KI-gestützter Prototyp zur automatisierten Erkennung und Interpretation von Symbolen und Daten in PDF-basierten Gleisplänen

Masterarbeit

Utkarsh Swain
Matrikel-Nr.: 542847

Gutachter:	Prof. Dr.-Ing. David Inkermann (Institut für Maschinenwesen)
2. Gutachter:	Prof. Dr.-Ing. Armin Lohrengel (Institut für Maschinenwesen)
Betreuer:	M. Sc. Thomas Schumacher (Institut für Maschinenwesen)
Betreuer in der Firma:	Dipl.-Ing. Tobias Wolff (Siemens Mobility GmbH)

Institut für Maschinenwesen
Technische Universität Clausthal

5. Februar 2026

Sperrvermerk Masterarbeit

Die vorliegende Masterarbeit mit dem Titel

> *„KI-gestützter Prototyp zur automatisierten Erkennung und Interpretation von Symbolen und Daten in PDF-basierten Gleisplänen“*

enthält interne vertrauliche Daten der Siemens Mobility GmbH.

Sie ist nur den Erst- und Zweitgutachtern sowie ggf. dem Prüfungsausschussvorsitzenden des Fachbereiches Maschinenbau zugänglich zu machen.

Veröffentlichungen und Vervielfältigungen der Masterarbeit, oder die Weitergabe der Masterarbeit – im Gesamten oder in Teilen sowie das Anfertigen von Kopien oder Abschriften – auch in digitaler Form – sind grundsätzlich untersagt.

Ausnahmen bedürfen der vorherigen schriftlichen Genehmigung der Siemens Mobility GmbH.

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides Statt, dass ich die bei der Technischen Universität Clausthal eingereichte Forschungsarbeit selbständig und ohne unerlaubte Hilfe angefertigt habe. Die benutzten Hilfsmittel sind vollständig angegeben.

_____

Datum und Unterschrift

Des Weiteren erkläre ich mich **nicht** damit einverstanden, dass meine Forschungsarbeit in der Instituts- und/oder der Universitätsbibliothek ausgelegt und aufbewahrt werden darf.

_____

Datum und Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Formelzeichen- und Abkürzungsverzeichnis

	Formelzeichen
A	Menge (z. B. Symbolmenge Version A)
α	Skalierungsfaktor für Padding
B	Menge (z. B. Symbolmenge Version B) oder Bounding Box
β	Skalierungsfaktor für Padding
$B_{\textbf{gt}}$	Ground Truth Bounding Box
$B_{\textbf{pred}}$	Vorhergesagte Bounding Box
c	Konfidenzwert (Wahrscheinlichkeit)
c_x, c_y	Zentrumskoordinaten einer Bounding Box
d	Euklidische Distanz
$\vec{d}$	Distanzvektor
d_k	Dimensionalität der Key-Vektoren
$d_{\textbf{Lev}}$	Levenshtein-Distanz
δ	Abweichungsschwellenwert
Δ	Differenzmenge oder Versatz
dx, dy	Distanzkomponenten
ϵ	Toleranzwert
e_x, e_y	Expansionsfaktoren für Bounding Boxes
f	Dateiname oder Feldbezeichner
γ	Normalisierte Konfidenz (OCR)
h	Höhe (Bounding Box oder Objekt)
h_c	Resultierende Zeichenhöhe in Pixeln
$\mathbf{H}$	Homographie-Matrix
H	Gesamthöhe des Bildes
k	Verhältnisfaktor (Schrifthöhe)
k_{dx}, k_{dy}	Klassenspezifische Multiplikatoren
K	Key-Matrix (Attention)

$\mathcal{L}$	Verlustfunktion (Loss)
λ	Gewichtungsfaktor in der Verlustfunktion
μ	Erwartungswert / Mittelwert
$\mathbf{M}$	Transformationsmatrix
$\mathbf{M}_{\textbf{affin}}$	Affine Rotationsmatrix
N	Anzahl (z. B. Objekte, Klassen)
O	Überlappung (Overlap)
p	Padding
P	Wahrscheinlichkeit
π	Pfad im CTC-Algorithmus
Q	Query-Matrix (Attention)
r	Auflösung oder Radius
$\mathbf{R}$	Rotationsmatrix
s	Schriftgröße oder Scrollposition
S	Schrittweite (Stride) oder Score
σ	Standardabweichung
t	Zeitindex oder Zeitdauer
τ	Schwellenwert (Threshold)
θ	Rotationswinkel
t_x, t_y	Translationskomponenten
T	Länge einer Sequenz oder Tile-Größe
v	Wert eines Datenfeldes
V	Value-Matrix (Attention)
w	Breite (Bounding Box oder Objekt)
W	Gesamtbreite des Bildes
x, y	Kartesische Koordinaten
z	Zoomfaktor
$\oplus$	Konkatenation
$\cap, \cup, \setminus$	Mengenoperationen

Abkürzungen

AABB	Axis-Aligned Bounding Box
ACID	Atomicity, Consistency, Isolation, Durability
AGPL	Affero General Public License
AP	Average Precision
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
AWS	Amazon Web Services
BOM	Bill of Materials
BSD	Berkeley Software Distribution
BYTEA	Byte Array
CAD	Computer-Aided Design
CER	Character Error Rate
CLI	Command Line Interface
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
CRNN	Convolutional Recurrent Neural Network
CSV	Comma-Separated Values
CTC	Connectionist Temporal Classification
CVAT	Computer Vision Annotation Tool
DB	Differentiable Binarization
DETR	Detection Transformer
DFL	Distribution Focal Loss
DIN	Deutsches Institut für Normung
DINOv2	Self-supervised Vision Transformer
DPI	Dots Per Inch
EN	Europäische Norm
ERP	Enterprise Resource Planning

FA Funktionale Anforderung

FDA Food and Drug Administration

FEM Finite Element Method

FN False Negative

FP False Positive

FPN Feature Pyramid Network

FPS Frames Per Second

GIN Generalized Inverted Index

GKS Gleiskoppelspule

GM Gleismagnet

GmbH Gesellschaft mit beschränkter Haftung

GPU Graphics Processing Unit

GUI Graphical User Interface

HBB Horizontal Bounding Box

HMM Hidden Markov Model

HOG Histogram of Oriented Gradients

ID Identifier

IO Input/Output

IoU Intersection over Union

ISO International Organization for Standardization

JPEG Joint Photographic Experts Group

JPG Joint Photographic Experts Group

JS JavaScript

JSON JavaScript Object Notation

JSONB JSON Binary

KI Künstliche Intelligenz

km Kilometer

LCAD Legacy Computer-Aided Design

LOC Lines of Code

LSTM	Long Short-Term Memory
mAP	mean Average Precision
MIT	Massachusetts Institute of Technology License
ML	Machine Learning
MVC	Model-View-Controller
NASA-TLX	NASA Task Load Index
NFA	Nicht-funktionale Anforderung
NMS	Non-Maximum Suppression
OBB	Oriented Bounding Box
OCR	Optical Character Recognition
OOM	Out of Memory
OpenGL	Open Graphics Library
ORDBMS	Object-Relational Database Management System
ORM	Object-Relational Mapping
PANet	Path Aggregation Network
PDF	Portable Document Format
PIL	Python Imaging Library
PNG	Portable Network Graphics
PSM	Page Segmentation Mode
PyQt	Python bindings for Qt
QA	Quality Assurance
Randl	Rohrleitungs- und Instrumentenfließschema
RAM	Random Access Memory
RGB	Red Green Blue
RNN	Recurrent Neural Network
ROI	Region of Interest
RPN	Region Proposal Network
RTMDet	Real-time Object Detection
SGD	Stochastic Gradient Descent

