erkannt. Daraus ergeben sich

insgesamt 264 "True Positive" (TP), 84 "False Positive" (FP) und 72 "False Negative" (FN). Aus diesen Werten lässt sich die Precision

$$P = \frac{TP}{TP + FP} = 75,86\% \tag{3.1}$$

berechnen sowie der Recall

$$R = \frac{TP}{TP + FN} = 78,57\%. \tag{3.2}$$

Im perfekten Fall sind die Precision und der Recall je 1. Dies ist für komplexe Aufgaben nicht erreichbar, jedoch können höhere Genauigkeiten als hier erreicht werden. Die Klassifizierung der Busse kann dennoch als Basis für die weitere Texterkennung verwendet werden. Dies ist mit dem geringen Zeitaufwand für eine Klassifikation zu begründen, wodurch bei einem nicht erkannten Bus eine erneute Detektion ermöglicht wird. Auch die falsch positiv erkannten Objekte sind nicht zwangsläufig ein Problem, da in der weiteren Verarbeitung der Text der Anzeigen gelesen wird und in diesen Fällen keine sinnvollen Ergebnisse entstehen. Es darf jedoch nicht zu viele falsch detektierte Busse geben, da sonst die Bearbeitung einer Aufnahme zu lange dauert.

Die Texterkennung soll in [2] über klassische Bildverarbeitung mittels eines k-Nearest-Neighbor-Algorithmus erfolgen. Eine zuverlässige Klassifizierung lässt sich hiermit jedoch noch nicht durchführen. Vielversprechende Möglichkeiten, die Genauigkeit zu verbessern, sollen die Verbesserung der Bildvorverarbeitung, welche bisher aus einem einfachen Schwellwertverfahren besteht, und des k-Nearest-Neighbor-Algorithmus sein.

3.1.3 Vorangegangene Forschungen

Die Erkennung von Bussen und ihrer Fahrtrichtung, um Menschen mit eingeschränktem Sichtfeld die Fortbewegung zu erleichtern, wurde schon in verschiedenen Arbeiten diskutiert.
In [17] wird ein Ansatz zur Erkennung von Bussen in Bildern und ein erster Ansatz zur Lokalisierung der Texte auf den Anzeigen beschrieben. Mit einer Genauigkeit von 80.93% ist die Buserkennung nicht wesentlich besser als die in [2] implementierte Methode.
Eine zweite Arbeit [5] hat die Erkennung aus [17] für die Verwendung von Videos angepasst und speziell eine Lokalisierung der Busnummer realisiert. Seit der Veröffentlichung dieser Arbeit gibt es einige Fortschritte in der Textdetektion, sodass in dem Ausschnitt

eines Busses mithilfe von zum Beispiel EAST schnell und akkurat Texte lokalisiert werden können.
Erst die dritte und neuste Ausarbeitung [52] implementiert auch eine Klassifizierung der Busnummer. Dabei wird die Lokalisierung nach [5] vorgenommen. Die Klassifizierung der Busnummer erfolgt hierbei über die *Google Cloud Vision API* [13] und erreicht eine maximale Genauigkeit von 67.9%. Für die Texterkennung müssen allerdings die Bilder in die *Google Cloud* hochgeladen werden, sodass eine ständige Internetverbindung gegeben sein muss.

In dieser Arbeit soll neben der Busnummer auch das Fahrtziel erkannt werden, da zum Beispiel für Busse des Hamburger Verkehrsverbunds weder die Busnummer noch das Fahrtziel eindeutig sind [15] und somit erst durch die Verbindung beider Informationen, die korrekte Linie bestimmt werden kann.

3.2 Einsatzszenarien

3.2.1 Aussehen und Geometrie der Fahrzeuge

Eine Herausforderung des Themas stellt die globale Einsetzbarkeit der Buserkennung dar, denn in den verschiedenen Regionen und Ländern gibt es unterschiedliche Anforderungen an die Technik, das Design und die Kosten für einen Bus. Die eingesetzten Fahrzeuge unterscheiden sich zum Teil stark. Die Buserkennung muss deshalb gegebenenfalls darauf eingestimmt sein.
In den meisten Teilen Deutschlands werden typischerweise Niederflurbusse für den Linienbusverkehr eingesetzt. Jedoch werden teilweise auch Reisebusse, welche sich im Aussehen von den Niederflurbussen unterscheiden, für den Linienverkehr eingesetzt.
Einige Busse fahren auch mit einem Elektroantrieb, wodurch auf dem Dach dieser Fahrzeuge eine große Batterie befestigt ist. Dies verändert ebenfalls die Geometrie des Busses, da die Anzeige nicht mehr am oberen Rand befestigt ist.
Auch die Farben der Linienbusse können zum Beispiel durch Außenwerbung stark variieren.

(a) (b)

Abbildung 3.1: Verschiedene Busarten, die in Deutschland eingesetzt werden: (a) Niederflurbus [42]; (b) Reisebus [48]

Deutlicher wird dies bei einem Blick in andere Länder, wie in Abbildung 3.2. So werden zum Beispiel in Syrien auch Busse in anderen Formaten eingesetzt. Teilweise besitzen diese keine elektronischen oder sogar gar keine Fahrtzielanzeigen.

Abbildung 3.2: Syrischer Bus mit anderer Bauweise zu den in Deutschland vorrangig eingesetzten Niederflurbussen [47]

Des Weiteren fahren zum Beispiel in London die typischen roten Doppeldecker-Busse (Routemaster). Bei diesen ist nicht nur die optische Erscheinung eine Herausforderung, sondern auch das Finden der Anzeige. Diese befindet sich hier etwa in mittlerer Höhe zwischen den beiden Decks des Busses. Bei älteren Modellen ist zudem die Nummer der Linie links vom Text, während bei neueren Auflagen die Nummer auf der rechten Seite steht (siehe Abbildung 3.3). Zuletzt ist auch zu beachten, dass in Großbritannien Linksverkehr herrscht, weshalb die Fahrgäste auf der linken Seite einsteigen. Daraus resultiert ein anderer Winkel der heranfahrenden Fahrzeuge zu den wartenden Personen.

(a) (b)

Abbildung 3.3: In London eingesetzte Routemaster Doppeldecker-Busse: (a) älteres Modell mit Busnummer links [43]; (b) neueres Modell mit Busnummer rechts [50]

Des weiteren werden auch verschiedene Anzeigetafeln verwendet (siehe Abbildung 3.3). So gibt es in Hamburg viele Busse mit oranger Schrift, bei denen die LEDs durchgehend leuchten. Es fahren jedoch auch viele Fahrzeuge mit weißer Schrift, bei denen die LEDs nur kurzzeitig aufleuchten, sodass auf Bildern oft nicht der gesamte Text zu sehen ist. Bei einer hohen Qualität der Bilder können die einzelnen LEDs wie in Abbildung 3.4 sichtbar werden, wodurch die Lesbarkeit verringert wird.

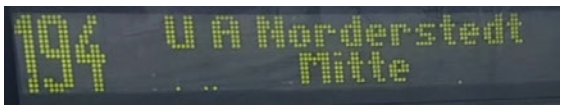


Abbildung 3.4: Bild einer Fahrtrichtungsanzeige bestehend aus einzelnen LEDs in hoher Auflösung und Schärfe

3.2.2 Sprache der zu erkennenden Anzeige

Um die Fahrtziele der Busse lesen zu können, muss die Texterkennung die Zeichen der Sprache auf der Anzeige kennen, denn viele Sprachen haben ihre eigenen Zeichen. In der deutschen Sprache wird das lateinische Alphabet verwendet, jedoch gibt es noch einige Sonderzeichen, wie zum Beispiel das 'ß'. In anderen Sprachen werden jedoch auch andere Alphabete verwendet, wie zum Beispiel im Russischen das kyrillische Alphabet.
Eine weitere Herausforderung bildet die Schreibrichtung, welche meistens von links nach rechts verläuft, die aber auch anders sein kann. Beispiel ist hierfür die arabische Sprache,

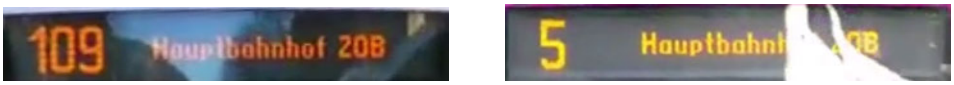
bei der von rechts nach links geschrieben wird.
Die Anforderungen zur Texterkennung sind besonders für Sprachen aus dem asiatischen Raum sehr hoch. In der chinesischen Schrift werden zum Beispiel 2500 verschiedene Zeichen benötigt, um 97.97% der im alltäglichen Gebrauch verwendeten Wörter lesen zu können [25]. Diese sind deshalb auch wesentlich komplexer und benötigen eine hohe Genauigkeit bei der Klassifizierung.

Abbildung 3.5: Chinesisches Zeichen für das Wort "Pferd"

Zuletzt muss auch beachtet werden, dass die meisten Fahrtziele der Linienbusse Eigennamen entsprechen und so keine typischen Wörter aus den jeweiligen Sprachen sind.

3.2.3 Umgebungsbedingungen durch Wetter- und Lichtverhältnisse

Auch äußerliche Einflüsse müssen beachtet werden. So soll die Texterkennung auch bei unterschiedlichen Wetterbedingungen funktionieren. Die Anzeige kann durch Schnee oder Regen verrauscht sein, sodass eine Auswertung schwerer fällt. Jedoch ist auch die Sonne ein Problem, da es zu teilweise sehr starken Reflexionen auf der Anzeige kommen kann. Eine vollständige Auswertung der Anzeige ist hier zum Teil nicht möglich (vergleiche Abbildung 3.6).



(a) (b)

Abbildung 3.6: Busanzeigen, die durch Spiegelungen (a) schlechter lesbar oder (b) teilweise unlesbar sind

Auch bei Nacht sollte die Auswertung der Anzeige funktionieren. Dies ist aufgrund der LED-Anzeigen möglich, jedoch sind die Konturen der Busse nur schlecht sichtbar, wodurch sich die Buserkennung erschwert. Zusätzlich stellen Spiegelungen durch Straßenbeleuchtungen eine Herausforderung dar.

3.3 Qualitative Anforderungen

3.3.1 Bildqualität

Die Genauigkeit der Texterkennung für die Busanzeigen ist gebunden an die Größe der Zeichen auf den Anzeigen und an die Bildqualität.
Bei einer perfekten Schärfe der Kamerabilder können wenige Pixel in der Höhe reichen, um einen lesbaren Text zu erhalten. Hierfür dürfen die relevanten Punkte in der realen Welt jedoch nur exakt auf den Pixeln liegen. Dies ist jedoch sehr unrealistisch, da auf Bildern eine gewisse Unschärfe durch einen nicht optimalen Fokus oder durch die Bewegung der Kamera oder des Busses entstehen kann. Eine Texterkennung ist nur dann sinnvoll, wenn die Schrift in ausreichender Größe und Schärfe zu sehen ist.
Um bei den heranfahrenden Bussen den Text schon früh aus den Anzeigen auszulesen zu können, ist eine möglichst hohe Qualität erforderlich. Eine Auflösung von 1920x1080 ist dabei sehr gut geeignet, da eine hohe Qualität geliefert wird, aber die Anzahl der Pixel nicht so hoch ist, dass eine Verarbeitung möglicherweise zu lange dauert. Wichtig ist, dass die Schärfe des Bildes möglichst hoch ist, sodass alle Pixel wertvolle Informationen enthalten und nicht stark mit ihren Nachbarn korrelieren. Um eine immer ähnliche Schärfe des Bildes zu gewährleisten, sollte die Kamera mit geschlossener Blende arbeiten, sodass Busse in verschiedenen Entfernungen gleichzeitig im Fokus liegen können [33]. Allerdings muss hierfür viel Licht vorhanden sein, da sonst eine höhere Belichtungszeit benötigt wird, was bei sich bewegenden Objekten zu verschwommenen Aufnahmen führt. Der Einfluss der Blende wird in Abbildung 3.7 verdeutlicht.

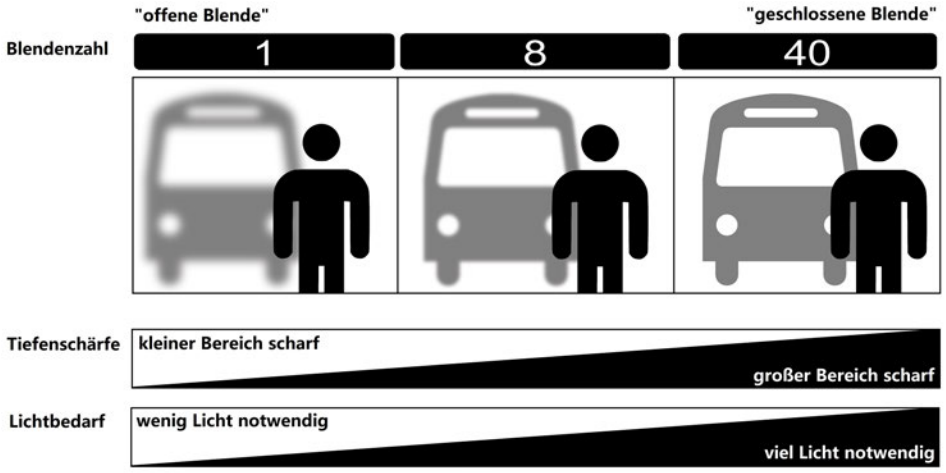


Abbildung 3.7: Zusammenhang der Tiefenschärfe eines Bildes und der Blende

Ein Zoom für die Kamera ist nicht zu empfehlen, da so das Einfangen der Busse im Bild schwieriger wird. Die Kamera sollte einen Bus aus wenigen Metern Entfernung komplett aufnehmen können, damit die Klassifizierung für langsam einfahrende Busse optimal ist. Ein Weitwinkel ist jedoch nicht zu empfehlen, da so die Entfernung der Fahrzeuge zum Nutzer geringer sein muss, um eine lesbare Anzeige auf dem Bild zu erhalten.

3.3.2 Zeitliche Anforderungen

Die Bus- und Fahrtrichtungserkennung muss in Echtzeit möglich sein, da die Anzeige der vorbeifahrenden Fahrzeuge nur relativ kurz zu sehen ist. Im besten Fall stellt sich der Nutzer an das Ende einer Haltestelle, damit der Bus auch nachdem er zum Stehen gekommen ist, noch im Bild der Kamera zu sehen ist und so länger detektiert werden kann. Allerdings stellen sich blinde Menschen häufig direkt am Haltestellenschild auf, um möglichst in der Nähe der vorderen Tür zu sein [26]. Deshalb muss eine Zeit für die Sichtbarkeit der Anzeige auf dem Bild der Kamera eingeschätzt werden und innerhalb dieser eine zuverlässige Fahrtrichtungserkennung erfolgen.

In verschiedenen Aufnahmen mit einem *Samsung Galaxy S7* kann bei einem einfahrenden Bus die Anzeige für etwa 5 Sekunden gelesen werden, bevor der Bus wieder aus dem Bild verschwindet. Innerhalb dieser Zeit muss eine Klassifikation der Fahrtrichtung möglich sein.

3.3.3 Zuverlässigkeit des Systems

Die Buserkennung sollte sehr zuverlässig arbeiten, allerdings ist eine unbedingte Bestimmung der Buslinie nicht nötig, da im Fall einer erfolglosen Klassifizierung auch eine Nachfrage beim Busfahrer möglich ist. Daher sollte das System nur Ausgaben liefern, wenn es sich sehr sicher ist.
In einigen Fällen ist die Bestimmung der Fahrtrichtung aus den Bildern der Kamera nicht möglich, da die Anzeige nicht lesbar ist, zum Beispiel durch Reflexion der Sonne oder weil sie von einem anderen Fahrzeug verdeckt wird. In den Fällen der erfolglosen Klassifizierung muss der Nutzer benachrichtigt werden, sodass eine rechtzeitige Nachfrage bei anderen Personen möglich ist und so der gewünschte Bus nicht verpasst wird.
Damit die Erkennung sinnvoll eingesetzt werden kann, muss diese bei guten Bedingungen zuverlässige Ergebnisse liefern.

3.4 Anforderungen

Aus den zuvor beschriebenen Einsatzszenarien und Herausforderungen können die Anforderungen an die Buserkennung abgeleitet und der genaue Einsatzbereich abgegrenzt werden.

3.4.1 Funktionale Anforderungen

- Die Anzeigen von Linienbussen des Hamburger Verkehrsverbunds sollen korrekt ausgelesen werden. Zur Anzeige gehören Busnummer und Fahrtziel.
- Die Fahrtrichtungserkennung soll mindestens tagsüber bei guten Lichtverhältnissen funktionieren.
- Die Erkennung muss in Echtzeit möglich sein.
- Das System muss auf einer mobilen Hardware lauffähig sein.

3.4.2 Nicht-funktionale Anforderungen

- Die Fahrtrichtungserkennung soll in Python geschrieben sein.
- Der Algorithmus soll mit frei verfügbarer Software auskommen.
- Eine Auswertung der Anzeigen muss ohne Internetverbindung möglich sein.
- Pro Sekunde soll mindestens eine Vorhersage getroffen werden können.

4 Konzept

In diesem Kapitel wird das Konzept zur Fahrtrichtungserkennung der Linienbusse sowie die Vorgehensweise bei der Entwicklung vorgestellt. Zudem wird der Ablauf des gesamten Systems geschildert.

4.1 Allgemeiner Ablauf

Die Erkennung von Linienbussen und ihren Fahrtzielen funktioniert nach dem Ablauf in Abbildung 4.1.