SPS	Speicherprogrammierbare Steuerung
SQL	Structured Query Language
STN	Spatial Transformer Network
SUS	System Usability Scale
SVM	Support Vector Machine
TCP	Tool Center Point
TP	True Positive
UCS	User Coordinate System
UI	User Interface
UID	Unique Identifier
UPSERT	Update or Insert
UX	User Experience
VDI	Verein Deutscher Ingenieure
ViT	Vision Transformer
VRAM	Video Random Access Memory
WER	Word Error Rate
XLSX	Excel Open XML Spreadsheet
XML	Extensible Markup Language
XSS	Cross-Site Scripting
YOLO	You Only Look Once

1. Abstract

Railway infrastructure planning relies heavily on technical layout documents in PDF or image format, which contain complex arrangements of symbols, text annotations, and positional information. Manual extraction and interpretation of this information is time-consuming, error-prone, and difficult to scale across projects and customers. This thesis presents the design and prototypical implementation of an intelligent, modular pipeline for automated analysis of railway track layouts (Gleispläne), developed in cooperation with Siemens Mobility GmbH. The system combines deep learning-based object detection using YOLOv8 with Oriented Bounding Boxes (OBB) for rotation-invariant symbol recognition, a multi-engine OCR cascade (PaddleOCR, Tesseract, EasyOCR) with orientation-adaptive preprocessing for robust text extraction, and an adaptive spatial linking algorithm that automatically associates detected symbols with their textual labels. Rule-based logic determines spatial relationships such as driving direction (Fahrtrichtung) and element groupings. All extracted objects and metadata are persisted in a PostgreSQL backend with JSONB storage, versioned records, and change tracking for full traceability between layout versions. The solution is evaluated on seven production A0 layouts from Siemens Mobility projects, achieving **98.76% end-to-end accuracy** for core symbol classes (signals, GKS plates, GM blocks), integrating detection, OCR, and spatial linking in a unified evaluation metric. The system reduces manual processing time by **75.3%**, from an average of 64 minutes to approximately 16 minutes per layout. A comprehensive PyQt5-based desktop application provides interactive visualization, manual correction tools, and validation overlays for human-in-the-loop quality assurance. This work contributes to the digitalization of railway infrastructure planning by enabling scalable, automated layout analysis that significantly reduces manual effort while improving data integrity and decision-making efficiency in safety-critical railway systems.

Keywords: Deep Learning, Object Detection, YOLOv8-OBB, Optical Character Recognition, Railway Infrastructure, Document Analysis, Automated Data Extraction

2. Einleitung

Die Digitalisierung revolutioniert zunehmend die traditionellen Arbeitsprozesse in der Industrie und eröffnet neue Möglichkeiten zur Steigerung der Effizienz, Verbesserung der Qualität und Senkung der Kosten. In vielen Bereichen des Ingenieurwesens, vom Maschinenbau über die Verfahrenstechnik bis hin zur Infrastrukturplanung, besteht ein signifikantes Potenzial für digitale Transformationsprozesse, insbesondere bei der Verarbeitung und Interpretation technischer Pläne und Zeichnungen. Die vorliegende Masterarbeit, die innerhalb der Siemens Mobility GmbH im Bereich der Bahninfrastruktur durchgeführt wurde, adressiert diese branchenübergreifende Herausforderung durch die Entwicklung eines KI-unterstützten Systems zur automatischen Verarbeitung technischer Pläne am konkreten Anwendungsfall der Gleispläne.

Im spezifischen Kontext der Schieneninfrastruktur bilden Stellwerke das Herzstück moderner Bahnsysteme und gewährleisten durch ihre komplexe Steuerungstechnik den sicheren und effizienten Zugverkehr. Die präzise Dokumentation ihrer Komponenten, von Signalen über Weichen bis hin zu Gleiskoppelspulen, ist dabei von essentieller Bedeutung für Planung, Wartung und Modernisierung. Ähnlich wie bei Hydraulikschaltplänen im Maschinenbau oder R&I-Diagrammen in der Verfahrenstechnik erfolgt die Übertragung dieser Informationen von technischen Zeichnungen in strukturierte Datenformate jedoch weitgehend manuell, was sowohl zeit- als auch ressourcenintensiv ist [1].

Die fortschreitende Entwicklung in den Bereichen Computer Vision, Machine Learning und Prozessautomatisierung eröffnet jetzt weitere Perspektiven für die Optimierung dieser Arbeitsabläufe. Diese technologischen Möglichkeiten bilden die Grundlage für den in dieser Arbeit verfolgten Ansatz, der darauf abzielt, den Prozess der Datenextraktion und -verwaltung grundlegend zu modernisieren.

Ziel dieser Arbeit ist es, theoretische Konzepte der künstlichen Intelligenz mit den praktischen Anforderungen der technischen Planverarbeitung zu verknüpfen. Durch die Entwicklung und Validierung eines KI-gestützten Ansatzes am Beispiel der Schienenverkehrstechnik wird demonstriert, wie die Lücke zwischen analogen Planungsunterlagen und digitalen Datenmodellen geschlossen werden kann.

2.1 Motivation

Die Digitalisierung von technischen Planungsprozessen stellt eine zentrale Herausforderung für Unternehmen im Infrastruktursektor und im allgemeinen Maschinenbau dar. Der Übergang von papierbasierter oder halbdigitaler Dokumentation zu durchgängig digitalen Workflows verspricht erhebliche Effizienzsteigerungen und bildet die Grundlage für moderne Konzepte wie den Digitalen Zwilling (ein virtuelles Abbild physischer Systeme zur Simulation und Analyse) oder

modellbasierte Systementwicklung.

Im Bereich der Eisenbahnsignaltechnik beruhen viele Arbeitsschritte noch auf der manuellen Auswertung von Gleisplänen, in denen eine Vielzahl von Symbolen, Textbausteinen und kundenindividuellen Darstellungen enthalten ist. Neben dem hohen Arbeitsaufwand erschwert diese Vorgehensweise die systematische Versionierung und Rückverfolgbarkeit von Änderungen über den gesamten Projektlebenszyklus [2].

Ziel der vorliegenden Masterarbeit ist es daher, eine intelligente und automatisierte Lösung zu entwickeln, die durch den Einsatz moderner Deep-Learning-Verfahren eine zuverlässige Erkennung und Interpretation von Symbolen und Text in technischen Plänen ermöglicht. In Kombination mit regelbasierter Logik sowie strukturierten Ausgabeformaten entsteht ein skalierbarer Prototyp zur Verarbeitung, Auswertung und semantischen Interpretation von Gleisplänen.

Die Umsetzung einer solchen Lösung verspricht nicht nur eine erhebliche Effizienzsteigerung in der Planungs- und Prüfphase, sondern schafft auch eine fundierte Grundlage für Rückverfolgbarkeit, Änderungsmanagement und automatisierte Konsistenzprüfungen. Damit leistet die Arbeit einen praxisnahen Beitrag zur digitalen Transformation innerhalb der Siemens Mobility GmbH und zeigt exemplarisch auf, wie moderne Verfahren der Computer Vision konkret auf industrielle Anwendungsfälle im Ingenieurwesen übertragen werden können.

Praxisbeispiel bei Siemens Mobility
Die Relevanz dieser Problemstellung zeigt sich konkret am Beispiel der Siemens Mobility GmbH:

- **Aktueller Aufwand**: 1,5-2 Stunden manuelle Datenübertragung pro Fahrstraße im 4-Augen-Prinzip
- **Fehlerquellen**: Zahlendreher, Positionsverwechslungen werden oft erst nach Testfahrten erkannt $\rightarrow$ kostenintensive Nacharbeiten
- **Fehlende Automatisierung**: Keine automatisierte Datenübertragung vom Gleisplan zur Prüftabelle im System verfügbar

Ziel ist eine signifikante Zeitreduktion durch KI-gestützte Automatisierung, wodurch Ingenieure nur noch KI-Ergebnisse validieren müssen statt komplett manuell zu arbeiten.

2.2 Problemstellung

In vielen technischen Disziplinen des Ingenieurwesens werden komplexe Systeme und Anlagen noch immer primär über grafische Pläne dokumentiert und kommuniziert. Ob Hydraulikschaltpläne im Maschinenbau, Rohrleitungs- und Instrumentenfließschemata (R&I-Diagramme) in der Verfahrenstechnik, elektrische Schaltpläne in der Elektrotechnik oder Gleispläne in der Bahntechnik – in all diesen Fachbereichen liegt wertvolles technisches Wissen in Form unstrukturierter Vektorgrafiken oder PDF-Dokumente vor [3]. Diese Pläne enthalten eine Vielzahl

semantisch relevanter Informationen: Komponenten werden durch standardisierte oder kundenspezifische Symbole repräsentiert, Verbindungen zwischen Elementen sind grafisch dargestellt, und technische Spezifikationen werden als Textannotationen beigefügt.

Für moderne digitale Arbeitsabläufe, etwa für die Erstellung eines Digitalen Zwillings, für automatisierte Konsistenzprüfungen oder für die Integration in Enterprise-Resource-Planning-Systeme (ERP, ein betriebswirtschaftliche Softwaresysteme zur integrierten Planung und Steuerung von Ressourcen, Prozessen und Daten über alle Unternehmensbereiche hinweg), müssen diese visuell kodierten Informationen jedoch in strukturierte, maschinenlesbare Datenformate überführt werden. Dieser Medienbruch zwischen rein visuellen oder unstrukturierten Plandarstellungen und semantischen Datenmodellen stellt eine zentrale Herausforderung dar, da die manuelle Übertragung nicht nur zeitaufwendig und fehleranfällig ist, sondern auch kaum skalierbar bleibt, wenn die Komplexität der Systeme oder die Anzahl der zu verarbeitenden Pläne zunimmt [2].

Erschwerend wirkt in der Praxis die hohe Varianz der Plandarstellungen: Unterschiedliche Normen und Hersteller führen zu abweichenden Symbolbibliotheken und heterogenen Layouts. Die Zuordnung von Texten zu Grafiksymbolen folgt dabei oft impliziten Konventionen, die nicht formalisiert sind. Ein weiteres Defizit traditioneller Workflows ist das Fehlen einer bidirektionalen Verknüpfung zwischen den strukturierten Daten und ihrer grafischen Entsprechung im Quelldokument. So lässt sich beispielsweise von einer Tabellenzeile (z. B. Signal AS102) nicht direkt zur exakten Position im Plan navigieren. Diese mangelnde Rückverfolgbarkeit behindert die Validierung massiv, da Ingenieure gezwungen sind, Einträge händisch zu suchen. Auch das Übertragen von Änderungen in Zielsysteme (Excel, Datenbanken, ERP) erfordert hohen Aufwand und birgt durch die fehlende Synchronisierung ein erhebliches Risiko für Dateninkonsistenzen.

Anwendungsfall Bahninfrastrukturplanung

Im konkreten Kontext der vorliegenden Arbeit manifestiert sich diese allgemeine Problemstellung im Bereich der Bahninfrastrukturplanung. Hier werden Gleispläne häufig als PDF- oder Bilddateien bereitgestellt, die Informationen über Stellwerke, Signale, Weichen, Gleiskoppelspulen und Positionsangaben enthalten. Diese grafischen Pläne müssen regelmäßig in strukturierte Datenformate wie Excel, XML oder Datenbanken überführt werden – etwa für Prüfroutinen, Dokumentation oder die Weiterverarbeitung in Planungssystemen.

Aktuell erfolgt dieser Prozess meist manuell, was arbeitsintensiv und fehlerträchtig ist und mit zunehmender Gleisplankomplexität kaum skalierbar bleibt [2]. Typische Gleispläne enthalten 50-200 verschiedene Objekte, die jeweils mit Positionsangaben, Bezeichnungen und technischen Parametern annotiert sind. Die Fehlerquote bei der manuellen Übertragung ist erheblich: Zahlendreher und Positionsverwechslungen werden oft erst nach kostspieligen Testfahrten erkannt. Zudem fehlt die systematische Rückverfolgbarkeit zwischen extrahierten Daten und ihrer Darstellung im Plan, und Änderungen in neuen Planversionen müssen mühsam manuell identifiziert und nachgepflegt werden.

Die zu lösende Herausforderung besteht darin:

- möglichst viele relevante Informationen automatisiert aus unstrukturierten Plänen zu extrahieren,
- die extrahierten Daten in maschinenlesbare Formate zu überführen,
- erkannte Symbole mit ihren zugehörigen Textinformationen semantisch zu verknüpfen,
- Rückverfolgbarkeit zwischen Plan und extrahierten Daten sowie Änderungsverfolgung zwischen Planversionen zu ermöglichen,
- sowie eine benutzerfreundliche Oberfläche für die Validierung und Nachbearbeitung bereitzustellen.

2.3 Zielsetzung der Arbeit

Ziel dieser Masterarbeit ist die Entwicklung eines skalierbaren Prototyps zur automatisierten Erkennung, Interpretation und strukturierten Verarbeitung von Gleisplänen. Im Fokus steht dabei die Extraktion von symbolischen und textlichen Inhalten aus Plänen mittels Deep-Learning-basierter Objekt- und Texterkennung. Der entwickelte Prototyp soll dabei so generisch gestaltet werden, dass er prinzipiell auch auf andere technische Zeichnungen (z.B. im Maschinenbau) übertragbar wäre.