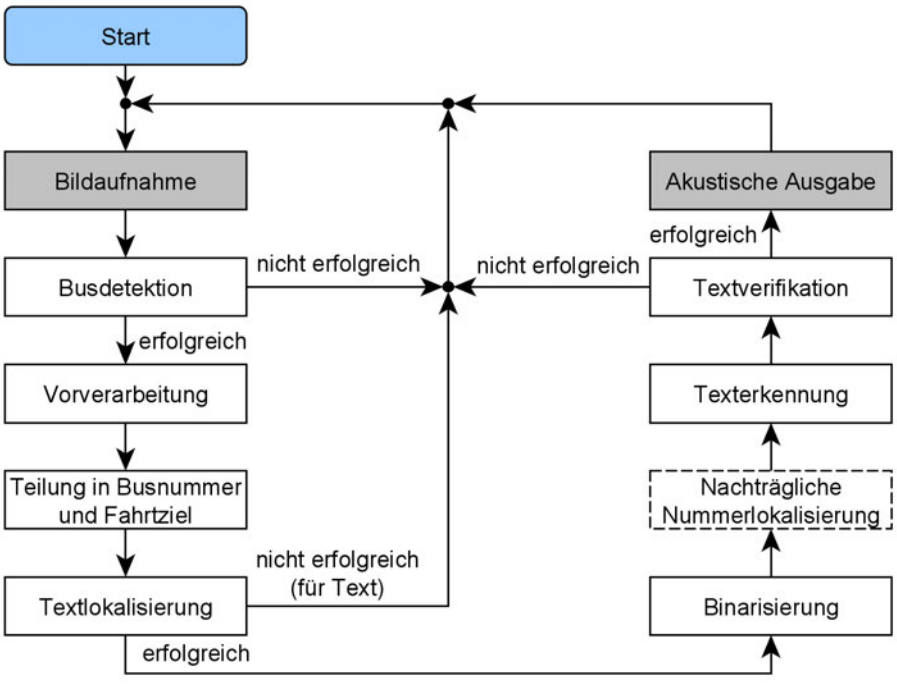


Abbildung 4.1: Ablaufdiagramm der Buserkennung

Zunächst werden die sichtbaren Busfronten aus dem Bildmaterial extrahiert. Sofern keine gefunden werden, kann direkt das nächste Bild verarbeitet werden. Im Falle einer erfolgreichen Erkennung muss das Bild für die weiteren Schritte vorverarbeitet werden. Anschließend erfolgt eine Aufteilung der Anzeige in Busnummer und Fahrtziel. In den jeweiligen Bildausschnitten wird der Text für die Klassifikation lokalisiert. Sofern dies für das Fahrtziel nicht möglich ist, wird die Verarbeitung mit dem nächsten Bild gestartet. Wenn jedoch erfolgreich Text identifiziert werden kann, soll dieser mit einem geeigneten Klassifikator ausgelesen werden. Hierfür muss zunächst das Bild binarisiert werden und, falls die Lokalisierung für die Nummer fehlgeschlagen ist, eine nachträgliche Lokalisierung durchgeführt werden. Die gelesenen Daten können zuletzt mit einer Datenbank möglicher Fahrtziele abgeglichen werden, sodass eine akustische Ausgabe nur bei hoher Sicherheit des Ergebnisses erfolgt.

4.2 Busdetektion

Im Ansatz aus [2] wird eine Detektion und Lokalisation von Bussen in Bildern über einen Haar Cascade Classifier realisiert. Zwar ist die Genauigkeit dieses Klassifikators mit einer Precision von $P = 75,86\%$ und einem Recall von $R = 78,57\%$ noch verbesserungswürdig, jedoch können falsch-positive Ergebnisse aus der Busdetektion mit der Texterkennung aufgefangen werden, da kein Text mit einer passenden Haltestelle gefunden werden sollte. Falsch-negative Ergebnisse können durch die hohe Geschwindigkeit der Detektion ausgeglichen werden. Des Weiteren liefert die Lokalisierung der Buserkennung konstant gute Ergebnisse, sodass die Busfronten stets ähnlich umrahmt werden. Aus diesen Gründen kann die Busdetektion aus [2] als Grundlage für die weiteren Schritte verwendet werden.

4.3 Vorverarbeitung

Die aufgenommenen Bilder sind teilweise für das Auslesen der Anzeige nicht gut geeignet, da zum Beispiel einige Bilder in einer so hohen Auflösung sind, dass die einzelnen LEDs der Anzeigen sichtbar werden. Die Lesbarkeit verringert sich hierdurch, weshalb diesem Effekt entgegengewirkt werden muss.
Zudem können Reflexionen auftreten, welche ebenfalls die Lesbarkeit stark verringern. Deshalb werden die Bilder vorverarbeitet, um gewisse Störeffekte herauszufiltern.

4.4 Textlokalisierung

Um die Zeichen auf der Anzeige auslesen zu können, müssen diese zunächst lokalisiert werden. Grob kann die Lokalisierung von Busnummer und Fahrtziel über die Geometrie der Busse und ihrer Anzeigen geschehen, da diese für Fahrzeuge des Hamburger Verkehrsverbunds immer gleich aufgebaut sind. Dabei werden die Anzeigen in Busnummer und Fahrtziel aufgeteilt. Um genauere Ergebnisse zu erhalten, kann zusätzlich eine Textdetektion durch EAST in dem zuvor grob ausgelegten Feld gestartet werden. Im Unterabschnitt 5.3.2 wird näher auf diesen Schritt eingegangen. Falls die Lokalisierung der Busnummern über EAST nicht funktioniert, kann nach der Binarisierung ein zweiter Versuch gestartet werden.

4.5 Binarisierung

Die Bilder werden vor der Texterkennung binarisiert. Das Ziel hierbei ist die Trennung von Informationen in Form der Zeichen und dem Hintergrund. In [2] ist dieser Schritt über ein einfaches Schwellwertverfahren realisiert, wobei dies häufig nicht die gewünschten Ergebnisse liefert. Deshalb werden in Abschnitt 5.4 verschiedene Ansätze zur Binarisierung untersucht.

4.6 Texterkennung

Die Texterkennung ist in [2] nicht ausreichend ausgereift. Der Ansatz verfolgt die Texterkennung mithilfe eines k-Nearest-Neighbour Klassifikators, jedoch werden andere Möglichkeiten der Texterkennung nicht weiter untersucht. Um eine möglichst genaue Bestimmung von Fahrtziel und Linie zu erhalten, sollen in Abschnitt 5.1 verschiedene Ansätze zur Klassifizierung untersucht, bewertet und der vielversprechendste für die Texterkennung eingesetzt werden.

4.7 Textverifikation

Der Hamburger Verkehrsverbund bietet die Möglichkeit, die Fahrplandaten für jeden Monat einzusehen [15]. Aus diesen können alle Endhaltestellen und die dazugehörigen Nummern ausgelesen werden, sodass eine Datenbank angelegt werden kann. Nach der Texterkennung kann so das Ergebnis verbessert, validiert oder für eine weitere Verarbeitung gespeichert werden. Falls kein passender Eintrag gefunden wird, kann so eine erneute Buserkennung stattfinden.

5 Entwicklung der Linienbuserkennung

In diesem Kapitel wird die Entwicklung der Fahrtrichtungserkennung beschrieben. Dabei werden für die einzelnen Aufgaben verschiedene Ansätze verglichen und der jeweils beste Ausgewählt.

5.1 Auswahl eines Verfahrens für die Texterkennung

Um das beste Verfahren zur Klassifikation von Text in Bilder auszuwählen, werden im Folgenden die klassische Bildverarbeitung, neuronale Netze und fertige Texterkennungssoftware miteinander verglichen. Dabei wird zunächst nur die Erkennung von Zahlen betrachtet.

5.1.1 Trainingsdatensatz

Für das Training und den Vergleich der verschiedenen Verfahren wird ein Trainingsdatensatz benötigt. Das präzise Ausschneiden und Bearbeiten der 26 unterschiedlichen Buchstaben in Klein- und Großschrift sowie den 10 Ziffern ist sehr zeitaufwendig, weshalb zunächst nur ein Trainingsdatensatz für die Ziffern synthetisch generiert wird. Da es sich bei den zu erkennenden Texten um sehr schlichte Schriftarten handelt, welche auch in verschiedenen Texteditoren genutzt werden können, ist die Verwendung synthetischer Daten gerechtfertigt. So lässt sich mithilfe von Word ein Datensatz der 10 Ziffern erstellen. Dabei werden etwa 20 verschiedene Schriftarten verwendet und diese in mehreren Variationen dargestellt. Da die Texterkennung für Busse des HVVs ausgelegt sein soll, wird ein Teil der Testdaten virtuell gedreht, sodass eine Ansicht von links unten entsteht und so der Blickwinkel einer wartenden Person an der Haltestelle simuliert wird.
Der Datensatz soll nach dem Vorbild des MNIST-Datensatzes [22] aufgebaut sein, welcher

60.000 handgeschriebene Ziffern beinhaltet. Die Bilder mit den Ziffern müssen dementsprechend auf die Größe 28x28 skaliert werden. Mithilfe von OpenCV können die Konturen der einzelnen Ziffern ausgeschnitten und anschließend auf die gewünschte Größe gebracht werden. Anders als beim MNIST-Datensatz, bei welchem der Schwerpunkt der Ziffern mittig im Bild platziert ist, wird die längere Seite der Konturen für diesen Datensatz unter Beibehaltung des Seitenverhältnis auf 28 Pixel skaliert und die kürzere Seite anschließend beidseitig mit einem weißen Rand erweitert. Des Weiteren sind die Ziffern hier in schwarz und der Hintergrund in weiß. Insgesamt befinden sich 1450 Bilder im Datensatz, sodass jede Ziffer 145-fach im Datensatz vertreten ist. Abbildung 5.1 zeigt einige Beispieldaten beider Datensätze.

(a) (b)

Abbildung 5.1: Beispieldaten aus dem MNIST Datensatz (a) und Auswahl verschiedener Bilder einer Ziffer aus dem Trainingsdatensatz (b)

5.1.2 k-Nearest-Neighbour Klassifikator

Der erste Ansatz zur Klassifizierung ist über die klassische Bildverarbeitung mittels eines k-Nearest-Neighbour Klassifikators, der auch im Ansatz zur Texterkennung in [2] verwendet wird.
Mit OpenCV kann dieser Klassifikator erstellt und trainiert werden. Anschließend können unbekannte Objekte den Klassen zugewiesen werden. Für die Erkennung der Ziffern des Trainingsdatensatzes wird dieser in 250 Validierungsdaten und 1200 Trainingsdaten eingeteilt. Als Merkmale werden der Umfang der Konturen, die Dichte und die Exzentrizität ausgewählt. Die Dichte

$$D = \frac{A_{cnt}}{A_{hull}} \tag{5.1}$$

berechnet sich aus der Fläche des Segments A_{cnt} im Verhältnis zur Fläche der konvexen Hülle A_{hull} der Kontur.

Abbildung 5.2: Beispiel einer konvexen Hülle (roter Rahmen)

Die Exzentrizität ist ein Maß der Relation von Länge und Breite eines Objektes und berechnet sich aus den zentralen Momenten des Objektes:

$$\epsilon = \frac{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}{(\mu_{20} + \mu_{02})^2} \tag{5.2}$$

Die zentralen Momente werden dabei über

$$\mu_{p,q} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (g(x,y) \cdot (x - x_s)^p \cdot (y - y_s)^q) \tag{5.3}$$

berechnet, wobei M der Breite und N der Höhe des Bildes entsprechen und $g(x,y)$ der Kontur, bei welcher die Pixel innerhalb dieser 1 und außerhalb 0 sind. (x_s, y_s) ist der Schwerpunkt der Kontur auf dem Bild und wird für Breite und Höhe einzeln

$$(x/y)_s = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} g(x,y) \cdot (x/y)}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} g(x,y)} \tag{5.4}$$

berechnet. Der Wertebereich der Exzentrizität liegt dabei zwischen 0 und 1, wobei ein perfekter Kreis 0 und eine unendlich dünne Gerade 1 entspricht.

Vor dem Training werden die Werte des Merkmalsvektors normiert, sodass diese mit einer Standardabweichung von 1 um den Wert 0 streuen. Mit der hierfür berechneten Standardabweichung und dem ursprünglichen Mittelwert werden auch die Merkmale der zu klassifizierenden Ziffern normiert.

Der Merkmalsraum für die gewählten Parameter zeigt nur wenige Cluster, wie in Abbildung 5.3 zu sehen.

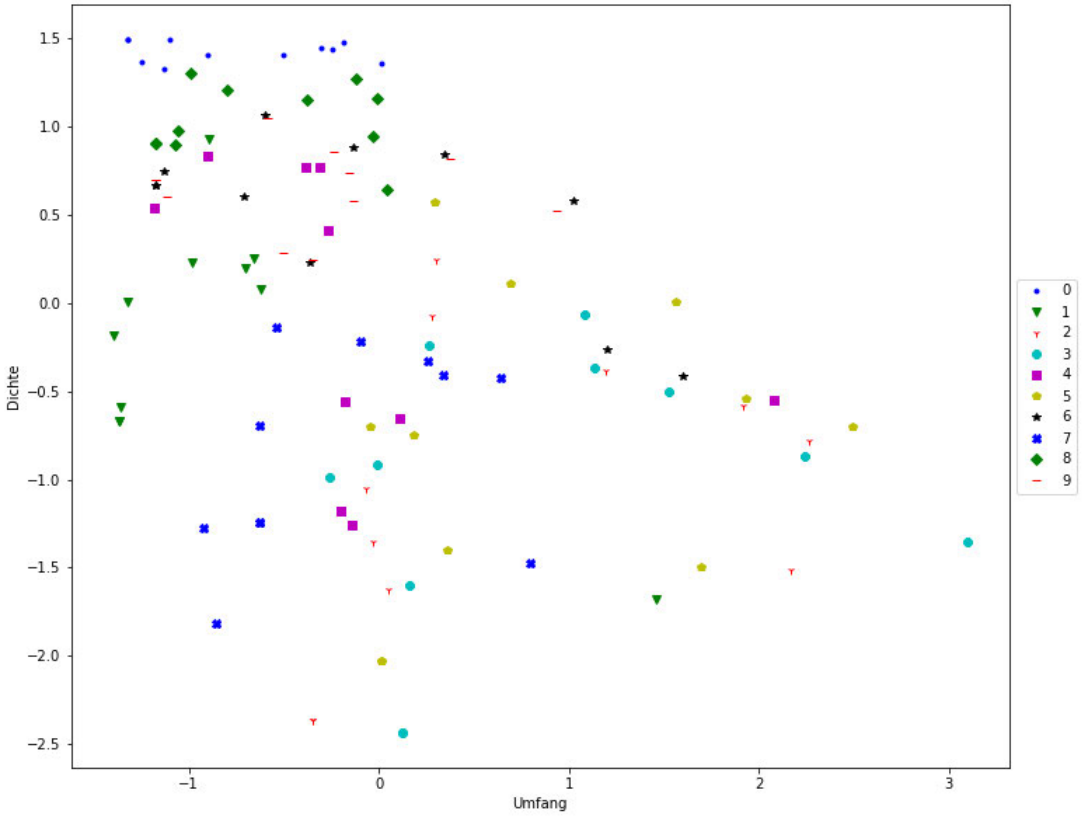


Abbildung 5.3: Merkmalsraum von Dichte und Umfang der Ziffern im Trainingsdatensatz

Im Merkmalsraum von Dichte und Umfang sind nur wenige Ziffern separierbar. So sind zum Beispiel für 0 oder 1 Cluster erkennbar, während die 3 und 5 sehr zerstreut und nicht überlappungsfrei sind.
Für $k = 20$ wird die Genauigkeit des k-Nearest-Neighbour Klassifikators bei den ausgewählten Merkmalen für den synthetischen Datensatz maximal. Es kann eine Genauigkeit von 52.0% erreicht werden.
Die Verteilung der Vorhersagen ist in Form einer Wahrheitsmatrix in Abbildung 5.4 zu sehen.

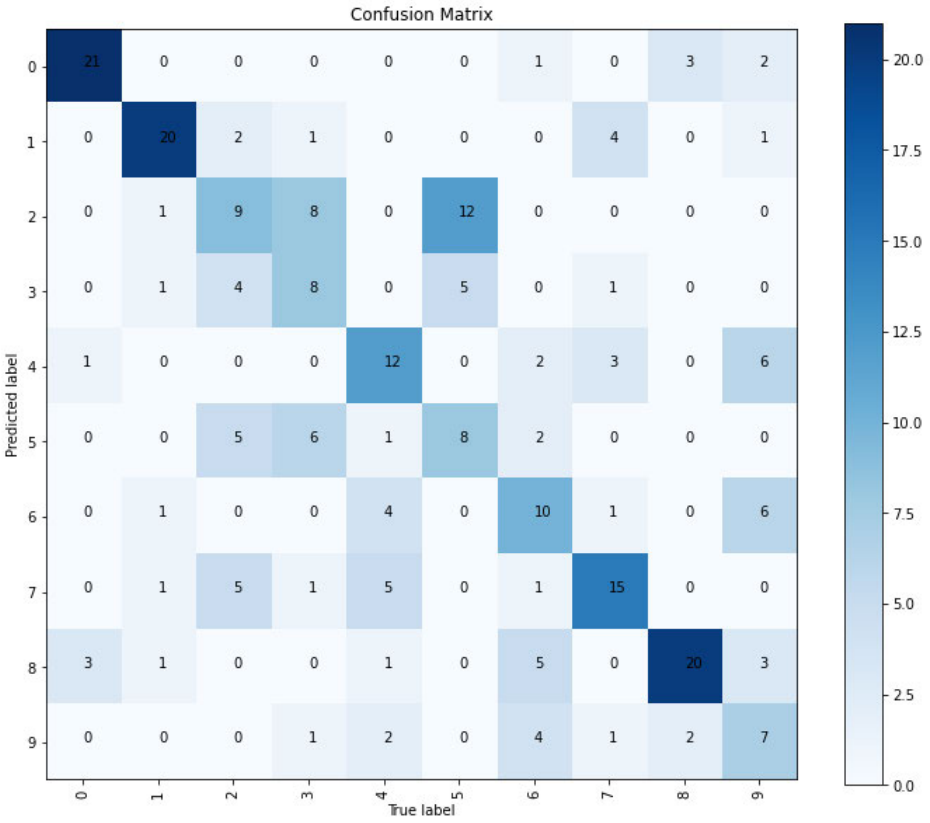


Abbildung 5.4: Wahrheitsmatrix des k-Nearest-Neighbour Klassifikators für die Testdaten

Während die 0, 1 und 8 relativ zuverlässig erkannt werden, sind die Vorhersagen für 3, 5 und 9 tendenziell als mangelhaft zu bewerten. Die erreichte Genauigkeit ist noch nicht ausreichend für eine zuverlässige Texterkennung. Zwar kann durch weitere Merkmale, wie der Anzahl der eingeschlossenen Flächen, die Genauigkeit verbessert werden, jedoch müssen für eine Texterkennung auch alle Buchstaben erkannt werden, wodurch sich die Komplexität erhöht. Der Ansatz über klassische Bildverarbeitung ist somit nicht optimal.