Konkret sollen folgende Teilziele erreicht werden:

1. **Automatisierte Symbolerkennung**
   Einsatz eines geeigneten Deep-Learning-Modells, um relevante Symbole (Signale, GKS-Platten, GM-Blöcke, Koordinaten usw.) präzise in Gleisplänen zu detektieren, unabhängig von Orientierung und Planvarianz.
2. **Texterkennung**
   Durchführung einer Textextraktion innerhalb oder neben den erkannten Symbolbereichen, um wichtige Informationen zu extrahieren, einschließlich Vorverarbeitung bei rotierten oder überlagerten Beschriftungen.
3. **Strukturierte Datenzuordnung**
   Abbildung der erkannten Objekte und Texte auf semantische Bedeutungen mittels räumlicher Verknüpfungsalgorithmen und regelbasierter Logik.
4. **Export in strukturierte Formate**
   Ausgabe der extrahierten und interpretierten Informationen in maschinenlesbare Zielformate wie Excel für nachgelagerte Planungsprozesse.
5. **Rückverfolgbarkeit & Änderungsverfolgung**
   Entwicklung von Mechanismen zur Verknüpfung zwischen Bilddaten (PDF) und extrahier-

ten Datenpunkten sowie zur automatisierten Erkennung von Änderungen in unterschiedlichen Planversionen.

6. **Benutzeroberfläche**
   Aufbau einer interaktiven Benutzeroberfläche zur Ergebnisdarstellung mit Visualisierung der erkannten Symbole, der zugehörigen Daten und Änderungen sowie der Nachverfolgbarkeit zwischen Daten und Symbolen.

Durch die Umsetzung dieser Ziele soll ein robuster, praxisnaher Demonstrator entstehen, der mit realen Kundendaten anwendbar ist, reproduzierbare Ergebnisse liefert und Potenzial für eine weiterführende Integration in bestehende Prozesse bietet.

2.4 Forschungsfragen

Abgeleitet aus der Problemstellung und den Zielsetzungen beschäftigt sich diese Arbeit mit der folgenden zentralen Forschungsfrage:

HF: Wie lässt sich der Prozess der Datenextraktion aus heterogenen technischen Zeichnungen durch den Einsatz von Deep Learning und hybriden Verarbeitungsstrategien automatisieren, um eine valide Überführung in strukturierte Datenmodelle zu gewährleisten?

Diese Hauptfrage wird in der vorliegenden Arbeit exemplarisch am Anwendungsfall der Gleispläne in der Bahninfrastruktur untersucht. Zur Beantwortung werden folgende Teilforschungsfragen (TF) adressiert:

- **TF1:** Inwieweit eignen sich aktuelle einstufige Objektdetektoren zur zuverlässigen Erkennung von kleinteiligen, rotierten Symbolen in technischen Zeichnungen und welche Vorverarbeitungsschritte sind notwendig, um die Präzision bei variierenden Layouts zu maximieren?

- **TF2:** Wie können geometrische Informationen und unstrukturierte Textdaten algorithmisch so verknüpft werden, dass eine korrekte semantische Zuordnung (z. B. Signalbezeichnung zu Signalsymbol) auch bei hoher Objektdichte erfolgt?

- **TF3:** Wie muss eine Systemarchitektur gestaltet sein, um neue Symbolklassen und Datenformate modular zu integrieren, wobei Änderungen auf einzelne Pipeline-Komponenten beschränkt bleiben?

- **TF4:** Welcher algorithmische Ansatz eignet sich, um in rein visuellen Daten semantische Änderungen zwischen zwei Planversionen robust zu identifizieren und visualisierbar zu machen?

2.5 Forschungsvorgehen

Zur Beantwortung der formulierten Forschungsfragen wurde ein methodischer Ansatz gewählt, der sich aus vier aufeinander aufbauenden Phasen zusammensetzt.

Phase 1: Literaturrecherche und Technologieanalyse. Zunächst wurde eine systematische Literaturrecherche in den Bereichen Objekterkennung in technischen Zeichnungen, optische Zeichenerkennung sowie branchenübergreifende Digitalisierung von Planungsunterlagen durchgeführt. Dabei wurden sowohl aktuelle Fachpublikationen aus Datenbanken wie IEEE Xplore, Springer und Google Scholar als auch die offizielle Dokumentation relevanter Open-Source-Werkzeuge (insbesondere YOLOv8, PaddleOCR, Tesseract und EasyOCR) ausgewertet. Die Ergebnisse dieser Phase bilden die Grundlage für Kapitel 3.

Phase 2: Anforderungserhebung und Domänenanalyse. Parallel zur Literaturrecherche wurden Experteninterviews mit Fachspezialisten der Siemens Mobility GmbH sowie Rücksprachen mit wissenschaftlichen Betreuern an der TU Clausthal geführt. Diese dienten der Identifikation praxisrelevanter Anforderungen, der Analyse bestehender manueller Arbeitsabläufe sowie dem Verständnis domänenspezifischer Besonderheiten der Gleisplanverarbeitung. Auf dieser Basis wurden die funktionalen und nicht-funktionalen Anforderungen systematisch abgeleitet (Kapitel 4).

Phase 3: Iterative Konzeption und Implementierung. Ausgehend von den erhobenen Anforderungen und den identifizierten technologischen Möglichkeiten wurde eine modulare Systemarchitektur konzipiert (Kapitel **??**) und prototypisch umgesetzt (Kapitel **??**). Die Entwicklung erfolgte iterativ: Einzelne Komponenten wie Objektdetektion, Texterkennung und Verknüpfungsalgorithmen wurden schrittweise implementiert, getestet und auf Basis der Ergebnisse optimiert.

Phase 4: Evaluation und Validierung. Die abschließende Evaluation umfasst sowohl komponentenbezogene Leistungskennzahlen als auch eine End-to-End-Bewertung des Gesamtsystems anhand realer Gleispläne (Kapitel **??**). Die Validierung erfolgte durch den systematischen Abgleich der automatisiert extrahierten Daten mit manuell erstellten Referenzlisten der Siemens Mobility GmbH.

2.6 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in acht Hauptkapitel:

- **Kapitel 1 (Abstract)** liefert eine kurze Zusammenfassung der Zielsetzung, Methodik und zentralen Ergebnisse der Arbeit.
- **Kapitel 2 (Einleitung)** führt in das Thema ein, erläutert die Motivation, beschreibt die zugrunde liegende Problemstellung sowie das Ziel der Arbeit.
- **Kapitel 3 (Theoretische und technische Grundlagen)** stellt die relevanten Konzepte

aus dem Bereich der grafischen Gleispläne, der Symbolik sowie der objekterkennenden Verfahren mit Deep Learning vor.