5.1.3 Künstliches neuronales Netzwerk

Eine weitere mögliche Realisierung der Texterkennung stellen die neuronalen Netze dar. Anders als bei der klassischen Bildverarbeitung müssen hier keine Merkmale extrahiert werden, da die neuronalen Netze diesen Schritt übernehmen. Wichtig ist deshalb die Auswahl des richtigen Netzes, das eine ausreichende Komplexität für die entsprechende Aufgabe bietet. Da die Bildgröße lediglich 28x28 entspricht und die zu erkennenden Objekte einfache Ziffern sind, ist keine hohe Komplexität des neuronalen Netzes erforderlich.

Voll verkettetes neuronales Netz

Einen erster Ansatz liefert ein vollkommen verkettetes Netz. Dabei ist die Eingangsschicht ein ”flatten layer”, welches aus dem zweidimensionalen Bild einen Vektor aus 784 Neuronen erzeugt. Als nächstes Folgen zwei ”dense layer” mit 256 und 32 Neuronen, welche die Sigmoid-Aktivierungsfunktion verwenden. Als Ausgabeschicht wird ebenfalls ein ”dense layer” verwendet, welches 10 Neuronen besitzt, die den einzelnen Klassen beziehungsweise Ziffern entsprechen. Auch diese Schicht verwendet die Sigmoid-Aktivierung. Als Kostenfunktion wird der Mean Squared Error eingesetzt. Die Struktur des Netzes ist in Abbildung 5.6 zu sehen.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input (Flatten)              (None, 784)               0
_________________________________________________________________
Hidden (Dense)               (None, 256)               200960
_________________________________________________________________
Hidden2 (Dense)              (None, 32)                8224
_________________________________________________________________
Output (Dense)               (None, 10)                330
=================================================================
Total params: 209,514
Trainable params: 209,514
Non-trainable params: 0
```

Abbildung 5.5: Aufbau voll verkettetes neuronales Netz für die Klassifizierung von Ziffern

Bei einer Batchgröße von 20 und einer optimal angepassten Lernrate von $lr = 2$ wird das Training für 35 Epochen durchgeführt (siehe Abbildung 5.6).

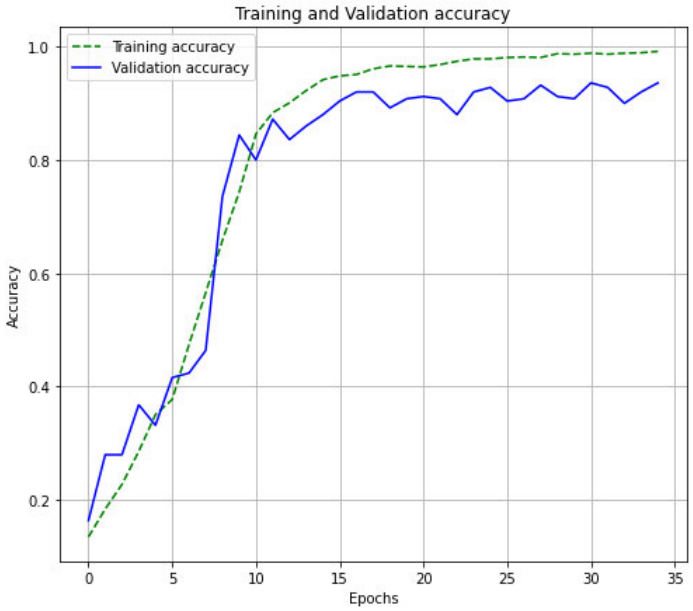


Abbildung 5.6: Training eines voll verketteten neuronalen Netzes mit Sigmoid-Aktivierungsfunktionen über 35 Epochen

Das gewählte Netz kann eine maximale Genauigkeit auf den Validierungsdaten von $acc_{max} = 93,60\%$ in Epoche 31 aufweisen. Allerdings zeigt sich schon der Effekt des Übertrainings, bei dem die Genauigkeit auf den Trainingsdaten weiter ansteigt, während sie auf den Validierungsdaten stagniert oder sogar fällt. Dies ist problematisch, da das Netz anfängt, die Trainingsdaten "auswendig" zu lernen.

Eine mögliche Lösung ist die Verwendung der ReLU-Aktivierungsfunktion in den versteckten Schichten und der Softmax-Aktivierung in der Ausgabeschicht. Zusätzlich kann die Cross-Entropy als Kostenfunktion eingesetzt werden. Durch diese Einstellung wird eine wesentlich kleinere Lernrate von $lr = 0.02$ benötigt.

Abbildung 5.7: Training eines voll verketteten neuronalen Netzes mit ReLU- und Softmax-Aktivierungsfunktionen über 35 Epochen

Durch die Anpassungen kann eine maximale Genauigkeit auf den Validierungsdaten von $acc_{max} = 95,20\%$ in Epoche 14 erreicht werden (vergleiche Abbildung 5.7). Dies bedeutet eine Verbesserung um $\Delta acc_{max} = 1,6\%$ gegenüber dem Netz mit Sigmoid-Aktivierungsfunktionen. Es zeigt sich jedoch, dass ab der Epoche 15 ein Übertraining einsetzt.

Faltendes neuronales Netz

Als weitere Möglichkeit kann ein faltendes neuronales Netz für die Klassifizierung eingesetzt werden. Für den MNIST-Datensatz kann unter optimalen Netzeinstellungen eine Genauigkeit von $acc_{max} = 99,47\%$ auf den Validierungsdaten erreicht werden [19].
Das aus [19] für den Datensatz zur Buserkennung angepasste Netz startet über zwei faltende Schichten mit ReLU-Aktivierung und 32 beziehungsweise 64 Filterkernen der Größe 3x3. Gefolgt werden die faltenden Schichten jeweils durch ein "pooling layer", welches den maximalen Wert aus einem 2x2 Filterkern auswählt. Über ein "flatten layer" wird anschließend auf ein voll verkettetes Netz umgeschaltet. Nach einem "dense layer" mit 128 Neuronen und ReLU-Aktivierungsfunktion wird über ein "dropout layer" mit

einer Rate von $0,5$ auf die Ausgangsschicht mit 10 Neuronen und Softmax-Aktivierung geschaltet. Der Aufbau des Netzes ist in Abbildung 5.8 zu sehen.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1600)              0
_________________________________________________________________
dense (Dense)                (None, 128)               204928
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 225,034
Trainable params: 225,034
Non-trainable params: 0
```

Abbildung 5.8: Aufbau des faltenden neuronalen Netzes für die Klassifizierung von Ziffern

Mit einer Lernrate von $lr = 0.01$, einem Momentum von $\beta = 0.9$ und der Cross-Entropy Kostenfunktion wird das Netz über 20 Epochen trainiert.

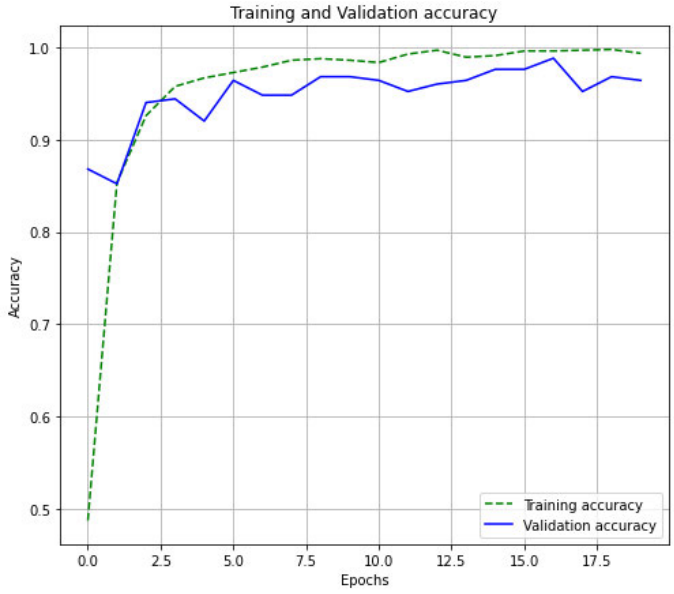


Abbildung 5.9: Training des faltenden neuronalen Netzes über 20 Epochen

Mit dem faltenden neuronalen Netz kann die Genauigkeit auf den Validierungsdaten erneut verbessert werden. In Epoche 17 ist diese mit $acc_{max} = 98,80\%$ maximal (siehe Abbildung 5.9) und somit um $\Delta acc_{max} = 3,6\%$ höher als bei der besten Konfiguration des voll verketteten Netzes. Zudem werden mit 225.034 Parametern nur etwas mehr Gewichte und Bias verwendet als beim voll verketteten Netz, welches 209.514 Parameter besitzt. Die Genauigkeit ist geringfügig unterhalb der für den MNIST-Datensatz erreichbaren $acc_{max} = 99,47\%$, dies liegt jedoch zum Teil an dem sehr viel kleineren Datensatz, der zur Verfügung steht.
Da das Training auf synthetischen Daten basiert, muss das Modell auch für reale Daten verifiziert werden. Hierfür wird ein kleiner Testdatensatz aus Ausschnitten der 10 Ziffern von Busanzeigen erstellt. Diese werden über eine geeignete Vorverarbeitung in die Form der Trainingsdaten gebracht und anschließend über eine Vorhersage durch das trainierte Netz klassifiziert. Alle Ziffern werden hierbei mit einer Wahrscheinlichkeit von über $99,1\%$ richtig erkannt. Somit ist das trainierte Modell hinreichend genau, um zuverlässige Vorhersagen für die Linienbuserkennung zu treffen.

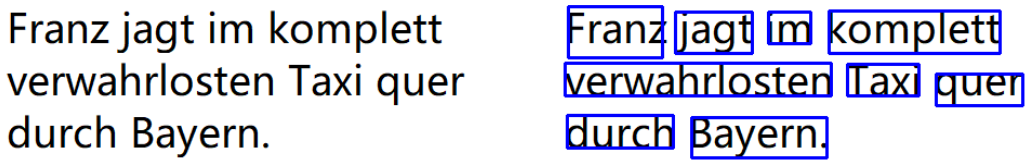
Ein Problem der Erkennung von Text durch ein neuronales Netz ist jedoch, dass die Buchstaben und Ziffern perfekt ausgeschnitten werden müssen. Sobald mehrere Buchstaben zusammenhängen, kann keine zuverlässige Klassifizierung vorgenommen werden und Leerzeichen können nicht entdeckt werden. Des Weiteren sind die genauen Positionen der einzelnen Buchstaben auf dem Bild nicht bekannt, wodurch eine zusätzliche Suche der Buchstaben erforderlich wird. Dies kann zum Beispiel durch die Verwendung von YOLO umgangen werden, jedoch wird hierfür erneut ein aufwändiges Training mit sehr hohen Anzahl an realen Bildern der Zeichen nötig.

5.1.4 Tesseract

Den dritten Ansatz zur Texterkennung stellt eine fertige Texterkennungssoftware dar, welche die Lokalisation und Klassifikation der Zeichen übernimmt.
Es gibt einige Softwares zur Texterkennung, wie *Google Cloud Vision API* [13], *ABBYY FineReader* [1] und *Tesseract* [40]. Zwar sind die kostenpflichtigen Varianten leistungsfähiger [14], jedoch wird im Folgenden die kostenfreie Software Tesseract verwendet, da keine spezielle Lizenz nötig ist. Falls es notwendig sein sollte, kann so nachträglich auch eine andere Software eingebaut werden, welche die Genauigkeit der Erkennung nur verbessern würde.

Tesseract ist eine in $C/C++$ geschriebene Texterkennungssoftware, die ursprünglich von Hewlett-Packard entwickelt und 2005 von Google übernommen wurde. Seitdem befindet sie sich in einer kontinuierlichen Weiterentwicklung, wobei der aktuelle Stand die Version 4.1.1 ist. Tesseract unterstützt über 100 verschiedene Sprachen und kann aufgrund der Apache-Lizenzierung von jedem frei verwendet werden [40].
Statt über den klassischen Ansatz mit Mustervergleichen, arbeitet Tesseract seit Version 4.00 über ein neuronales Netz, welches mit 400.000 Textzeilen in 4.500 verschiedenen Schriftarten trainiert wurde. Um die Texterkennung auf die eigenen Daten anzupassenkann das Netz auch zusätzlich trainiert werden. Dieser Schritt wird für die Linienbuserkennung zunächst übersprungen, da in den Anzeigen der Busse keine ungewöhnlichen Schriftarten verwendet werden. Zudem kann deshalb der aufwendige Schritt der Datenerhebung und des Trainings vernachlässigt werden und so zunächst nur eine Abschätzung der erreichbaren Genauigkeit mit Tesseract aufgestellt werden. Tesseract bietet auch die Möglichkeit eines Lexikons, welches nach Bedarf angepasst werden kann, sodass beispielsweise auch Eigennamen in diesem vorkommen können. Für die verschiedenen Haltestellen wäre dies ein geeigneter Schritt, jedoch nutzt Tesseract das Lexikon nur, um nachträglich die Sicherheit des vorhergesagten Wortes zu überprüfen [11].
Die Texterkennung kann in 14 verschiedenen Modi eingesetzt werden. Dabei kann das Bild zum Beispiel als ein einzelner Buchstabe, als Wort oder als Textzeile behandelt werden. Es besteht jedoch auch die Möglichkeit, das Bild automatisch für Texte zu scannen und diese auslesen zu lassen.
Tesseract ist dafür ausgelegt, eingescannte Dokumente zu erkennen. Deshalb erfolgt standardmäßig eine Erkennung von schwarzen Zeichen auf weißem Hintergrund. Bildrauschen und nicht horizontal ausgerichtete Texte beeinträchtigen die Qualität der Erkennung. Für Bilder von beispielsweise Straßenschildern ist Tesseract nicht ausgelegt. Hierfür müssen die Bilder geeignet vorverarbeitet werden, sodass im besten Fall nur noch der Text als schwarze Zeichen auf weißem Hintergrund stehen bleibt.
Tesseract funktioniert bei hoher Auflösung besser, sodass die zu erkennenden Bilder eine gewisse Größe und Qualität besitzen sollten. Die optimale Größe der Buchstaben sollte im besten Fall zwischen 30 und 33 Pixel Höhe liegen, da Tesseract hier die höchste Genauigkeit hat [24]. Bei einer kleineren Größe ist der Abfall der Genauigkeit relativ stark, während sie bei einer höheren Auflösung nur leicht verringert wird. Dies gilt jedoch für Texte aus digitalen Dokumenten, welche eine optimale Qualität bieten. Für Texte aus Bildern können deshalb ganz andere Werte gelten. Empfehlenswert ist daher eine möglichst hohe Auflösung der Buchstaben.

Für den gesamten Trainingsdatensatz aus Unterabschnitt 5.1.1 kann bei der Klassifizierung mit Tesseract eine Genauigkeit von 79% erreicht werden. Dies ist zwar wesentlich geringer als beim künstlichen neuronalen Netz, jedoch sind etwa 20% der Testdaten rotiert, was die Genauigkeit der Klassifikation von Tesseract stark verringert. Wenn eine Rotation der Zahlen vorgenommen werden kann, lässt sich die Genauigkeit wieder verbessern. Ebenso könnte ein Training von Tesseract auf den zu erwartenden Zeichen hier zu Verbesserungen führen.
Besonders bei Texten lässt sich der Vorteil von Tesseract gegenüber den neuronalen Netzen verdeutlichen, da für diese eine genaue Lokalisation der einzelnen Buchstaben nötig ist. Durch die Punkte auf manchen Buchstaben, wie beim 'i', erschwert sich dies weiter. Auch sobald mehrere Zeichen zusammenhängen, ist eine Klassifizierung dieser nicht mehr möglich. Tesseract übernimmt diese Schritte für den Nutzer und liefert so brauchbare Ergebnisse. Aus der Abbildung 5.10 kann Tesseract mit einer Sicherheit von über 90% für jedes einzelne Wort den richtigen Text auslesen.



(a) (b)

Abbildung 5.10: Test der Klassifikation von Texten aus Bildern: (a) Originalbild; (b) von Tesseract detektierte Wörter

Aufgrund der hohen Genauigkeit der Klassifikation und der Möglichkeit, Texte automatisch in Bildern zu finden, wird im weiteren Verlauf Tesseract zur Texterkennung eingesetzt.

5.2 Vorverarbeitung

5.2.1 Abschätzung der Bildqualität

Die Verarbeitung für die Texterkennung ist abhängig von der Qualität der Bilder, denn bei einer zu geringen Auflösung ist der Text nicht mehr lesbar, während bei einer sehr hohen Auflösung wie in Abbildung 5.11 die einzelnen LEDs der Anzeigen sichtbar werden und so eine Weiterverarbeitung erschwert wird.

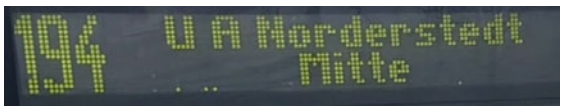


Abbildung 5.11: Bild einer Fahrtrichtungsanzeige bestehend aus einzelnen LEDs in hoher Auflösung und Schärfe

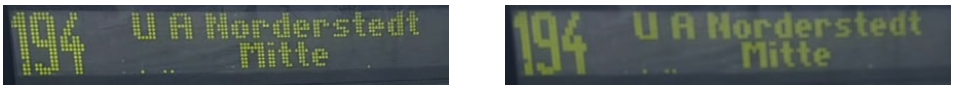
Für die Verarbeitung der Bilder ist eine Abschätzung ihrer Qualität sehr nützlich, damit bei einer zu geringen Auflösung die Texterkennung übersprungen werden kann und bei einer hohen Auflösung weitere Verarbeitungsschritte eingeleitet werden.
Da die Zielhardware noch nicht zur Verfügung steht, kann dieser Schritt nicht sinnvoll ausgearbeitet werden, denn die Qualität der Aufnahmen ist nicht nur von der Auflösung der Kamera abhängig, sondern wird auch durch andere Faktoren wie dem Objektiv beeinflusst.

Über die Größe des Ausschnittes eines detektierten Busses lässt sich bei bekannter Kamera ermitteln, wie weit dieser entfernt ist. Die Verarbeitung kann so in verschiedene Fälle eingeteilt werden. Für die Fallunterscheidung eignet sich die Breite der Ausschnitte in Pixel am besten, da die Breite der verschiedenen Fahrzeuge relativ ähnlich ist, während die Höhe zum Beispiel durch ein zweites Deck stärker variieren kann.
Der erste Schwellwert $w < t_{min}$ ist für zu kleine Ausschnitte vorgesehen, bei dem zwar ein Bus erkannt wird, aber die verhältnismäßig kleine Schrift auf der Anzeige nicht auswertbar ist. Für diese Fälle muss keine Textextrahierung vorgenommen werden. So kann auch ein gewisser Teil der falsch positiven Busdetektionen herausgefiltert werden, da diese häufig einen kleinen Ausschnitt im Bild beschreiben.
Für Ausschnitte mit einer Breite von $t_{min} \leq w < t_{low}$ ist die Bildqualität gerade ausreichend für die Texterkennung, sodass in diesem Fall das Auslesen der Anzeige probiert

werden kann. Um keine Informationen der kleinen Schrift zu verlieren, wird keine weitere Filterung vorgenommen.

Abbildung 5.12: Bild einer lesbaren Fahrtrichtungsanzeige in sehr geringer Qualität

Optimal sind die Busse mit einer Breite von $t_{low} \leq w < t_{high}$ Pixeln, weil der Text in ausreichender Größe zu sehen ist, während die einzelnen LEDs nicht sichtbar sind. Mithilfe eines einfachen Blur-Filter, welcher für jeden Punkt eines Bildes den Mittelwert aus einem Fenster der Größe m x n um diesen Punkt berechnet, kann leichtes Bildrauschen eliminiert werden. Ein 3 x 3 Filterkern liefert zufriedenstellende Ergebnisse.
Im letzten Fall $t_{high} \leq w$ ist der Bus in zu hoher Auflösung, sodass die einzelnen LEDs der Anzeige sichtbar werden. In diesem Fall kann das Bild einfach auf die Breite $w = t_{high}$ skaliert und mit dem Blur-Filter die Lücken zwischen den LEDs geschlossen werden (vergleiche Abbildung 5.13).



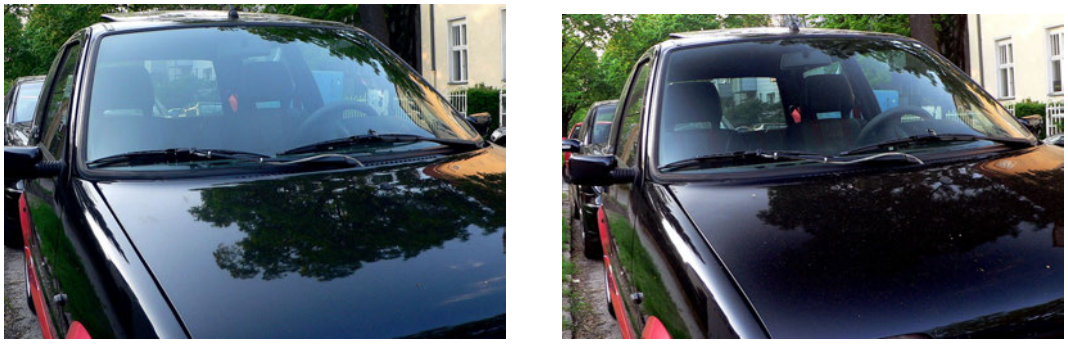
(a) (b)

Abbildung 5.13: Beispiel der Skalierung und Weichzeichnung eines zu hoch aufgelösten Bildes (a) in ein für die Texterkennung optimales Bild (b)

Für die Implementierung der Buserkennung müssen die Schwellwerte auf die Eigenschaften der Kamera angepasst werden.

5.2.2 Entspiegelung

Eine Herausforderung bei der Texterkennung sind die Spiegelungen im Glas der Anzeigen. Als Quelle hierfür ist zum Beispiel die direkte Sonneneinstrahlung verantwortlich, welche allerdings häufig so stark ist, dass ein Lesen der Anzeige unmöglich wird. Aber auch der Himmel sorgt durch die hohe Helligkeit für starke Reflexionen, welche die Lesbarkeit der Anzeigen hingegen nur reduzieren. Dies gilt vor allem bei klarem Himmel, jedoch auch bei bedeckter Wetterlage. Andere Reflexionen sind tagsüber eher vernachlässigbar. Eine Filterung der Reflexionen kann die Genauigkeit der Fahrtrichtungserkennung anheben.
Eine Möglichkeit stellt dabei die Verwendung eines Polarisationsfilters am Objektiv der Kamera dar, welcher in der richtigen Ausrichtung die durch die Reflexion polarisierten Lichtstrahlen nicht durchlässt (siehe Abbildung 5.14).



(a) (b)

Abbildung 5.14: Windschutzscheibe eines Autos fotografiert (a) ohne Polarisationsfilter [44] und (b) mit optimal ausgerichtetem Polarisationsfilter [45]

Dieser Ansatz ist jedoch außerhalb des Rahmens dieser Ausarbeitung. Deshalb ist die nachträgliche Filterung in der Software eine weitere Möglichkeit. Hier können die Eigenschaften der Fahrtzielanzeigen genutzt werden, da diese für Fahrzeuge des Hamburger Verkehrsverbunds stets einen schwarzen Hintergrund und weiß, orange oder grün leuchtende Schrift besitzen. Der blaue Kanal bietet somit lediglich bei der weißen Schrift Informationen, welche jedoch auch in den beiden anderen Kanälen enthalten sind. Somit kann dieser für die Textverarbeitung vernachlässigt werden.
Bei einer Betrachtung der BGR-Werte von Bildern des Himmels bei klaren Wetterverhältnissen fällt auf, dass Blau der dominierende Farbanteil ist und Rot den geringsten Mittelwert besitzt. In einem Ausschnitt von Videos, die über einen Zeitraum von einem Jahr den Himmel über San Francisco zeigen [28], können Mittelwerte für die BGR-Farben

bestimmt werden. Blau hat mit 183, 23 den höchsten Wert, während Grün mit 144, 66 in der Mitte liegt und Rot mit 114, 67 den geringsten Anteil hat. Dies liegt an der Rayleigh-Streuung [32]. Diese beschreibt das Verhalten der Streuung von elektromagnetischen Wellen an Teilchen mit geringerem Durchmesser als der Wellenlänge. Der Effekt tritt an der Atmosphäre der Erde auf und sorgt für die Streuung des Sonnenlichts. Wenn die Sonne senkrecht am Himmel steht, ist der Weg für das Licht durch die Atmosphäre am kürzesten, sodass vor allem Wellen mit geringerer Wellenlänge gestreut werden. Dabei setzen sich die gestreuten Wellen zu bläulichem Licht zusammen. Da Rot mit $\lambda \approx 620 - 780\ nm$ [20] die höchste Wellenlänge der drei BGR-Farben besitzt, ist der Rot-Anteil im Himmel am geringsten (vergleiche Abbildung 5.15).

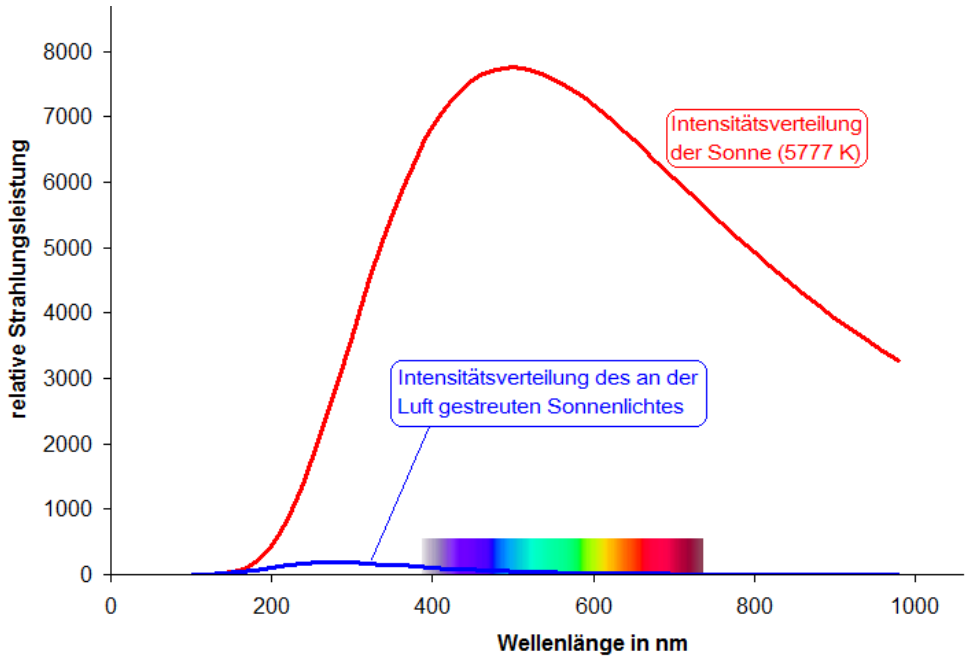


Abbildung 5.15: Intensität des direkten und des an der Luft gestreuten Sonnenlichtes [46]

Hieraus ergibt sich, dass der rote Kanal weniger anfällig für gewisse Reflexionen ist.
Die unterschiedlichen Intensitäten der Farbkanäle können für einen bedeckten Himmel hingegen nicht deutlich beobachtet werden. Dadurch sind der grüne und rote Kanal gleichermaßen verantwortlich für diese Spiegelungen.
Zwar sind einige LED-Anzeigen der Busse für das menschliche Auge grün, jedoch ist hier trotzdem ein hoher roter Anteil vorhanden, sodass die Informationen nicht verloren gehen, wenn der grüne Kanal wegfällt (siehe Abbildung 5.16).

(a) (b)

Abbildung 5.16: Grüne Anzeige der Busnummer: (a) Originalbild; (b) roter Farbkanal als Grauwert

Da bei klarem Himmel durch die erhöhte Helligkeit die stärksten Reflexionen entstehen, verbessert das Vernachlässigen des grünen Kanals die Lesbarkeit in den kritischsten Fällen. Deshalb funktioniert die Verarbeitung über lediglich den roten Kanal am besten. Eine Ausnahme hierfür kann die Zeit während eines Sonnenuntergangs sein, da währenddessen auch die längeren Wellen des roten Lichtes durch den verlängerten Weg durch die Atmosphäre gestreut werden und so der Himmel rötlich erscheint. Dieser Fall wird in dieser Ausarbeitung jedoch zunächst vernachlässigt, da er vergleichsweise selten auftreten kann. Denkbar wäre hier eine Verwendung einer speziell angepassten Verarbeitung im kritischen Zeitraum.

Um die leuchtenden LEDs vom Hintergrund besser abzuheben, kann eine Multiplikation der Rot-Werte der einzelnen Pixel mit sich selber vorgenommen werden. Durch eine anschließende Division von 255 kann ein Überlauf verhindert werden. So bleiben helle Pixel, aus denen die Texte bestehen, relativ gleich, während dunklere Pixel nahezu 0 werden.

In Abbildung 5.17 wird die Entspiegelung einer Anzeigetafel mit orangefarbener Schrift dargestellt. In diesem Fall ist die Reflexion des blauen Himmels problematisch für die Lesbarkeit. Durch die Verwendung von lediglich des roten Kanals des Bildes kann die Schrift der Anzeige stärker hervorgehoben werden. Auch die Spiegelung des Himmels ist weniger stark, allerdings verschwimmen die Kanten der Schrift etwas durch das leicht diffuse Licht der LEDs. Durch die skalierte Multiplikation des roten Kanals mit sich selbst,wird der Hintergrund noch dunkler, während die Schrift ihre Helligkeit beibehält. So wird auch das diffuse Licht, welches für die Glättung der Kanten verantwortlich ist, ebenfalls minimiert.

(a) (b)

(c) (d)

Abbildung 5.17: Filterung der Reflexionen über Auswertung der Farbkanäle bei einer roten Anzeige und klarem Himmel: (a) Originalausschnitt; (b) Bild in Graustufen; (c) roter Farbkanal des Bildes als Grauwert; (d) angepasster roter Farbkanal als Grauwert

Abbildung 5.18 zeigt die Verarbeitung für eine Anzeige mit grüner Schrift bei bedecktem Himmel. Hier bringt die Verwendung von lediglich des roten Kanals keine klare Verbesserung gegenüber dem einfachen schwarz-weiß Bild, allerdings ist auch keine Verschlechterung erkennbar. Erst die Weiterverarbeitung des roten Kanals hebt die Schrift deutlich vom Hintergrund ab.

(a) (b) (c) (d)

Abbildung 5.18: Filterung der Reflexionen über Auswertung der Farbkanäle bei einer grünen Anzeige und bedecktem Himmel: (a) originaler Ausschnitt; (b) Bild in Graustufen; (c) roter Farbkanal des Bildes als Grauwert; (d) angepasster roter Farbkanal als Grauwert

Die Entspiegelung durch ausschließlich die Verwendung des roten Kanals ist sehr effektiv, sofern die Anzeige aus weißer oder orangefarbener Schrift besteht und ein klarer Himmel die Reflexionen verursacht. Im unvorteilhaftesten Fall ist der Text grün und die Spiegelungen durch einen bedeckten Himmel verursacht. Durch die Anpassung der Farbwerte kann jedoch auch hier die Lesbarkeit verbessert werden.

5.3 Textlokalisierung

Tesseract ist für die Erkennung eingescannter Dokumente mit schwarzen Ziffern auf weißem Hintergrund ausgelegt und liefert somit bei der Anwendung auf Bildern realer Szenen unzureichende Ergebnisse. Deshalb muss der Text aus dem Bild der Busfront aufbereitet werden. Hierfür ist eine Lokalisierung des Textes sinnvoll.
Da bei der Erkennung von einigen Zeichenkombinationen Probleme auftreten, sodass zum Beispiel 'ZOB' als '208' oder '50' als 'SO' interpretiert werden, ist bei der Texterkennung mittels Tesseract eine getrennte Erkennung von Busnummer und Fahrtziel vorteilhaft. Zwar können auch mehrere Texte unterschiedlicher Größe in einem Schritt klassifiziert werden, allerdings lässt sich Tesseract durch die Teilung in Text und Nummer besser auf die spezifische Anwendung vorbereiten. Über eine Whitelist kann eingestellt werden, welche Zeichen erkannt werden dürfen, sodass die oben genannten Fehler bei der einzelnen Erkennung vermieden werden.

5.3.1 Geometrische Einteilung

Die einfachste Methode zur Einteilung in Text und Nummer ist die Teilung des Bildes. Dies ist möglich, da die Anzeigen der HVV-Busse immer gleich aufgebaut sind. Die Nummer befindet sich bei frontaler Ansicht immer in der linken oberen Ecke, während der Text mittig bis rechts auf gleicher Höhe ist (vergleiche Abbildung 5.19).

Abbildung 5.19: Aufbau der Anzeigen von Bussen des Hamburger Verkehrsverbunds

Die Ausschnitte für die Textfelder können allerdings nicht perfekt passend gewählt werden, da die Nummern und vor allem der Text unterschiedlich lang sind. Des Weiteren kann durch eine nicht optimal horizontal ausgerichtete Kamera oder einen leicht seitlichen Blickwinkel auf den Bus der Text auf der Anzeige einen Winkel haben. Da die Busse über einen "Haar Cascade Classifier" erkannt und ausgeschnitten werden, kann die Position des Busses zudem leicht variieren. Aus diesen Gründen müssen die Bildausschnitte großzügig gewählt werden, sodass stets gegeben ist, dass die Nummer beziehungsweise

der Text vollständig im Bild ist. Der Ausschnitt darf jedoch nicht zu groß gewählt werden, da sonst die Teilung in Liniennummer und Fahrtziel nicht funktioniert.
Die Bildausschnitte müssen eine gewisse Höhe besitzen, da manche Busse auf dem Dach Batterien oder Klimaanlagen besitzen, wodurch die Anzeige nicht am oberen Rand des Busses positioniert ist. Bei der mittigen Trennung zwischen Nummer und Text darf maximal ein Zeichen im falschen Feld liegen, da einzelne Zeichen von EAST häufig nicht erkannt werden.

Der Ausschnitt der beiden Felder erfolgt über die Höhe h und Breite w des Bildes der Busfront. Die Grenzen sind manuell auf 660 verschiedene Ausschnitte der Busdetektion angepasst. Dabei geht die obere Grenze bis zum Rand des Ausschnittes des Busses, während die untere Grenze für beide Textfelder bei $y_{max} = 0,3 \cdot h$ liegt.
Für die Nummer ist die linke Grenze bei $x_{min,num} = 0,1 \cdot w$ und die rechte Grenze bei $x_{max,num} = 0,375 \cdot w$. Beim Text ist die linke Begrenzung bei $x_{min,txt} = 0,\bar{3} \cdot w$ und die rechte bei $x_{max,txt} = 0,9 \cdot w$.
Die Einteilung lässt sich in Abbildung 5.20 beobachten.