- **Kapitel 4 (Anforderungsanalyse)** identifiziert die funktionalen und nicht-funktionalen Anforderungen an das System und beschreibt die Herausforderungen in der praktischen Umsetzung.
- **Kapitel 5 (Konzeption des Prototyps)** erläutert die Architektur und den geplanten Workflow des Systems von der PDF-Konvertierung bis zur strukturierten Ausgabe.
- **Kapitel 6 (Implementierung)** beschreibt die technische Realisierung der Komponenten, inklusive des Trainings von Detektionsmodellen, der Texterkennung, des Mappings sowie der Versionierung und Rückverfolgbarkeit.
- **Kapitel 7 (Evaluation)** bewertet das System anhand verschiedener Testmethoden und diskutiert die Ergebnisse und Validierungen mit Siemens-internen Daten.
- **Kapitel 8 (Diskussion)** fasst die wichtigsten Erkenntnisse zusammen, reflektiert kritisch den Entwicklungsprozess und gibt einen Ausblick auf mögliche Weiterentwicklungen.

3. Theoretische und technische Grundlagen

Im Rahmen dieses Kapitels werden die maßgeblichen Grundlagen dargelegt, auf denen das entwickelte System basiert. Dabei umfassen die *theoretischen Grundlagen* die konzeptionellen Aspekte wie die Struktur von Gleisplänen, die Klassifikation bahntechnischer Symbole sowie die Prinzipien der Objekterkennung, während die *technischen Grundlagen* die eingesetzten Technologien und Werkzeuge behandeln. Der Fokus liegt auf drei Kernbereichen, die den in Kapitel 2 definierten Zielen entsprechen: (1) automatisierte Objekterkennung zur Symboldetektion, (2) Textextraktion zur Erfassung von Bezeichnungen und Positionsangaben, sowie (3) strukturierte Informationsverarbeitung zur semantischen Verknüpfung der erkannten Elemente.

3.1 Grafische Gleispläne im Bahnkontext

Die Dokumentation einer Bahnanlage stellt eine komplexe Aufgabe dar, da diese typischerweise aus einer Vielzahl von Gleisabschnitten, Weichen und Signalen besteht. Die manuelle Erfassung aller Anlagendetails erfordert üblicherweise mehrere Wochen Arbeit, was die Skalierbarkeit traditioneller Dokumentationsansätze erheblich einschränkt.[4]

Diese zeitintensive manuelle Arbeit motiviert den Einsatz automatisierter Verfahren zur Datenextraktion. Die folgenden Abschnitte legen die theoretischen Grundlagen dar, auf denen eine solche Automatisierung aufbauen kann.

3.1.1 Grundlagen des Gleisplans

Der Gleisplan bildet die fundamentale Datengrundlage für die Planung, den Bau und den operativen Betrieb von Bahnanlagen. Er repräsentiert die technische Infrastruktur und dient als visuelle Schnittstelle zwischen der physischen Außenanlage und der sicherungstechnischen Logik (Stellwerk).
Gleispläne existieren in zwei Darstellungsformen [5]: maßstäbliche Lagepläne (topografisch, bautechnisch) und schematische Übersichtspläne (topologisch, sicherungstechnisch). Diese Arbeit fokussiert letztere, da sie die logischen Fahrwegbeziehungen priorisieren und die Grundlage für die Stellwerksplanung bilden.

Gleispläne enthalten drei Symbolkategorien [5]: (1) Fahrwegelemente (Gleise, Weichen), (2) Sicherungstechnik (Signale, Achszähler), (3) Metadaten (Bezeichner, Kilometrierung). Jedes Element trägt eindeutige Labels zur Identifikation.

Obwohl diese Pläne in der heutigen Ingenieurspraxis mittels CAD-Werkzeugen (Computer Aided Design) erstellt werden, liegt die primäre Herausforderung für eine automatisierte Auswertung nicht im Dateiformat, sondern in der korrekten semantischen Interpretation dieser abstrahierten Symbolik und ihrer logischen Verknüpfung [5].

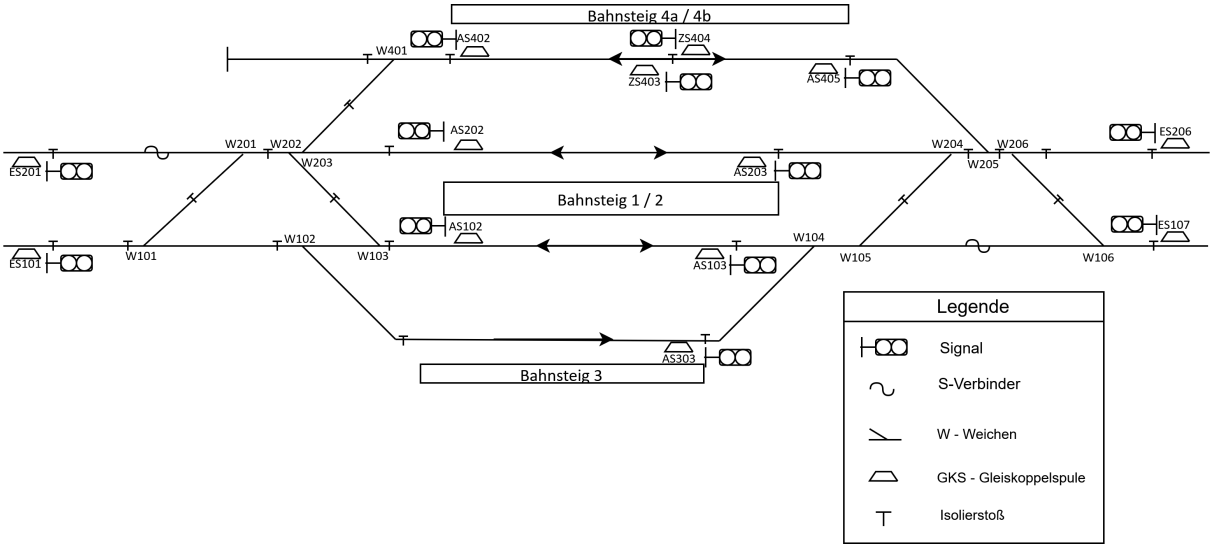


Abbildung 3.1: Ausschnitt eines schematischen Gleisplans mit drei Bahnsteigbereichen (4a/4b, 1/2, 3), wobei Richtungspfeile die zulässigen Fahrtrichtungen kennzeichnen (Beispielhafte Darstellung)

Abbildung 3.1 zeigt einen typischen Bahnhofsbereich mit drei Bahnsteigen (4a/4b, 1/2, 3) und der zugehörigen Infrastruktur. Die Darstellung umfasst verschiedene bahntechnische Symbole. **Signale** werden durch ein einheitliches Grundsymbol dargestellt und durch ihre Bezeichnung in drei Kategorien unterteilt: **Einfahrtsignale (ES)** regeln die Einfahrt in den Bahnhof, **Ausfahrtsignale (AS)** regeln die Ausfahrt, und **Zwischensignale (ZS)** regeln Fahrten innerhalb des Bahnhofsbereichs. Die **Weichen (W)** ermöglichen die flexible Fahrweggestaltung zwischen den verschiedenen Gleisen. **Gleiskoppelspulen (GKS)**, als trapezförmige Symbole dargestellt, dienen der punktförmigen Zugbeeinflussung. Die elektrische Trennung von Gleisabschnitten zur Belegungserkennung wird durch **Isolierstöße** (T-förmige Symbole) realisiert. Ergänzend sind **S-Verbinder** (S-förmige Symbole) dargestellt, die Schienenverbindungen zwischen Gleisabschnitten markieren. **Richtungspfeile** kennzeichnen die zulässigen Befahrrichtungen der einzelnen Gleise – beispielsweise zeigt die bidirektionale Pfeildarstellung bei Bahnsteig 1/2 die Befahrbarkeit aus beiden Richtungen an [5].