Abbildung 5.20: Ausschnitt von Zahl (grün) und Text (rot) anhand der Geometrie des Bildes

Mit den gewählten Bildausschnitten kann die Trennung von Fahrtziel und Linie zuverlässig vorgenommen werden. Zusätzlich werden andere Texte wie Werbung und Nummernschild aus dem Bild gelöscht.

5.3.2 EAST

Um den Text aus den grob ausgeschnittenen Bereichen möglichst genau herauszufiltern, wird die EAST Textdetektion eingesetzt, da diese schnell und zuverlässig Texte in Bildern finden kann [54]. Als Eingang der Lokalisation werden die bereits vorverarbeiteten groben Ausschnitte von Busnummer und Fahrtziel verwendet.

Fahrtziel

Für die Lokalisierung der Fahrtziele werden die Bilder zunächst auf 192 Pixel Breite und 96 Pixel Höhe skaliert, wobei das Seitenverhältnis aus der vorigen groben Einteilung einigermaßen beibehalten werden kann. Zwar ist bei diesen Maßen der Text nicht mehr lesbar, jedoch kann EAST diesen trotzdem lokalisieren. Durch das Herunterskalieren werden andere Störfaktoren eliminiert, sodass die Anzeigen zuverlässig erkannt werden. Ein zusätzlicher positiver Effekt ist die Verbesserung der Laufzeit.
Der Schwellwert für die erkannten Bereiche kann sehr klein gewählt werden, da durch das vorige grobe Ausschneiden die Wahrscheinlichkeit, andere Texte im Bild zu haben, sehr gering wird. Statt die gefundenen Regionen durch Non-Maximum Suppression zu filtern, kann deshalb einfach der minimale und maximale X- und Y-Wert aus allen gefundenen Textfeldern gewählt werden. So wird in der Regel ein etwas größerer Ausschnitt gewählt, damit sichergestellt wird, dass kein Teil abgeschnitten wird (siehe Abbildung 5.21). Dies ist auch bei zweizeiligen Einträgen hilfreich, wenn ein Teil einer Zeile fehlt.

Abbildung 5.21: Texterkennung durch EAST (grüner Rahmen) auf einer (a) Fahrtrichtungsanzeige (b) ohne Weiterverarbeitung, (c) mit Non-Maximum Suppression und (d) über Maximal- und Minimalwerte

Da der Text der Anzeigen auf dem Bild schräg stehen kann und Tesseract Probleme damit hat [39], ist eine Winkelkorrektur sinnvoll. EAST liefert für jedes Textfeld den Rotationswinkel. Da die Felder in dieser Verarbeitung zu einem großen Feld zusammengeführt werden, müssen die Winkel ebenfalls zusammengelegt werden. Hier bietet sich der Median an, da so einzelne Ausreißer gänzlich ignoriert werden. Der Mittelpunkt für die Rotation wird im Bild auf den Maximalwert der Breite und Höhe gelegt.

Busnummer

Die genauere Lokalisierung der Nummer kann nach demselben Prinzip erfolgen, jedoch muss das Seitenverhältnis für die Detektion angepasst werden. Es kann beobachtet werden, dass die Detektion der Busnummern auf den Anzeigen über EAST nicht so zuverlässig funktioniert wie die der Texte. Nur bei Bildern mit einer höheren Qualität werden die Zahlen erkannt, weshalb die Ausschnitte auf 256x256 Pixel skaliert werden. Bei Bildern mit geringerer Qualität ist auch nach der Skalierung keine Erkennung durch EAST möglich.
Sofern ein Textfeld lokalisiert wird, kann wie bei der Fahrtrichtung vorgegangen werden. Anderenfalls lässt sich nur der Rotationswinkel aus der Lokalisierung des Textes übernehmen, sofern einer gefunden wurde. In diesem Fall wird mit dem vollen groben Ausschnitt der Busnummer weitergearbeitet.

5.4 Binarisierung

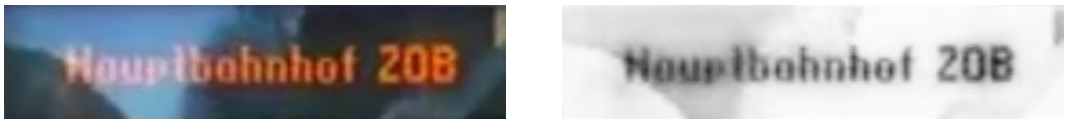
5.4.1 Grauwertbild

Die ausgeschnittenen Texte, welche durch die Entspiegelung in Grauwerten vorliegen, sind noch nicht bereit für eine Klassifizierung mittels Tesseract. Der Text auf den Bildern ist in Weiß auf schwarzem Hintergrund, während Tesseract für die Erkennung schwarzer Zeichen auf weißem Hintergrund ausgelegt ist. Ein Umkehren der Werte durch

$$g(x, y) \;=\; 255 - g(x, y) \tag{5.5}$$

reicht allerdings nicht aus, da der Hintergrund nicht einheitlich ist und der Text teilweise recht verschwommen wirken kann.

Für die vorverarbeitete Anzeige aus Abbildung 5.17 (d) ließt Tesseract nach der Invertierung durch Formel 5.5 'Wemsilbohnhot ZUB'.



(a) (b)

Abbildung 5.22: (a) Originalbild und (b) nach farblicher Vorverarbeitung invertierte Anzeige

5.4.2 Einfaches Schwellwertverfahren

Durch ein Schwellwertverfahren kann das Bild aus Graustufen in ein binäres Bild umgewandelt und im besten Fall der Text klar vom Hintergrund getrennt werden. Ein Problem bei der Verwendung eines einfachen Schwellenwertes ist die Bestimmung der optimalen Schwellwerte, welcher zudem für jedes Bild anders ist.
In Abbildung 5.23 werden zwei Anzeigen nach der Anpassung der Farbkanäle und anschließender Invertierung mit einem festen Schwellwert von 150 binarisiert. Dabei ist der Schwellwert für die erste Anzeige optimal angepasst. Das binarisierte Bild der ersten Anzeige liefert ein zufriedenstellendes Ergebnis, sodass Tesseract 'Hauptbannbot ZOB' lesen kann. Der Text im zweiten Bild ist jedoch nicht lesbar, da der gewählte Schwellwert unpassend ist.

(a) (b)

(c) (d)

Abbildung 5.23: Binarisierung zweier unterschiedlicher Anzeigen nach Vorverarbeitung mit einem Schwellwert von 150: (a) unbearbeiteter Ausschnitt der Anzeige auf den der Schwellwert angepasst ist; (b) binäres Bild der angepassten Anzeige; (c) unbearbeiteter Ausschnitt einer zweiten Anzeige; (d) binäres Bild der zweiten Anzeige

5.4.3 Otsu-Binarisierung

Das Problem einen passenden Schwellwert zu finden löst die Otsu-Binarisierung [29]. Dabei werden die Graustufen des Bildes so in zwei Klassen eingeteilt, dass die Streuung innerhalb der Klassen minimal und zwischen den beiden Klassen so groß wie möglich ist. Für die Grauwerte $0 \leq g \leq G$, mit $G = 255$ als Maximalwert kann die Wahrscheinlichkeit der Zugehörigkeit zu den Klassen K_1 und K_2, welche durch den Schwellwert t getrennt sind, durch

$$P_0(t) \;=\; \sum_{g=0}^{t} p(g) \qquad \textit{und} \qquad P_1(t) \;=\; \sum_{g=t+1}^{G} p(g) \tag{5.6}$$

berechnet werden, wobei $p(g)$ der Wahrscheinlichkeit des jeweiligen Grauwertes entspricht.
Mit den mittleren Grauwerten $\bar{g_0}$ und $\bar{g_1}$ können die Varianzen innerhalb der einzelnen Klassen

$$\sigma_0^2(t) \;=\; \sum_{g=0}^{t} (g - \bar{g_0})^2 \cdot p(g) \qquad \textit{und} \qquad \sigma_1^2(t) \;=\; \sum_{g=t+1}^{G} (g - \bar{g_1})^2 \cdot p(g) \tag{5.7}$$

berechnet werden, aus welchen sich die Gesamtvarianz innerhalb der Klassen

$$\sigma_{in}^2(t) \;=\; P_0(t) \cdot \sigma_0^2(t) + P_1(t) \cdot \sigma_1^2(t) \tag{5.8}$$

zusammensetzt. Die Varianz zwischen den Klassen kann durch

$$\sigma_{zw}^2(t) \;=\; P_0(t) \cdot (\bar{g_0} - \bar{g})^2 + P_1(t) \cdot (\bar{g_1} - \bar{g})^2 \tag{5.9}$$

berechnet werden, wobei $\bar{g}$ dem mittleren Grauwert des gesamten Bildes entspricht.
Der Schwellwert t ist für ein Bild optimal, wenn der Quotient

$$Q(t) \;=\; \frac{\sigma_{zw}^2(t)}{\sigma_{in}^2(t)} \tag{5.10}$$

maximal wird.

Angewendet auf die zwei Ausschnitte der Fahrtzielanzeigen aus Abbildung 5.23 kann

die Otsu-Binarisierung für beide einen passenden Schwellwert finden. Allerdings liegen in der Regel die durch Otsu berechneten Schwellwerte für diesen Anwendungsfall so, dass die Zeichen teilweise ineinander verschwimmen. Der Hintergrund ist ansonsten jedoch einheitlich weiß (siehe Abbildung 5.24).

(a) (b)

(c) (d)

Abbildung 5.24: Binarisierung zweier unterschiedlicher Anzeigen nach Vorverarbeitung mit Otsu-Binarisierung: (a) unbearbeiteter Ausschnitt der ersten Anzeige; (b) binäres Bild der ersten Anzeige; (c) unbearbeiteter Ausschnitt der zweiten Anzeige; (d) binäres Bild der zweiten Anzeige

Die Trennung durch einen einfachen Schwellwert ist deshalb nicht ausreichend, da zusätzlich in vielen Bildern die Helligkeit nicht gleichmäßig ist. Ein Schwellwert kann so passend für einen Teil des Bildes sein, während er für einen anderen Teil des Bildes unzureichende Ergebnisse liefert.
Für die Anzeige aus Abbildung 5.22 ist der mittlere Teil des Fahrtziels nach der Otsu-Binarisierung relativ gut lesbar, während die ersten Buchstaben zu einer unidentifizierbaren Fläche verschmelzen, da der Wert für diesen Bereich zu hoch ist (vergleiche Abbildung 5.25).

Abbildung 5.25: Otsu-Binarisierung einer Anzeige mit unterschiedlichen Lichtverhältnissen

5.4.4 Adaptives Gaußsches Schwellwertverfahren

Eine weitere Möglichkeit besteht darin, den Schwellwert für jeden Punkt des Bildes in Abhängigkeit seiner Nachbarn zu berechnen. Dies funktioniert zum Beispiel über das adaptive Gaußsche Schwellwertverfahren [31]. In OpenCV berechnet dieses den Schwellwert $T(x, y)$ durch die Kreuzkorrelation von einem Fenster zuvor festgelegter Größe k_{size} um (x, y) mit der Gaußschen Fensterfunktion. Diese ist über

$$G_i \;=\; a \cdot e^{\frac{-(i-(k_{size}-1)/2)^2}{2\cdot\sigma^2}} \tag{5.11}$$

definiert, wobei $i = 0..k_{size}$ entspricht und a so gewählt wird, dass

$$\sum_{i=0}^{k_{size}} G_i \;=\; 1 \tag{5.12}$$

gilt. Die Standardabweichung der Gaußschen Fensterfunktion berechnet sich aus:

$$\sigma = 0.3 \cdot ((k_{size} - 1) \cdot 0.5 - 1) + 0.8 \tag{5.13}$$

Ein festlegbarer Schwellwert C wird zusätzlich von $T(x, y)$ subtrahiert, sodass die Empfindlichkeit angepasst werden kann.

Mit $k_{size,txt} = 11$ und $C = 2$ kann das adaptive Gaußsche Schwellwertverfahren für die Fahrtrichtungserkennung gute Ergebnisse liefern. Da die Zahlen der Busnummer tendenziell etwas größer sind, muss hier ein größerer Bereich eingestellt werden. Für die Verarbeitung der Busnummer eignet sich $k_{size,num} = 21$. Abbildung 5.26 zeigt die Binarisierung von der Anzeige aus Abbildung 5.22 über das adaptive Gaußsche Schwellwertverfahren. Besonders die Kanten der Buchstaben werden durch dieses Verfahren klar herausgearbeitet. Zwar kann Tesseract aus diesem Bild schon 'Haupt hnhof.ZOB.' lesen, allerdings enthält das Bild ein noch zu hohes Hintergrundrauschen, was die Qualität der Klassifizierung verschlechtert.

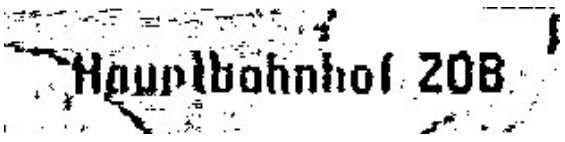


Abbildung 5.26: Binarisierung über das adaptive Gaußsche Schwellwertverfahren

5.4.5 Kombination von Gauß und Otsu

Um die Anzeigen mit möglichst wenig Hintergrundrauschen und gleichzeitig einer hohen Qualität der Buchstaben zu binarisieren, können das adaptive Gaußsche Schwellwertverfahren und die Otsu-Binarisierung kombiniert werden. Hierfür wird eine Disjunktion der Bilder aus beiden Verfahren durchgeführt. Dabei bleiben nur die Pixel dunkel, welche bei beiden Algorithmen einer 0 entsprechen. So wird das Hintergrundrauschen des Gaußschen Schwellwertverfahren eliminiert und gleichzeitig können die klaren Kanten der Buchstaben beibehalten werden.
Für die Anzeige aus Abbildung 5.22 kann über dieses Verfahren eine sehr gute Trennung von Text und Hintergrund realisiert werden. In Abbildung 5.27 ist nur sehr wenig Rauschen noch sichtbar, während der Text gut lesbar ist. Tesseract kann nach diesem Verarbeitungsverfahren 'Hauptbahnhot ZOB' lesen, was dem tatsächlichen Text 'Hauptbahnhof ZOB' sehr ähnlich ist.

Abbildung 5.27: Binarisierung über ODER-Verknüpfung des adaptiven Gaußschen Schwellwertverfahren und der Otsu-Binarisierung

Diese Verarbeitung funktioniert optimal, wenn der Text zuvor möglichst gut ausgeschnitten wird. Sollte die Lokalisierung des Textes fehlschlagen, so kann möglicherweise die Otsu-Binarisierung durch einen sehr hellen Hintergrund keinen passenden Schwellwert finden. Im unvorteilhaftesten Fall liegen der Text und der Hintergrund der Anzeige auf einer Seite des Schwellwertes. In diesem Fall gehen die gesamten Informationen verloren. Die Gefahr hierfür besteht durch die Vorverarbeitung besonders bei Anzeigen mit grünen LEDs. Abbildung 5.28 zeigt einen solchen Fall.

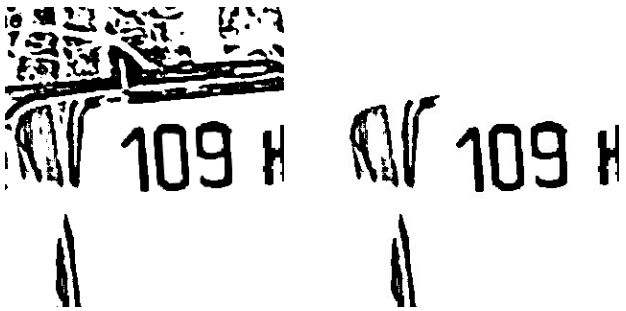
(a) (b)

(c) (d)

Abbildung 5.28: Fehlerhafte Binarisierung durch schlechte Textlokalisierung: (a) Originalausschnitt des Textfelds; (b) schlechte Lokalisierung des Texts durch EAST (grüner Rahmen); (c) vorverarbeitetes Bild für die Binarisierung; (d) fehlerhafte Binarisierung

5.5 Nachträgliche Nummerlokalisierung

Da die genaue Lokalisierung der Busnummern durch EAST nicht mit der gewünschten Genauigkeit erfolgt und so nach der Binarisierung Reste vom Hintergrund die Klassifizierung stören, ist eine nachträgliche Zeichenlokalisierung sinnvoll. Hierfür können die Eigenschaften der Konturen des invertierten binarisierten Bildes genutzt werden.
Da die Zeichen durch den geometrischen Ausschnitt (img) stets in einer ähnlichen Größe sind, können zu kleine und zu große Konturen (cnt) gefiltert werden. Als Grenzen für die erlaubten Dimensionen werden hier für die Höhe $0,15 \cdot h_{img} < h_{cnt} < 0,5 \cdot h_{img}$ und für die Breite $0,05 \cdot h_{img} < h_{cnt} < 0,5 \cdot h_{img}$ gewählt. Besonders die Breite ist großzügig ausgelegt, da die 1 schmaler als andere Ziffern ist und fusionierte Nummern so nicht direkt verworfen werden. Auch bei einer zu geringen Fläche der gefundenen Silhouette muss diese aussortiert werden. Als Schwellwert kommt hier $0,005 \cdot A_{img}$ in Frage, unterhalb dessen die Formen keine plausiblen Kandidaten sind. Abbildung 5.29 zeigt ein Beispiel dieser Filterung.



(a) (b)

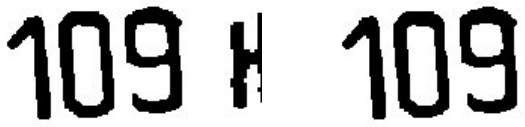
Abbildung 5.29: Nachträgliche Filterung für die Lokalisierung von Ziffern: (a) originaler Ausschnitt; (b) Filterung zu kleiner und zu großer Konturen

Im nächsten Schritt müssen die Konturen mit einer ähnlichen Größe wie die der Ziffern gefiltert werden. Hierfür kann sich zunutze gemacht werden, dass die Nummer im besten Fall mittig und nur selten am Rand des Bildes liegt. Des Weiteren sind die störenden Formen meist am linken und oberen Rand des Bildes zu sehen, da dort der Bus endet und der tendenziell hellere Hintergrund zu sehen ist, der störende Konturen verursacht. Deshalb wird die Silhouette, welche ihren Schwerpunkt am nächsten zur Mitte des Bildes hat, als Ziffer festgelegt. Der Schwerpunkt der Konturen setzt sich aus den Punkten der Formel 5.4 zusammen.
Da die Busnummer aus mehreren Zeichen bestehen kann, muss noch weitergesucht werden. Zunächst werden nur Formen mit einer ähnlichen Höhe von $0,8 \cdot h_{mid} < h_{cnt} < 1,2 \cdot h_{mid}$ beibehalten, wobei h_{mid} die Größe der zuvor bestimmten Ziffer ist. Zudem müssen die Positionen der Ziffern auf der Ordinate ähnlich sein, wodurch Konturen mit einem wesentlich höheren oder tieferen Schwerpunkt verworfen werden. Die Abbildung 5.30 zeigt diesen Verarbeitungsschritt.

(a) (b)

Abbildung 5.30: Nachträgliche Filterung für die Lokalisierung von Ziffern: (a) vorgefiltertes Bild; (b) Filterung von Konturen mit anderer Höhe oder Position auf der Ordinate als die zentralste Kontur

Die Ziffern einer Zahl stehen in der x-Ebene nah beieinander, weshalb zuletzt auch die Silhouetten entfernt werden, welche einen zu hohen Abstand zu den Konturen besitzen, die direkt oder über andere Konturen mit der zuvor bestimmten Ziffer verbunden sind. Hierfür werden alle Konturen entfernt, die auf der Abszisse über die halbe Breite der festgelegten Ziffer vom äußeren Rand einer zur Mitte verbundenen Kontur entfernt sind (vergleiche Abbildung 5.31.)



(a) (b)

Abbildung 5.31: Nachträgliche Filterung für die Lokalisierung von Ziffern: (a) vorgefiltertes Bild; (b) Filterung von Konturen mit höherer Entfernung zu anderen, mit der zentralen Ziffer verbundenen, Konturen

Die übrigbleibenden Formen können als Ziffern angenommen werden und für die Klassifizierung verwendet werden.

Diese Art der Lokalisierung ist nicht optimal, da im Fall einer schlecht zentrierten Busnummer, eine falsche Kontur ausgewählt werden kann. Trotzdem kann die nachträgliche Lokalisierung der Klassifizierung helfen, bessere Ergebnisse zu erhalten, da Tesseract bei gestörten Bildern Probleme hat.
Falls eine Lokalisierung mit EAST erfolgreich ist, muss die zweite Suche nicht durchgeführt werden. Wenn bei grüner Schrift EAST erfolglos ist, kann die zweite Lokalisierung häufig ebenfalls keinen Text finden, da dieser bei der Binarisierung verlorengehen kann.

5.6 Texterkennung über Tesseract

5.6.1 Fahrtziel

Um das Fahrtziel zu erkennen, wird der binarisierte Ausschnitt der Anzeige an Tesseract weitergegeben. Als Sprache wird bei Tesseract Deutsch ausgewählt, da so auch die Umlaute erfasst werden können. Dabei wird zusätzlich eine Whitelist mit allen Buchstaben und benötigten Zeichen angelegt. Ausgenommen sind hierbei alle Ziffern, da in keinem Fahrtziel eine Nummer enthalten ist und so Fehlklassifikationen, wie in Abschnitt 5.3 beschrieben, vermieden werden können. Tesseract wird zudem auf die Klassifizierung eines Textblockes eingestellt, da das Ziel in bis zu zwei Zeilen aufgeteilt sein kann.

5.6.2 Busnummer

Bei der Busnummer kann ähnlich wie beim Text vorgegangen werden. Hier enthält die Whitelist hingegen ausschließlich Ziffern. Bei der Sprache wird zudem Englisch verwendet, da so die Erkennung leicht verbessert ist. Dies hängt mit dem separaten Training für die einzelnen Sprachen zusammen, wodurch sich die Genauigkeit der Erkennung für verschiedene Sprachen unterscheidet [36]. Der Verarbeitungsmodus wird auf die Erkennung eines Wortes eingestellt, da die Buslinie immer aus einer zusammenhängenden Zahl besteht.

5.7 Ergebnisverifikation

Um möglichst gute und zuverlässige Ergebnisse zu erhalten, ist eine Überprüfung der Ausgabe der Texterkennung sinnvoll. Zum einen kann in manchen Fällen von Tesseract kein oder nur ein völlig falscher Text erkannt werden, was vor der Audioausgabe abgefangen werden muss. Zum anderen kann aber auch das Ergebnis bis auf kleine Abweichungen richtig sein, wie bei der Texterkennung in Abbildung 5.27 bei der 'Hauptbahnhof ZOB' als 'Hauptbahnhot ZOB' erkannt wird. In diesem Fall soll natürlich nur der korrekte Text vorgelesen werden. Die Vorhersagen von Busnummer und Text können sich im besten Fall gegenseitig bestätigen. Für eine robuste Erkennung müssen kleine Fehler erkannt und behoben werden.

5.7.1 Fahrtziel

Um zu prüfen, ob der ermittelte Text tatsächlich einer Buslinie entspricht, wird eine Datenbank aus Fahrtzielen und den dazugehörigen Linien angelegt. Der Hamburger Verkehrsverbund bietet hierfür die Möglichkeit, die Rohdaten der Fahrpläne einzusehen [15]. Datenbanken für andere Verkehrsbetriebe können nachträglich hinzugefügt werden. Durch einen Zugriff auf die GPS-Koordinaten kann beim Start der Buserkennung die passende Datenbank für den Standort ausgewählt und eingelesen werden. Die Implementierung über den Standort ist jedoch außerhalb des Rahmens dieser Arbeit.
Um den von Tesseract ausgelesenen Text mit den möglichen Fahrtzielen abzugleichen, kann zum Beispiel die Levenshtein-Distanz [35] verwendet werden. Diese berechnet sich aus der Anzahl der benötigten eingesetzten, ersetzten oder gelöschten Zeichen, um die erste Zeichenkette in die Zweite zu transformieren. So ergibt sich eine Levenshtein-Distanz von 3 zwischen den Wörtern 'Samstag' und 'Sonntag' (siehe Abbildung 5.32). Die maximale Distanz ist dabei die Anzahl der Zeichen im längeren Wort und die minimale Distanz ist der Betrag der Subtraktion der Längen beider Wörter.

	S	A	M	S	T	A	G
S	0	1	2	3	4	5	6
O	1	1	2	3	4	5	6
N	2	2	2	3	4	5	6
N	3	3	3	3	4	5	6
T	4	4	4	4	3	4	5
A	5	4	5	5	4	3	4
G	6	5	5	6	5	4	3

Abbildung 5.32: Levenshtein-Distanz zwischen den Wörtern 'Samstag' und 'Sonntag'

Der Eintrag mit der geringsten Levenshtein-Distanz zu dem ermittelten Text ist das ähnlichste Fahrtziel. Da jedoch auch für falsche Vorhersagen ein Eintrag mit dem geringsten Abstand gefunden wird, muss abschließend eine prozentuale Übereinstimmung errechnet werden, anhand welcher das Ergebnis verifiziert wird. Hierfür wird der Quotient von der Levenshtein-Distanz zum ähnlichsten Eintrag d_{lev} und der Länge des Eintrags $l_{Eintrag}$

gebildet:

$$F_1 = \frac{d_{lev}}{l_{Eintrag}}. \tag{5.14}$$

Wenn der So berechnete Fehleranteil F oberhalb einer festgelegten Grenze liegt, muss die Vorhersage verworfen werden. Im Weiteren wird zunächst die Grenze bei $F_1 = 0.38$ gesetzt, da die durchschnittliche minimale Levenshtein-Distanz der möglichen Fahrtziele vom Hamburger Verkehrsverbund $\varnothing d_{lev,best} = 6,61$ und die durchschnittliche Länge der Texte $\varnothing l_{Eintrag} = 17,33$ beträgt.

Für die Fahrtzielanzeigen von Fahrzeugen des Hamburger Verkehrsverbunds gibt es einige Sonderfälle. So haben beschleunigte Buslinien beispielsweise die Bezeichnung 'Schnellbus' vor dem Ziel stehen, während in der Datenbank diese Zusätze nicht enthalten sind. Diese Begriffe müssen gesondert herausgearbeitet werden, damit sie beim Finden des nächsten Eintrags nicht stören. Zudem kann so dem Nutzer diese Information ebenfalls mitgeteilt werden. Sofern ein Wort mit einer Levenshtein-Distanz von $d_{lev} \leq 2$ zu 'Schnellbus' gefunden wird, kann angenommen werden, dass es sich um einen solchen handelt.
Bei manchen Fahrtzielen ist zusätzlich ein Zwischenstopp angegeben, welcher nicht in der Datenbank enthalten ist und so Probleme bei der Vorhersage verursacht. Diese stehen immer hinter dem Zusatz 'über', sodass der erkannte Text nach einem Wort mit einer Distanz von $d_{lev} \leq 1$ zu 'über' wegfallen kann.
Diese Sonderfälle müssen für andere Regionen gegebenenfalls einzeln angepasst werden.

5.7.2 Busnummer

Nach der Überprüfung des Textes muss auch die Linie abgeglichen werden. Hierfür müssen alle Fahrtziele mit der erkannten Linie aus der Datenbank gelesen und mit dem vorhergesagten Text verglichen werden. Stimmt eines der Fahrtziele mit dem erkannten Text überein, handelt es sich hierbei höchstwahrscheinlich um den tatsächlichen Bus und das Ergebnis kann akustisch ausgegeben werden. Falls kein passender Eintrag gefunden wird, kann der Quotient aus Levenshtein-Distanz zwischen den Einträgen und der Vorhersage d_{lev} und der Länge der Vorhersage $l_{Vorhersage}$

$$F_2 = \frac{d_{lev}}{l_{Vorhersage}} \tag{5.15}$$

berechnet werden. Wenn hier ein geringer Wert vorliegt, ist wahrscheinlich die Nummer korrekt und bei der Texterkennung ein Fehler aufgetreten. Der Schwellwert hierfür sollte recht niedrig sein, da die Texterkennung aufgrund der längeren Zeichenketten robuster gegenüber einzelnen Fehlern ist. Dies liegt an der höheren Levenshtein-Distanz zwischen den Einträgen. So ist 'Hagenow ZOB' das ähnlichste Fahrtziel zu 'Hauptbahnhof ZOB' mit einer Levenshtein-Distanz von 8. Für die Linie '5' hingegen können 11 Linien gefunden werden, die eine Distanz von lediglich 1 besitzen. Deshalb muss die Nummer verworfen werden, falls die Fahrtziele der ermittelten Linie zu stark vom vorhergesagten Text abweichen. Ein sinnvoller Schwellwert kann $F_2 = 0.15$ sein.
Sofern kein Text, jedoch eine Linie ausgelesen wird, kann die Linie zwischengespeichert werden und bei erneuter Erkennung im nächsten Bild als richtig angenommen werden.

6 Evaluation

In diesem Kapitel werden die Ergebnisse aus der Entwicklung mithilfe eines geeigneten Verifikationsdatensatzes errechnet und bewertet.

6.1 Verifikationsdatensatz

Für die Verifikation der Texterkennung wird ein Verifikationsdatensatz angelegt. Dieser besteht aus 50 verschiedenen Busfronten, die von der Busdetektion [2] aus Bildern mehrerer Videos ausgeschnitten wurden. Dabei decken die Bilder möglichst viele unterschiedliche Einsatzfälle und Situationen ab.
Im Datensatz sind verschiedene Bustypen und somit auch unterschiedliche Anzeigen vorhanden. Die Farbe der Schriften variiert zwischen Orange, Grün und Weiß. Es gibt im Verifikationsdatensatz zudem Buslinien mit 1 bis 3 Ziffern in der Nummer und Anzeigen mit Texten, die über 1 bis 2 Zeilen gehen. Dabei ist auch die Schriftgröße der Texte unterschiedlich. Bei den Fahrtzielen sind bei manchen Bildern zudem Zusätze wie 'Schnellbus' oder 'über Tangstedt' enthalten.
Die Busse und somit auch die Texte in den Anzeigen sind teilweise nicht optimal horizontal ausgerichtet.
Die Wetterbedingungen variieren zwischen bedeckt und sonnig, wodurch auch unterschiedliche Spiegelungen entstehen. Teilweise sind sehr helle Spiegelungen auf den Anzeigen, wodurch diese schwerer lesbar werden. In einem Fall ist ein Teil des Textes nicht mehr lesbar, jedoch kann ein Mensch aus dem lesbaren Teil die Haltestelle erkennen.
Obwohl die Erkennung für Tageslicht ausgelegt ist, wird dem Datensatz ein Bild bei Nacht hinzugefügt. Allerdings ist dieses ohne kritische Spiegelungen, um Probleme bei der Verarbeitung zu vermeiden.
Die Hintergründe in den Bildern sind ebenfalls nicht gleich. In mehreren Bildern sind Schriftzüge von Schildern oder Reklamen im Hintergrund erkennbar.
Auch die Qualität der Bilder unterscheidet sich, wobei die Anzeigen in allen Bildern für

einen Menschen lesbar sind. Einige Bilder haben nur eine geringe Auflösung, während andere zum Teil sehr hoch aufgelöst sind, wodurch die einzelnen LEDs in den Anzeigen sichtbar werden und die Zeichen keine zusammenhängenden Blöcke mehr ergeben. Es unterscheidet sich auch die Schärfe, da die Bilder mit unterschiedlichen Kameras aufgenommen wurden und zudem einige Aufnahmen leicht verschwommen sind.
Die Abbildung 6.1 zeigt 4 Busse mit verschiedenen der erläuterten Herausforderungen.

(a) (b)

(c) (d)

Abbildung 6.1: Beispielbilder aus dem Verifikationsdatensatz mit den unterschiedlichen Herausforderungen: (a) grüner Text über zwei Zeilen mit Abkürzungen aus einem Buchstaben; (b) unscharfes Bild mit weißem Text über zwei Zeilen und einem Textzusatz; (c) schwierige Lichtverhältnisse durch Spiegelungen sowie orange Schrift und eine Abkürzung; (d) Nummer bestehend aus einer Ziffer und teilweise unkenntlicher Text durch Spiegelungen bei geneigter Anzeige in Orange

Der vorgestellte Datensatz wird im Folgenden verwendet, um die einzelnen Verarbeitungsschritte zu evaluieren. Nachfolgend sind in Tabelle 6.1 die Verteilung der unterschiedlichen Bedingungen aufgelistet.

Tabelle 6.1: Anzahl der Bilder mit verschiedenen Eigenschaften im Verifikationsdatensatz

Besonderheit	Anteil im Verifikationsdatensatz
Text über eine Zeile	21/50
Text über zwei Zeilen	29/50
Orange Schrift	28/50
Weiße Schrift	11/50
Grüne Schrift	11/50
Bilder mit Spiegelungen	17/50

6.2 Textlokalisierung und Bildverarbeitung

6.2.1 Geometrische Einteilung

Mit den gewählten Bildausschnitten aus Abschnitt 5.3 kann die Trennung von Fahrtziel und Linie zuverlässig vorgenommen werden. Bei dem Verifikationsdatensatz werden lediglich bei zwei Bildern Teile des Textes abgeschnitten, jedoch ist dies tragbar, da in diesen Fällen der Großteil des Textes noch vorhanden ist. Bei lediglich einem Ausschnitt der Zahlen sind zwei Buchstaben zu sehen, wodurch es hier zu einem Fehler bei der späteren Lokalisierung mit EAST kommen kann. Tabelle 6.2 zeigt die Anzahl der Bilder im Verifikationsdatensatz, bei denen die geometrische Einteilung erfolgreich ist.

Tabelle 6.2: Genauigkeit der groben Ausschnitte von Text und Nummer auf dem Verifikationsdatensatz

Ausschnitt	Eigenschaft	Anteil im Verifikationsdatensatz
Text	vollständig	48/50
	enthält < 2 Nummern	50/50
Nummer	vollständig	50/50
	enthält < 2 Buchstaben	49/50

6.2.2 EAST

Die genaue Textlokalisierung durch EAST wird nach der geometrischen Einteilung für das Fahrtziel und die Busnummer vorgenommen.

Fahrtziel

Bei der Erkennung des Textes auf den Anzeigen muss dieser möglichst vollständig erkannt und ausgeschnitten werden. Hierbei ist auch der Winkel entscheidend, da die Klassifizierung mit Tesseract einen optimal horizontal ausgerichteten Text benötigt. Zudem wird ein rechteckiger Ausschnitt um den lokalisierten Text gewählt, welcher bei perfekter Ausrichtung am kleinsten ist.
Wenn fast alle Buchstaben im Ausschnitt liegen, kann das Ziel durch die Textverifikation erkannt werden, wodurch dies die Bedingung für eine ausreichend gute Lokalisierung ist. Ist der Ausschnitt zu klein, kann zwar möglicherweise trotzdem durch die Levenshtein-Distanz das richtige Ziel erkannt werden, jedoch wird die Wahrscheinlichkeit für eine Fehlklassifizierung stark angehoben. Bei einem zu großen Ausschnitt können bei der Binarisierung die Informationen verlorengehen oder Hintergrundrauschen verursacht werden. In beiden Fällen wird die Klassifizierung erschwert.