3.1.2 Bahntechnische Symbole und ihre Klassifikation

In den Gleisplänen kommen zahlreiche Symbole zum Einsatz, um die verschiedenen technischen Objekte, Anlagenkomponenten und zusätzlichen Informationen übersichtlich darzustellen [5]. Gleispläne dienen als fundamentale Arbeitsgrundlage für das Engineering von Stellwerken und bilden die zentrale Planungsbasis für verschiedenste Gewerke der Bahntechnik [5, 6]. Stellwerke nutzen diese Pläne beispielsweise für das Einstellen von Signalen mit Fahrstraßenelementen und Ausschlüssen, während die Zugsicherung auf Basis dieser Pläne die Betrachtung der Durchrutschwege und Fahrwegelemente durchführt [5]. Die Zugsicherungssysteme werden dabei vom Stellwerk gesteuert, entweder direkt über die Stellwerkslogik oder indirekt über nachgelagerte Systeme [5]. Alle dargestellten Elemente wie Signale, Weichen,

Isolierstöße, Achszähler, Gleiskoppelspulen, Gleismagnete und S-Verbinder sind Bestandteile der bahntechnischen Infrastruktur, wobei jedes Gewerk abhängig von seiner spezifischen Aufgabenstellung unterschiedliche Teilmengen dieser Elemente benötigt [5]. Da der Gleisplan somit eine zentrale Schnittstelle zwischen technischer Planung, betrieblicher Umsetzung und sicherheitstechnischen Systemen darstellt, ist eine präzise, klare und einheitliche Symbolklassifizierung notwendig [7]. Die für diese Arbeit relevanten Symbolklassen gliedern sich in vier funktionale Kategorien (siehe Tabelle 3.1):

Kategorie	**Elemente und Funktion**
Stellwerkselemente	**Signale**: Fahrterlaubnis und Geschwindigkeitsvorgabe[5]. **Weichen**: Fahrwegverzweigungen[5].
Zugbeeinflussung	**Balisen**, **Gleismagnete (GM)**, **Gleiskoppelspulen (GKS)**: Übertragung von Informationen (Fahrt-/Haltinformationen, ggf. auch Geschwindigkeitsvorgaben und weitere Daten) vom Stellwerk an den Zug[5].
Belegungserkennung	**Isolierstöße**: Abgrenzung von Gleisabschnitten zur Belegungserkennung mittels Gleisstromkreisen[5]. **Achszähler**: Zählpunktbasierte Belegungserkennung durch Erfassung ein- und ausfahrender Achsen[5].
Metadaten	**Positionsangaben**: Kilometrierung, Lagepunkte[6]. **Gleisabschnittsnamen**: Eindeutige Gleisidentifikation[6].

Tabelle 3.1: Klassifikation bahntechnischer Symbole nach funktionalen Kategorien

Die automatisierte Analyse visueller Informationen bildet das Fundament zahlreicher technischer Anwendungen von der Qualitätskontrolle in der industriellen Fertigung über die medizinische Bildanalyse bis hin zur autonomen Navigation. Während Menschen mühelos in der Lage sind, Objekte in Bildern zu identifizieren und zu lokalisieren, erfordert die Replikation dieser Fähigkeit in Maschinen die Lösung komplexer Teilprobleme. Historisch wurde diese Aufgabe durch explizite Programmierung von Erkennungsregeln adressiert. Der konzeptionelle Durchbruch gelang durch maschinelles Lernen, insbesondere durch tiefe neuronale Netze, die relevante Merkmale selbstständig aus Trainingsdaten extrahieren können [8, 9]. Diese Methodik bildet das Fundament für die in dieser Arbeit entwickelte Symbol- und Texterkennung in Gleisplänen.

Für technische Anwendungen wie die Analyse von Konstruktionszeichnungen oder Gleisplänen manifestieren sich spezifische Herausforderungen, die über die Verarbeitung natürlicher Fotografien hinausgehen. Im Folgenden werden zunächst die besonderen Eigenschaften technischer CAD-Zeichnungen wie hohe Objektdichte, standardisierte Symbolik und beliebige Symbolorientierung erläutert, bevor die grundlegenden methodischen Ansätze der Objekterkennung systematisch hergeleitet werden. Das Kapitel folgt dabei einer Progression vom Allgemeinen zum Spezifischen: Ausgehend von universellen Erkennungsprinzipien wird schrittweise die Anwendung auf bahntechnische Gleispläne konkretisiert.

3.1.3 Objekterkennung in technischen CAD-Zeichnungen

Im Gegensatz zu natürlichen Szenen weisen technische Konstruktionszeichnungen spezifische Charakteristika auf, die dedizierte Lösungsansätze erfordern. Diese domänenspezifischen Besonderheiten haben direkten Einfluss auf die Wahl geeigneter Erkennungsverfahren und definieren die in Kapitel 4 formulierten funktionalen Anforderungen an das System.

Domänenspezifische Eigenschaften

Technische Zeichnungen unterscheiden sich fundamental von natürlichen Bildern in ihrer Entstehung und Struktur [10]. CAD-Systeme erzeugen vektorbasierte Repräsentationen, die geometrische Primitive wie Linien, Kreise und Polylinien sowie zugeordnete Metadaten enthalten. Die Konvertierung in Rasterformate (PDF, PNG) für ML-basierte Analyse führt jedoch zu einem Informationsverlust hinsichtlich topologischer Beziehungen und semantischer Layer-Informationen. Topologische Beziehungen beschreiben die Verbindungsstruktur zwischen Elementen – etwa welche Gleisabschnitte durch eine Weiche verbunden sind oder welches Signal zu welchem Gleis gehört. Semantische Layer-Informationen kodieren die funktionale Zuordnung von Objekten zu Kategorien wie Gleisführung, Signaltechnik oder Oberleitung. Bei der Rasterisierung werden diese Strukturinformationen eliminiert; das Bild wird zu einer Ansammlung von Pixeln ohne inhärente Bedeutung. Diese Transformation von strukturierten zu unstrukturierten Daten motiviert den Einsatz von Deep-Learning-Verfahren, die robuste Mustererkennung auch ohne explizite topologische Information ermöglichen.