Tabelle 6.3: Genauigkeit der Lokalisierung des Fahrtziels mittels EAST auf dem Verifikationsdatensatz

Qualität der Erkennung	Anteil im Verifikationsdatensatz
Text erkannt	50/50
Guter Ausschnitt	44/50
Zu großer Ausschnitt	1/50
Zu kleiner Ausschnitt	5/50
Gute Textausrichtung	42/50

Die Tabelle 6.3 zeigt, dass in allen Bildern ein Text erkannt werden kann, jedoch ist das lokalisierte Textfeld teilweise zu klein. Dies ist ausschließlich bei Fahrtzielen über zwei Zeilen der Fall, wobei dann die untere Zeile nicht erkannt wird. $\frac{24}{29} = 82,76\%$ der zweizeiligen Anzeigen werden vollständig erkannt, während $\frac{21}{21} = 100\%$ der einzeiligen Texte vollständig lokalisiert werden. Als Korrektur könnte das erkannte Textfeld um die eigene Höhe nach unten erweitert werden, sodass auch die zweite Zeile immer im Feld liegt. Allerdings würde so in allen anderen Fällen der Text nicht mehr optimal ausgeschnitten

werden, wobei dies nicht so kritisch ist, da unterhalb der Anzeigen die Lichtverhältnisse ähnlich bleiben und so die Binarisierung weiterhin funktionieren kann. Da im Regelfall auch bei zweizeiligen Anzeigen die Erkennung erfolgreich ist, wird auf diese Korrektur verzichtet.
Die Ausrichtung der Texte ist in 42 Fällen nahezu horizontal, während in 8 Fällen noch ein kleiner Winkel vorhanden ist, welcher jedoch nicht unbedingt kritisch für die Klassifikation ist. Die größte Abweichung im Winkel ist in Abbildung 6.2 zu sehen.

Abbildung 6.2: Text Detektion über EAST (grüner Rahmen) mit nicht optimaler Winkelkorrektur

Busnummer

Bei der Busnummer ist es wichtig, dass die Nummer vollständig zu sehen ist, da eine Korrektur wie beim Fahrtziel nicht möglich ist. Tabelle 6.4 zeigt die Anzahl der erfolgreichen Lokalisierungen der Busnummer im Verifikationsdatensatz.

Tabelle 6.4: Genauigkeit der Lokalisierung der Busnummer mittels EAST auf dem Verifikationsdatensatz

Qualität der Erkennung	Anteil im Verifikationsdatensatz
Nummer erkannt	9/50
Guter Ausschnitt	9/50
Gute Textausrichtung	7/50

Anders als bei der Textlokalisierung kann für die Nummern lediglich eine geringe Genauigkeit von $\frac{9}{50} = 18\%$ erreicht werden. Dieses Ergebnis ist nicht ausreichend, weshalb die nachträgliche Nummerlokalisierung beim binarisierten Bild angewendet wird. Von den gefundenen Ausschnitten ist die Ausrichtung in 2 Fällen nicht optimal, jedoch ausreichend für die Weiterverarbeitung.

6.2.3 Nachträgliche Nummerlokalisierung

Da die Lokalisierung der Nummern über EAST nicht die gewünschten Ergebnisse liefert, wird zusätzlich eine Erkennung nach der Verarbeitung des Bildes eingesetzt. Diese kann jedoch nur dann funktionieren, wenn bei der Binarisierung die Informationen nicht verlorengehen. Die Ergebnisse der nachträglichen Nummerlokalisierung sind in Tabelle 6.5 eingetragen.

Tabelle 6.5: Genauigkeit der Binarisierung ohne vorige Lokalisierung durch EAST und Genauigkeit der nachträglichen Lokalisierung der Busnummer

Eigenschaften der Algorithmen	Anteil der verarbeiteten Bilder
Binarisiertes Bild enthält Informationen	43/50
Zahl lokalisiert	42/50

Bei der Verarbeitung der Bilder ohne vorige Lokalisierung durch EAST werden die Informationen in Form der Ziffern in $\frac{43}{50} = 86\%$ der Fälle beibehalten. Wie in Unterabschnitt 5.4.4 beschrieben, ist die Verarbeitung auf einem großen Ausschnitt mit grüner Schrift fehleranfällig, weshalb 6 der 11 grünen Anzeigen nach der Binarisierung keine Informationen mehr enthalten. Lediglich bei einer Anzeige mit Schrift in Orange, gehen die Ziffern durch die Verarbeitung verloren.
Bei den Bildern, welche nach der Verarbeitung die Zahl vollständig enthalten, kann diese in $\frac{42}{43} = 97,67\%$ der Fälle erfolgreich extrahiert werden. Lediglich in zwei der erfolgreichen Fällen sind einzelne Ziffern nicht ganz vollständig, wobei für einen Menschen die Zahl lesbar bleibt (siehe Abbildung 6.3).

(a) (b)

Abbildung 6.3: Unvollständige Lokalisierung einer Ziffer: (a) Ausschnitt des binarisierten Bildes; (b) unvollständige nachträgliche Lokalisierung

6.3 Texterkennung über Tesseract und Ergebnisverifikation

Die verarbeiteten Ausschnitte der Bilder des Datensatzes werden von Tesseract ausgelesen. Im Folgenden werden die einzelnen Ergebnisse separat betrachtet, da die Zusammensetzung von Busnummer und Fahrtziel nur für Videos funktioniert. Die genaue Auswertung des Verifikationsdatensets befindet sich im Anhang.

6.3.1 Fahrtziel

Das Auslesen des Textes ist neben der Genauigkeit von Tesseract auch von der Qualität der Vorverarbeitung abhängig. Nach einer schlechten Binarisierung ist das korrekte Auslesen des Fahrtziels schwierig.

Tabelle 6.6: Genauigkeit der Bildverarbeitung und Klassifizierung des Fahrtziels mit Tesseract

Qualität der Texterkennung	Anteil im Verifikationsdatensatz
Fahrtziel erkennbar	47/50
Fahrtziel vollständig	43/50
Fahrtziel richtig klassifiziert	40/50

Tabelle 6.6 zeigt, dass nach der Verarbeitung $\frac{47}{50} = 93\%$ der Fahrtziele für einen Menschen identifizierbar sind. Allerdings sind bei 4 der erkennbaren Ziele nur ein Teil der Buchstaben zu sehen, wodurch eine Klassifizierung schwierig ist.
Mittels Tesseract und Textverifikation können insgesamt $\frac{40}{50} = 80\%$ der Fahrtziele richtig klassifiziert werden. Auf die zumindest teilweise erkennbaren Bilder angepasst ergibt dies eine Genauigkeit von $\frac{40}{47} = 85,11\%$.
Die nicht erfolgreichen Klassifizierungen sind vor allem bei den nicht vollständig ausgeschnittenen Texten und bei Bildern in geringer Qualität wie in Abbildung 6.4. Allerdings können teilweise auch in diesen Fällen die richtigen Fahrtziele erkannt werden.

(a) (b)

Abbildung 6.4: Fehlerhafte Texterkennung bei geringer Bildqualität: (a) originaler, grober Ausschnitt; (b) verarbeiteter Ausschnitt zu Klassifizierung mit Tesseract

In lediglich einem Fall kann ein guter Ausschnitt in hoher Qualität nicht korrekt klassifiziert werden. Aus Abbildung 6.5 liest Tesseract 'Gla H ütt' statt 'Glashütte Markt über Tangstedt'. Dabei ist die einzige mögliche Fehlerquelle die nicht optimale horizontale Ausrichtung des Textes. Für die Verhinderung dieser Fehlerquelle kann eine nachträgliche Winkelanpassung hinzugefügt werden.

Abbildung 6.5: Nach Vorverarbeitung gut lesbare Anzeige mit schlechtem Ergebnis bei der Klassifizierung durch Tesseract

6.3.2 Busnummer

Für die Erkennung der Busnummer gelten die gleichen Anforderungen an die Verarbeitung wie bei der Fahrtrichtung. Da die Lokalisierung über EAST nicht die gewünschte Genauigkeit besitzt, ist die Verarbeitung hier um eine zweite Lokalisierung erweitert.
In Tabelle 6.7 sind die Ergebnisse der Texterkennung von den Busnummern über Tesseract dargestellt.

Tabelle 6.7: Genauigkeit der Bildverarbeitung und Klassifizierung der Busnummer mit Tesseract

Qualität der Busnummererkennung	Anteil im Verifikationsdatensatz
Busnummer erkennbar	46/50
Busnummer vollständig	45/50
Busnummer richtig klassifiziert	32/50

Obwohl die Busnummer häufiger nach der Verarbeitung vollständig lesbar als das Fahrtziel ist, kann nur eine Genauigkeit von $\frac{32}{50} = 64\%$ erreicht werden. Dies ist unter anderem darauf zurückzuführen, dass keine Korrektur von kleinen Fehlern vorgenommen werden kann wie beim Fahrtziel. Zusätzlich enthalten die Zahlen zum Teil gewisse Störeffekte (vergleiche Abbildung 6.6), welche zwar die Lesbarkeit für einen Menschen nicht beeinträchtigen, sich aber negativ auf die Klassifizierung auswirken. Ein Teil dieser Effekte kann durch die Anpassung des Algorithmus auf die gewählte Kamera eliminiert werden, wodurch sich die Genauigkeit weiter verbessert.

Abbildung 6.6: Ausschnitt einer von Tesseract zu klassifizierenden Zahl mit stärkeren Störeffekte

Die Genauigkeit der Klassifizierung der Busnummer erreicht fast die in [52] beschriebene Genauigkeit von 67.9%.
Um die Genauigkeit der Erkennung der Busnummer zu verbessern, muss hier ein Training von Tesseract vorgenommen oder ein verbessertes System zur Klassifizierung verwendet werden. Hierfür kommt zum Beispiel das Modell aus Unterabschnitt 5.1.3 in Frage, wobei eine genaue Lokalisierung der einzelnen Ziffern nötig wird. Ohne weitere Verarbeitungsschritte kann mit dem Netz nur eine Genauigkeit von $\frac{25}{50} = 50\%$ erreicht werden, was schlechter ist als bei der Erkennung mit Tesseract. Dies liegt unter Anderem daran, dass häufig mehrere Ziffern zu einer Kontur verschmelzen.

6.4 Laufzeit der Fahrtrichtungserkennung

Da für die Fahrtrichtungserkennung nur eine begrenzte Zeit zur Verfügung steht, muss diese möglichst schnell sein, um im besten Fall mehrfach die Vorhersage für einen Bus überprüfen zu können. Die Messung der Laufzeit des Systems erfolgt auf einem Intel Core i7-6700K Prozessor unter Ubuntu als Windows 10 Subsystem für Linux.
In Tabelle 6.8 wird die Laufzeit des Algorithmus zur Erkennung von Busnummer und Fahrtziel aus den bereits ausgeschnittenen Busfronten untersucht.

Tabelle 6.8: Laufzeit von Tesseract, EAST und der gesamten Fahrtrichtungserkennung für den Verifikationsdatensatz

Algorithmus		$\varnothing t$ in ms
Fahrtrichtungserkennung		675,0
EAST	Fahrtziel	156,3
	Busnummer	133,9
Tesseract	Fahrtziel	188,7
	Busnummer	183,1

Für das Auslesen von Busnummer und Fahrtziel wird im Schnitt $t_{ges} = 675,0\ ms$ benötigt. Die Zeiten für die Textdetektion durch EAST und für die Klassifikation durch Tesseract liegen bei $t_{EAST} = 156,3\ ms + 133,9\ ms = 290,2\ ms$ und $t_{Tess} = 188,7\ ms + 183,1\ ms = 371,8\ ms$, wodurch diese Teile hauptverantwortlich für die Gesamtlaufzeit sind.
Durch Auslagerung der Berechnung auf eine parallel berechnende Hardware wie einer Grafikkarte kann die Textdetektion durch EAST beschleunigt werden [54], während bei Tesseract dies nur eine geringfügige Verbesserung bewirkt [24].

Die Buslokalisierung über den Haar Cascade Classifier benötigt eine durchschnittliche Laufzeit von $t_{HCC} = 67,3\ ms$ für ein Bild der Größe 1920x1080 Pixel. Damit ist diese Klassifizierung sehr schnell und nicht problematisch für die Laufzeit des gesamten Systems.

In der im Unterabschnitt 3.3.2 abgeschätzten Zeit zur Erkennung von etwa 5 s, können je nach Anzahl der gefundenen Busse unterschiedlich viele Bilder verarbeitet werden. Nur in seltenen Fällen sollten mehr als 2 Busse gleichzeitig zu sehen sein, wodurch sich eine

Bearbeitungszeit von maximal

$$t_{max,2} \;=\; t_{HCC} + 2 \cdot t_{ges} = 1417,3ms \tag{6.1}$$

ergibt. So wird ein Bus mindestens 3-mal verarbeitet, bevor dieser aus dem Sichtfeld der Kamera verschwindet. Meistens sollte jedoch nur ein Bus gleichzeitig zu sehen sein, wodurch sich die Zeit der Verarbeitung weiter auf

$$t_{max,1} \;=\; t_{HCC} + t_{ges} = 742,3ms \tag{6.2}$$

verringert. So kann ein Bus 6- bis 7-mal verarbeitet werden, wodurch eine korrekte Klassifizierung sicher sein sollte.

Um eine mehrfache Verarbeitung eines Busses zu gewährleisten, sollte die Fahrtrichtung maximal für die zwei größten und somit dichtesten Fahrzeuge ausgewertet werden.

6.5 Überprüfung der Anforderungen

Das vorgestellte System kann die zuvor aufgestellten Anforderungen Erfüllen. Busse des Hamburger Verkehrsverbundes können mit einer ausreichenden Geschwindigkeit und Genauigkeit erkannt werden. Lediglich die Implementierung auf dem Zielsystem bleibt offen, wodurch eine Endgültige Aussage über die Geschwindigkeit nicht möglich ist.

6.6 Limitierungen

Die Buserkennung ist wie in Abschnitt 3.4 auf die Erkennung von Fahrzeugen des Hamburger Verkehrsverbundes bei Tageslicht ausgelegt, wobei auch bei Dunkelheit die Klassifizierung erfolgen kann, sofern der Bus ausreichend von außen beleuchtet ist und keine starken Reflexionen auf der Anzeige die Lesbarkeit stören. Es lassen sich zudem maximal zwei Fahrzeuge gleichzeitig untersuchen.

7 Fazit

In dieser Arbeit konnte eine videobasierte Fahrtzielerkennung entwickelt werden, welche Menschen mit eingeschränktem Sichtfeld helfen kann, sich selbstständiger im Straßenverkehr fortzubewegen. Die Erkennung ist dabei für die Nutzung bei Tageslicht am Standort Hamburg und somit für Fahrzeuge des Hamburger Verkehrsverbunds ausgelegt.
Aufbauend auf einem in [2] vorgestellten Haar Cascade Classifier, welcher die Lokalisierung der Busse in den Bildern übernimmt, wurde ein Verfahren zum Auslesen der Fahrtrichtungsanzeigen entwickelt. Die vorgestellte Methode kann mit einer Genauigkeit von 80% das Fahrtziel bestimmen, während die Buslinie mit 64% Genauigkeit erkannt wird.
Zwar ist eine eindeutige Aussage über die Fahrtrichtung erst möglich wenn sowohl Fahrtziel als auch Busnummer bekannt sind, jedoch kann in den meisten Fällen der richtige Bus auch aus lediglich einer der beiden Komponenten bestimmt werden. Deshalb ist die vorgestellte Methode geeignet als Fahrtrichtungserkennung für die Verwendung im elektronischen Blindenführhund.

Der nächste Schritt der Entwicklung ist die Implementierung der Fahrtzielerkennung auf der Hardware des elektronischen Blindenhundes. Dies beinhaltet ebenfalls die Abstimmung der Software an die Eigenschaften der verwendeten Kamera. Des Weiteren muss die akustische Ausgabe für den Nutzer eingearbeitet werden, wobei eine Steuerung nötig ist, welche die Ergebnisse der Busnummer- und Fahrtzielerkennung zusammenführt und eine Entscheidung trifft, welche Informationen ausgegeben werden. Dabei ist auch eine Nachverfolgung der im Bild zu sehenden Busse notwendig, um auch mehrere Fahrzeuge gleichzeitig zuordnen zu können. Diese kann zum Beispiel über die Position der Busse im Bild arbeiten.

Um die Funktionalität zu erweitern, sollte zunächst die Verarbeitung der Erkennung auch für nächtliche Verhältnisse angepasst werden. Hierfür muss die softwareseitige Entspiegelung nach Tageszeit oder gemessener Helligkeit dynamisch angeglichen werden.

Zusätzlich zu der frontalen Erkennung der Busse wäre auch eine Erkennung von der Seite möglich, da viele Busse auch eine seitliche, zum Fahrgast ausgerichtete Anzeige besitzen. So lassen sich auch stehende Busse erkennen.
Damit die Fahrtzielerkennung auch außerhalb von Hamburg genutzt werden kann, muss die Erkennung für weitere Standorte angepasst werden. Hierfür müssen weitere Datenbanken eingearbeitet werden und gegebenenfalls die grobe Einteilung der Busfronten in Text und Nummer angepasst werden.

Um die Qualität der Erkennung zu verbessern, kann Tesseract entweder mit den Daten aus der Verarbeitung trainiert oder durch eine verbesserte Texterkennungssoftware ausgetauscht werden. Dies ist vor allem hilfreich bei der Erkennung der Busnummer, da die Genauigkeit von 64% noch nicht ganz zufriedenstellend ist.
Der vielversprechendste Ansatz, die Genauigkeit anzuheben, ist eine Bestimmung des Standortes vom Nutzer, um die Anzahl der möglichen Buslinien, die an der Haltestelle einen Stopp einlegen, einzugrenzen. So müsste in den meisten Fällen nur zwischen wenigen Linien unterschieden werden, sodass die richtige Zuordnung leichter fällt. Die Fahrpläne und Standorte der Haltestellen müssen hierfür entweder in einer Datenbank abgelegt oder flexibel über Dienste wie *Google Maps* abgefragt werden.
Zuletzt kann auch die Buserkennung mit dem Haar Cascade Classifier durch beispielsweise YOLO ausgetauscht werden, um eine höhere Genauigkeit zu erreichen. Dabei ist jedoch auch zu beachten, dass die Geschwindigkeit der neuen Klassifikation nicht zu gering sein darf, damit die Echtzeitanforderungen an das System nicht überschritten werden.