Bahntechnische Symbole folgen standardisierten Bibliotheken gemäß EN-Normen [11] oder kundenspezifischen Richtlinien. Die Variabilität innerhalb einer Symbolklasse ist somit deutlich geringer als bei natürlichen Objekten wie Fahrzeugen oder Personen, jedoch erschweren kundenspezifische Anpassungen die Generalisierung über verschiedene Planungsunternehmen hinweg. Ein durchschnittlicher Bahnhofsgleisplan enthält zwischen 100 und 300 relevante Symbole (Signale, Zugbeeinflussungselemente, Koordinaten) auf einer Fläche von wenigen Quadratmetern, was eine deutlich höhere Objektdichte als in typischen Computer-Vision-Benchmark-Datensätzen wie COCO [12] oder Pascal VOC [13] darstellt. Zum Vergleich: COCO-Bilder enthalten im Durchschnitt 7 bis 10 Objekte, während ein Gleisplanausschnitt von vergleichbarer Pixelgröße 50 bis 100 Symbole aufweisen kann.

Anders als bei Objekten mit kanonischer Orientierung – etwa Straßenverkehrsschilder (typischerweise frontal) oder Fußgänger (vertikal stehend) – können bahntechnische Symbole und Textannotationen entlang der Gleisachsen in beliebigen Winkeln orientiert sein. Ein Weichensymbol kann beispielsweise in 16 verschiedenen Rotationen auftreten, die jeweils $22,5°$ Schritte zwischen $0°$ und $360°$ abdecken. Signale werden parallel zur Gleisachse positioniert, Gleisnummern folgen der Krümmung des Gleises. Diese Rotationsvariabilität ohne bevorzugte Ausrichtung erfordert rotationsinvariante Detektionsarchitekturen, deren Notwendigkeit in Abschnitt 3.2 detailliert erläutert wird.

Stand der Forschung

Die Anwendung von Machine Learning auf technische Zeichnungen ist ein aktives Forschungsfeld mit Anwendungen in verschiedenen Ingenieurdisziplinen. Ahmed et al. [14] entwickelten frühe Ansätze zur automatischen Segmentierung von architektonischen Grundrissen unter Verwendung morphologischer Operationen und Hough-Transformation. Moreno-García et al. [15] entwickelten ein umfassendes Framework zur Digitalisierung komplexer technischer Zeichnungen und demonstrierten die Überlegenheit von Deep-Learning-Ansätzen gegenüber klassischen Feature-Engineering-Methoden. Rahul et al. [16] extrahierten Informationen aus P&ID-Diagrammen der Prozessindustrie mittels Faster R-CNN und erreichten dabei mean Average Precision-Werte von über 85%.

Jamieson et al. [17] liefern eine umfassende Übersicht über Deep-Learning-Methoden für Engineering-Diagramme und identifizieren als zentrale Herausforderungen die hohe Variabilität kundenspezifischer Symbolbibliotheken, den Mangel an annotierten Trainingsdaten sowie die Notwendigkeit rotationsinvarianter Detektionsverfahren. Für den bahntechnischen Bereich existieren bisher hauptsächlich proprietäre Lösungen einzelner Eisenbahnunternehmen ohne publizierte wissenschaftliche Evaluierung. Die vorliegende Arbeit adressiert diese Forschungslücke durch Entwicklung und Evaluation eines prototypischen Systems auf Basis moderner One-Stage-Detektoren (Detektoren wie YOLO, die Objekte in einem einzigen Netzwerkdurchlauf lokalisieren und klassifizieren).

3.1.4 Entwicklung der Objekterkennung

Die erwähnten Detektionsverfahren basieren auf *Deep Learning*, einer Form des maschinellen Lernens, bei der künstliche neuronale Netze – insbesondere *Convolutional Neural Networks* (CNNs) – relevante Merkmale automatisch aus Trainingsbildern lernen [18]. Im Gegensatz zu klassischen Verfahren, bei denen Erkennungsmerkmale manuell definiert werden müssen, ermöglichen CNNs die End-to-End-Optimierung des gesamten Erkennungssystems. Die folgenden Abschnitte beschreiben zunächst die historische Entwicklung dieser Technologie, bevor spezialisierte Architekturen für die Objekterkennung detailliert werden. Die Objekterkennung hat sich von regelbasierten Verfahren über klassisches maschinelles Lernen zu Deep-Learning-Ansätzen entwickelt [19]. Für die vorliegende Arbeit werden ausschließlich CNN-basierte Detektoren eingesetzt, da diese bei technischen Zeichnungen signifikant höhere Erkennungsraten erreichen als klassische Verfahren (vgl. Kapitel **??**, Tabelle **??**).

3.1.5 Architekturen für Deep-Learning-basierte Objekterkennung

Die Implementierung von CNN-basierten Objektdetektoren lässt sich in zwei Architekturparadigmen unterteilen, die sich in ihrer Balance zwischen Detektionsgenauigkeit und Recheneffizienz unterscheiden [19]. Die Wahl der Architektur hat direkte Auswirkungen auf die Praxistauglichkeit des Systems.

Zweistufige vs. einstufige Detektoren

Zweistufige Detektoren wie Faster R-CNN [20] separieren den Erkennungsprozess konzeptionell in zwei aufeinanderfolgende Phasen. Zunächst generiert ein Region Proposal Network (RPN) zwischen 100 und 300 Kandidatenregionen, die potenziell Objekte enthalten könnten. Diese Regionen werden anschließend durch ein separates Klassifikationsnetzwerk evaluiert, das sowohl die Objektklasse bestimmt als auch die Bounding-Box-Koordinaten verfeinert. Der Vorteil liegt in hoher Detektionspräzision, insbesondere bei kleinen oder teilweise verdeckten Objekten. Die sequenzielle Verarbeitung hunderter Kandidatenregionen führt jedoch zu Bildraten von lediglich 5 bis 10 Bildern pro Sekunde (Frames Per Second, FPS) auf moderner GPU-Hardware (Graphics Processing Unit).

Für die Gleisplananalyse erweist sich diese Geschwindigkeit als kritisch. Ein durchschnittlicher Plan umfasst nach der Kachelungsstrategie (siehe Abschnitt **??**) zwischen 150 und 300 Bildausschnitte. Während YOLO-Modelle typischerweise mit Eingangsgrößen von 640×640 oder 1024×1024 Pixeln arbeiten, werden für die Gleisplananalyse aufgrund der sehr kleinen Symbolgrößen Kacheln von 2048×2048 Pixeln verwendet (Details zur Kachelungsstrategie in Abschnitt **??**). Bei einer Verarbeitungszeit von 100 bis 200 Millisekunden pro Kachel würde die Gesamtanalyse zwischen 4 bis 15 Minuten dauern. Bei größeren Eingabebildern erfolgt ein automatisches Downsampling auf die konfigurierte Eingabegröße, wobei die Ausgabekoordinaten entsprechend zurückskaliert werden.