Literaturverzeichnis

[1] ABBYY: *ABBYY Cloud OCR SDK | Cloudbasierte Dokumentenverarbeitung für Ihre Anwendungen*. – URL https://www.abbyy.com/de/cloud-ocr-sdk/. – [Online; Stand 19. April 2021]

[2] Azad, Zaheen: *Real-time Bus number detection*. 2019. – Projektbericht, HAW Hamburg

[3] Bundesregierung: *Autonomes Fahren in die Praxis holen*. 2021. – URL https://www.bundesregierung.de/breg-de/aktuelles/faq-autonomes-fahren-1852070. – [Online; Stand 5. April 2021]

[4] BUS-BILD.de. – URL https://www.bus-bild.de/. – [Online; Stand 20. März 2021]

[5] Cheng, C. C. ; Tsai, C. M. ; Yeh, Z. M.: Detection of Bus Route Number via Motion and YCbCr Features. In: *2014 International Symposium on Computer, Consumer and Control*, 2014, S. 31–34

[6] Cortes, Corinna ; Vapnik, Vladimir: Support-Vector Networks. In: *Mach. Learn.* 20 (1995), September, Nr. 3, S. 273–297. – URL https://doi.org/10.1023/A:1022627411411. – ISSN 0885-6125

[7] Dahlkemper, Jörg: *Deep Learning in der Bildverarbeitung | Deep Learning*. 2020. – Vorlesung, HAW Hamburg

[8] Dahlkemper, Jörg: *Deep Learning in der Bildverarbeitung | Einführung in das Thema*. 2020. – Vorlesung, HAW Hamburg

[9] Dahlkemper, Jörg: *Deep Learning in der Bildverarbeitung | Face Recognition*. 2020. – Vorlesung, HAW Hamburg

[10] Dahlkemper, Jörg: *Deep Learning in der Bildverarbeitung | Training von Künstlichen Neuronalen Netzen*. 2020. – Vorlesung, HAW Hamburg

[11] DAVIDEROMANO: *Change Tesseract output with words coming from an external dictionary.* 2019. – URL https://github.com/tesseract-ocr/tesseract/issues/2391. – [Online; Stand 17. März 2021]

[12] GASTEIGER, J. ; ZUPAN, J.: Neuronale Netze in der Chemie, 1993, S. 510–536

[13] GOOGLE: *Vision AI.* 2021. – URL https://cloud.google.com/vision/. – [Online; Stand 3. April 2021]

[14] GRINGEL, Fabian: *Test von OCR-Scannern: die beste Software für Texterkennung.* 2020. – URL https://dida.do/de/blog/how-to-choose-the-best-ocr-tool-for-your-project. – [Online; Stand 21. März 2021]

[15] HAMBURGER VERKEHRSVERBUND. – URL https://suche.transparenz.hamburg.de/dataset?esq_type=app&esq_type=dataset&esq_type=document&f=PDF&f=HTML&f=ZIP&f=CSV&f=XML&f=xlsx&f=XLS&f=gml&f=wms&f=wfs&f=rar&f=TXT&f=jpg&f=jpeg&f=ascii&f=png&f=dxf&f=web&f=citygml&f=ert&f=docx&f=ov2&f=tiff&f=xsd&l=dl-de-by-2.0&l=dl-de-zero-2.0&esq_not_all_versions=true&esq_coverage_type=publication&e=vergleichbar&g=transport-und-verkehr&change_search=1&sort=publishing_date+desc%2Ctitle_sort+asc&esq_not_all_versions=true. – [Online; Stand 22. März 2021]

[16] HAMBURGER VERKEHRSVERBUND: *Der HVV in Zahlen 2018 / HVV figures for 2018.* 2018. – URL https://web.archive.org/web/20190722080602/https://www.hvv.de/resource/blob/18704/81a66a9e6639e33b08ef717d6fb041af/zahlenspiegel-data.pdf. – [Online; Stand 21. April 2021]

[17] HANGRONG PAN ; YI, C. ; TIAN, Y.: A primary travelling assistant system of bus detection and recognition for visually impaired people. In: *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2013, S. 1–6

[18] HOSANG, J. ; BENENSON, R. ; SCHIELE, B.: Learning non-maximum suppression. In: *CVPR*, 2017

[19] HUANG, S.: Influence of Different Convolutional Neural Network Settings on the Performance of MNIST Handwritten Digits Recognition. In: *2020 International Conference on Artificial Intelligence and Education (ICAIE)*, 2020, S. 1–6

[20] HUG-TECHNIK.COM: *Umrechnungstabelle Wellenlängen der sichtbaren Farben.* – URL https://www.hug-technik.com/inhalt/ta/farben.html. – [Online; Stand 17. April 2021]

[21] JASON BROWNLEE: *A Gentle Introduction to the Rectified Linear Unit (ReLU).* 2020. – URL https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/. – [Online; Stand 5. April 2021]

[22] KERAS: *MNIST digits classification dataset.* 2021. – URL https://keras.io/api/datasets/mnist/. – [Online; Stand 5. April 2021]

[23] KORSTANJE, Joos: *The k-Nearest Neighbors (kNN) Algorithm in Python.* 2021. – URL https://realpython.com/knn-python/. – [Online; Stand 15. April 2021]

[24] LIAO, Chejui: *Improving the quality of the output.* 2020. – URL https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html. – [Online; Stand 29. März 2021]

[25] LUZ SÁENZ GARZA, José de la: *Wie viele chinesische Schriftzeichen gibt es?.* – URL https://www.hutong-school.com/de/wie-viele-chinesische-schriftzeichen-gibt-es. – [Online; Stand 17. April 2021]

[26] LYDIASWELT: *Blind Bus fahren, das ist für mich wichtig.* – URL https://lydiaswelt.com/2018/07/04/blind-mit-dem-bus-fahren/. – [Online; Stand 9. Februar 2021]

[27] MICHAEL NIELSEN: *Improving the way neural networks learn.* 2019. – URL http://neuralnetworksanddeeplearning.com/chap3.html. – [Online; Stand 5. April 2021]

[28] MURPHY, Ken: *A History of the Sky.* 2011. – URL https://www.youtube.com/watch?v=PNln_me-XjI&t. – [Online; Stand 26. März 2021]

[29] NZUZI, V. Poltavets E. Graziani D.: *Algorithmische Anwendungen | Projekt Bildsegmentierung.* 2006. – URL http://www.gm.fh-koeln.de/~hk/lehre/ala/ws0506/Praktikum/Projekt/F_rot/Bildsegmentierung_Otsu.pdf. – [Online; Stand 6. März 2021]

[30] OPENCV: *About Open Cv.* 2021. – URL https://opencv.org/about/. – [Online; Stand 4. April 2021]

[31] OpenCV-Python Tutorials: *Image Thresholding*. – URL https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html. – [Online; Stand 6. April 2021]

[32] Physik für alle!: *Rayleigh-Streuung*. – URL https://physik.cosmos-indirekt.de/Physik-Schule/Rayleigh-Streuung. – [Online; Stand 6. April 2021]

[33] Pixolum: *Blende in der Fotografie > Erklärung für Anfänger*. 2020. – URL https://www.pixolum.com/blog/fotografie/blende. – [Online; Stand 5. April 2021]

[34] Redmon, J. ; Divvala, S. ; Girshick, R. ; Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, S. 779–788

[35] Schuster, Jörg: *Probabilistic models of pronunciation and spelling | Minimale Editierdistanz nach Levenshtein*. 2002. – URL https://www.cis.uni-muenchen.de/~micha/praesentationen/rechtschreibkorrektur/Levenshtein.html. – [Online; Stand 6. April 2021]

[36] Smith, Ray: *Building a Multilingual OCR Engine | Training LSTM networks on 100 languages and test results*. 2016. – URL https://github.com/tesseract-ocr/docs/blob/master/das_tutorial2016/7Building%20a%20Multi-Lingual%20OCR%20Engine.pdf. – [Online; Stand 1. April 2021]

[37] Statistisches Bundesamt: *Pressemitteilung Nr. 381 vom 24. Oktober 2016*. – URL https://www.destatis.de/DE/Presse/Pressemitteilungen/2016/10/PD16_381_227.html. – [Online; Stand 9. Februar 2021]

[38] Szegedy, C. ; Wei Liu ; Yangqing Jia ; Sermanet, P. ; Reed, S. ; Anguelov, D. ; Erhan, D. ; Vanhoucke, V. ; Rabinovich, A.: Going deeper with convolutions. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, S. 1–9

[39] Tesseract: *OCR Engine Comparison — Tesseract vs. EasyOCR*. 2021. – URL https://medium.com/swlh/ocr-engine-comparison-tesseract-vs-easyocr-729be893d3ae. – [Online; Stand 3. April 2021]

[40] Tesseract: *Tesseract User Manual*. 2021. – URL https://tesseract-ocr.github.io/tessdoc/. – [Online; Stand 17. März 2021]

[41] Viola, P. ; Jones, M.: Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* Bd. 1, 2001, S. I–I

[42] Wikimedia Commons: *File:EVAG O530 3413 Holthuser Tal.jpg — Wikimedia Commons, the free media repository*. 2020. – URL https://commons.wikimedia.org/w/index.php?title=File:EVAG_O530_3413_Holthuser_Tal.jpg&oldid=463289780. – [Online; Stand 21. März 2021]

[43] Wikimedia Commons: *File:First London Routemaster bus RM1640 (640 DYE) heritage route 9 Trafalgar Square 8 July 2006.jpg — Wikimedia Commons, the free media repository*. 2020. – URL https://commons.wikimedia.org/w/index.php?title=File:First_London_Routemaster_bus_RM1640_(640_DYE)_heritage_route_9_Trafalgar_Square_8_July_2006.jpg&oldid=475757837. – [Online; Stand 21 März 2021]

[44] Wikimedia Commons: *File:Polfilter ohne.jpg — Wikimedia Commons, the free media repository*. 2020. – URL https://commons.wikimedia.org/w/index.php?title=File:Polfilter_ohne.jpg&oldid=442939354. – [Online; Stand 25. März 2021]

[45] Wikimedia Commons: *File:Polfilter richtig.jpg — Wikimedia Commons, the free media repository*. 2020. – URL https://commons.wikimedia.org/w/index.php?title=File:Polfilter_richtig.jpg&oldid=442939348. – [Online; Stand 25. März 2021]

[46] Wikimedia Commons: *File:Rayleigh-Streuung von Sonnenlicht.png — Wikimedia Commons, the free media repository*. 2020. – URL https://commons.wikimedia.org/w/index.php?title=File:Rayleigh-Streuung_von_Sonnenlicht.png&oldid=456781824. – [Online; Stand 26. März 2021]

[47] Wikimedia Commons: *File:Syrian-bus.JPG — Wikimedia Commons, the free media repository*. 2020. – URL https://commons.wikimedia.org/w/index.

php?title=File:Syrian-bus.JPG&oldid=478763973. – [Online; Stand 21. März 2021]

[48] Wikimedia Commons: *File:Travego 100 2660.jpg — Wikimedia Commons, the free media repository*. 2020. – URL https://commons.wikimedia.org/w/index.php?title=File:Travego_100_2660.jpg&oldid=486003635. – [Online; Stand 21. März 2021]

[49] Wikimedia Commons: *File:VJ featureTypes.svg — Wikimedia Commons, the free media repository*. 2020. – URL https://commons.wikimedia.org/w/index.php?title=File:VJ_featureTypes.svg&oldid=452809388. – [Online; Stand 3. April 2021]

[50] Wikimedia Commons: *File:Arriva London bus VLA157 (LJ55 BSU) 2005 Volvo B7TL Alexander Dennis ALX400, Whitehall, route 159, 9 December 2005.jpg — Wikimedia Commons, the free media repository*. 2021. – URL https://commons.wikimedia.org/w/index.php?title=File:Arriva_London_bus_VLA157_(LJ55_BSU)_2005_Volvo_B7TL_Alexander_Dennis_ALX400,_Whitehall,_route_159,_9_December_2005.jpg&oldid=534943471. – [Online; Stand 4. April 2021]

[51] Wikimedia Commons: *File:Typical cnn.png — Wikimedia Commons, the free media repository*. 2021. – URL https://commons.wikimedia.org/w/index.php?title=File:Typical_cnn.png&oldid=535187198. – [Online; Stand 5. April 2021]

[52] Wongta, P. ; Chalidabhongse, T. H.: Vision-Based Bus Route Number Reader for Visually Impaired Travelers. In: *2018 2nd International Conference on Imaging, Signal Processing and Communication (ICISPC)*, 2018, S. 64–69

[53] Zhou, X. ; Yao, C. ; Wen, H. ; Wang, Y. ; Zhou, S. ; He, W. ; Liang, J.: EAST: An Efficient and Accurate Scene Text Detector. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, S. 2644

[54] Zhou, X. ; Yao, C. ; Wen, H. ; Wang, Y. ; Zhou, S. ; He, W. ; Liang, J.: EAST: An Efficient and Accurate Scene Text Detector. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, S. 2642–2651

A Anhang

A.1 Messtabellen Fahrtrichtungserkennung

Tabelle A.1: Ergebnisse der Fahrtzielerkennung auf dem Verifikationsdatensatz

Bild	Busanzeige	Fahrtrichtungserkennung	Ok
1	Schnelsen Kalvslohtwiete	Kalvslohtwiete	x
2	Friedrichsgabe Tycho-Brahe-Kehre	Friedrichsgabe Tycho-Brahe-Kehre	x
3	Friedrichsgabe Tycho-Brahe-Kehre	Friedrichsgabe Tycho-Brahe-Kehre	x
4	Friedrichsgabe Tycho-Brahe-Kehre	Friedrichsgabe Tycho-Brahe-Kehre	x
5	Friedrichsgabe Tycho-Brahe-Kehre	Friedrichsgabe Tycho-Brahe-Kehre	x
6	U A Norderstedt Mitte	Jehrden	
7	U A Norderstedt Mitte		
8	U A Norderstedt Mitte	U A Norderstedt Mitte	x
9	U A Norderstedt Mitte	U A Norderstedt Mitte	x
10	U A Norderstedt Mitte	U A Norderstedt Mitte	x
11	U A Norderstedt Mitte		
12	U A Norderstedt Mitte	Zeven Bahnhof Süd	
13	Harkshörn Oststraße	Harkshörn	
14	Harkshörn Oststraße	Harkshörn Oststraße	x
15	Harkshörn Oststraße	Harkshörn Oststraße	x
16	Harkshörn Oststraße	Harkshörn Oststraße	x
17	A Quickborn	A Quickborn	x
18	A Quickborn	A Quickborn	x
19	A Quickborn	A Quickborn	x
20	Glashütte Markt über Tangstedt	Glashütte Markt	x
21	Glashütte Markt über Tangstedt	Garstedt	
22	Schulau Fähranleger über S Wedel		

23	Schulau Fähranleger über S Wedel	Schulau Fähranleger	x
24	U Garstedt über ARRIBA-Bad	U Berne	
25	Norderstedt Glashütte Markt	Norderstedt Mitte	
26	Norderstedt Glashütte Markt	Glashütte Markt	x
27	Harksheide Schulweg	Harksheide Schulweg	x
28	Harksheide Schulweg	Harksheide Schulweg	x
29	Schnellbus Lufthansa-Basis	Schnellbus Lufthansa-Basis	x
30	Schnellbus Lufthansa-Basis	Schnellbus Lufthansa-Basis	x
31	Schnellbus Lufthansa-Basis	Schnellbus Lufthansa-Basis	x
32	Schnellbus Lufthansa-Basis	Schnellbus Lufthansa-Basis	x
33	Hauptbahnhof ZOB	S Reeperbahn	
34	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
35	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
36	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
37	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
38	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
39	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
40	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
41	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
42	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
43	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
44	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
45	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
46	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
47	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
48	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
49	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x
50	Hauptbahnhof ZOB	Hauptbahnhof ZOB	x

Bei den Bildern 1, 25 und 26 ist jeweils ein Zusatz vor dem eigentlichen Ziel, welcher in der HVV-Rohdaten nicht enthalten ist. Dies kann in einem Fall nicht kompensiert werden. Insgesamt werden 40 der 50 Fahrtziele richtig Klassifiziert.

Tabelle A.2: Ergebnisse der Busnummererkennung auf dem Verifikationsdatensatz

Bild	Liniennummer	Erkennung	Ok	Bild	Liniennummer	Erkennung	Ok
1	183	183	x	26	493	63	
2	394	354		27	494		
3	394	3		28	494	494	x
4	394	394	x	29	34		
5	394	394	x	30	34		
6	194	194	x	31	34	34	x
7	194	194	x	32	34	34	x
8	194	194	x	33	5	5	x
9	194	194	x	34	5	5	x
10	194	194	x	35	5	5	x
11	194	194	x	36	5	5	x
12	194			37	5	5	x
13	393	3		38	5		
14	393	3		39	5	5	x
15	393	393	x	40	5	5	x
16	393			41	5	5	x
17	194	84		42	5	5	x
18	194			43	5	5	x
19	194	4		44	109	109	x
20	378	8		45	109	109	x
21	378	38		46	109	109	x
22	594			47	109	109	x
23	594	594	x	48	109	109	x
24	393	393	x	49	109	109	x
25	493			50	109	109	x

Bei den Busnummern können 32 der 50 Anzeigen richtig erkannt werden.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort Datum Unterschrift im Original