Einstufige Detektoren wie YOLO (You Only Look Once) [21] adressieren diese Geschwindigkeitsproblematik durch eine fundamental andere Strategie. YOLO unterteilt das Eingabebild in ein regelmäßiges Raster von Zellen (beispielsweise 13×13 oder 26×26 Zellen bei kleineren Eingabegrößen, bzw. entsprechend größere Raster bei höheren Auflösungen). Für jede Zelle sagt das Netz in einem einzigen Vorwärtsdurchlauf direkt mehrere Aspekte vorher: Liegt in dieser Zelle ein Objekt? Wenn ja, welcher Objektklasse gehört es an? Wie hoch ist die Konfidenz dieser Vorhersage? Und wo genau innerhalb der Zelle befindet sich das Objekt? Diese parallele Verarbeitung aller Bildbereiche ermöglicht Inferenzgeschwindigkeiten von 40 bis über 100 Bildern pro Sekunde [22]. Moderne Varianten wie YOLOv8 erreichen durch architektonische Verbesserungen – insbesondere Feature Pyramid Networks (FPN) [23] für Multi-Scale-Detektion und verbesserte Loss-Funktionen – vergleichbare oder sogar überlegene Präzision gegenüber zweistufigen Detektoren bei Beibehaltung der Echtzeitfähigkeit. FPN ermöglicht dabei die gleichzeitige Erkennung von Symbolen unterschiedlicher Größe, was für Gleispläne essentiell ist, da kleine Elemente wie Isolierstöße und große Symbole wie Signale im selben Bild detektiert werden müssen.

Die Abwägung zwischen beiden Paradigmen muss im Kontext der Anwendungsanforderungen erfolgen. Für technische Gleispläne mit gut sichtbaren, scharf gerenderten Symbolen rechtfertigt die hohe Verarbeitungsgeschwindigkeit marginale Genauigkeitseinbußen. Einstufige Architekturen können bei technischen Zeichnungen vergleichbare Präzision erreichen.

Abbildung 3.2 illustriert den fundamentalen Unterschied zwischen beiden Architekturparadigmen.

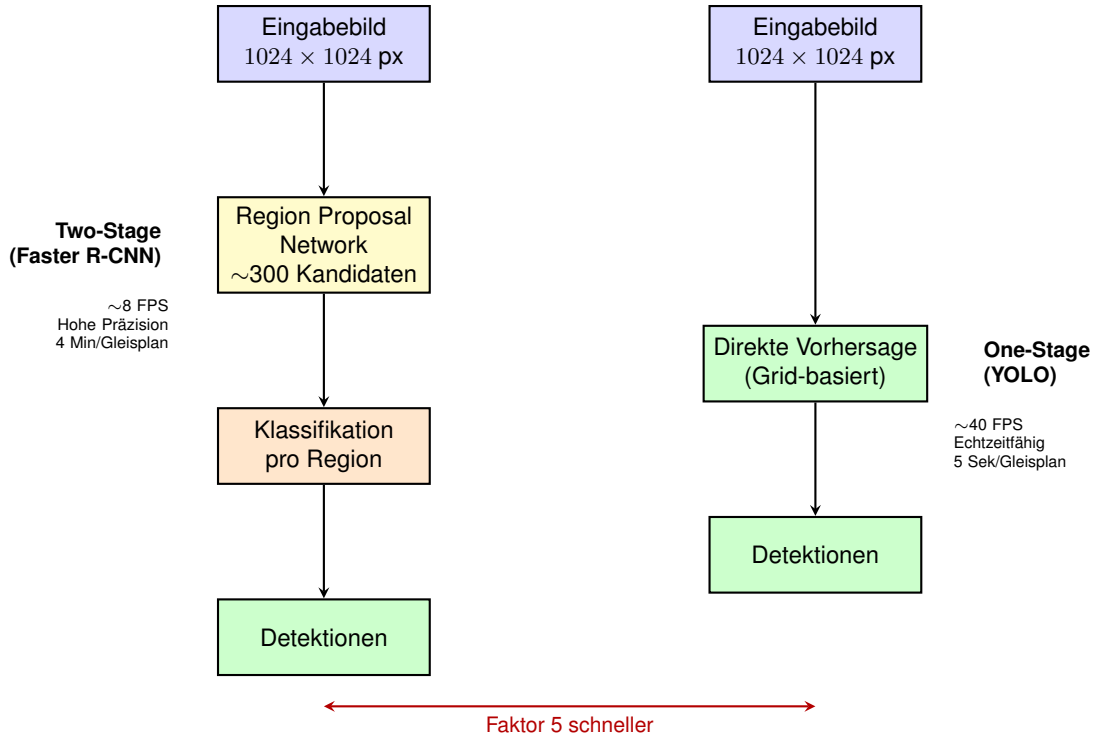


Abbildung 3.2: Architekturvergleich zwischen zweistufigen und einstufigen Objektdetektoren.

3.1.6 Modelltraining

Das Training moderner Objektdetektoren wie YOLOv8 erfolgt überwacht (supervised) mit annotierten Bounding Boxes [22]. Jedes Trainingsbeispiel besteht aus einem Eingabebild und den zugehörigen Ground-Truth-Annotationen, die Position, Größe und Klassenzugehörigkeit jedes Objekts spezifizieren.

Transfer Learning. Anstatt ein Modell von Grund auf zu trainieren, wird typischerweise mit vortrainierten Gewichten initialisiert [24]. YOLOv8-Modelle werden standardmäßig auf dem COCO-Datensatz (Common Objects in Context) mit 80 Objektklassen vortrainiert. Dieses Transfer Learning ermöglicht es, bereits erlernte low-level Features (Kanten, Texturen) auf die Zieldomäne zu übertragen und beschleunigt die Konvergenz erheblich, insbesondere bei kleineren domänenspezifischen Datensätzen wie sie für Gleispläne typisch sind.

Datenaugmentation. Zur Verbesserung der Generalisierungsfähigkeit und Vermeidung von Overfitting setzt YOLOv8 verschiedene Augmentationstechniken ein [25]:

- *Mosaic*: Kombiniert vier Trainingsbilder zu einem, wodurch das Modell lernt, Objekte in verschiedenen Kontexten zu erkennen
- *MixUp*: Überlagert zwei Bilder mit ihren Labels, was die Entscheidungsgrenzen glättet
- *Geometrische Transformationen*: Rotation, Skalierung und Spiegelung erhöhen die Varianz der Trainingsdaten

- *Farbaugmentation*: Variation von Helligkeit, Kontrast und Sättigung verbessert die Robustheit gegenüber unterschiedlichen Scanqualitäten

Die kombinierte Loss-Funktion (vgl. Abschnitt 3.2.4) optimiert simultan Lokalisierung und Klassifikation. Die konkreten Trainingsparameter und domänenspezifischen Anpassungen für die Gleisplanerkennung werden in Kapitel **??**, Abschnitt **??** dokumentiert.

3.1.7 Evaluationsmetriken

Die Bewertung von Objektdetektoren erfordert standardisierte Metriken, die sowohl Lokalisierungsgenauigkeit als auch Klassifikationsgüte quantifizieren. Im Gegensatz zu einfachen Klassifikationsproblemen müssen Detektoren beide Aspekte simultan korrekt vorhersagen. Die nachfolgenden Metriken bilden die Grundlage für die Systemevaluation.

Intersection over Union

Die Intersection over Union (IoU) [13] quantifiziert den Überlappungsgrad zwischen vorhergesagter Box B_{pred} und Ground-Truth-Box B_{gt}:

$$\text{IoU}(B_{\text{pred}}, B_{\text{gt}}) = \frac{\text{Area}(B_{\text{pred}} \cap B_{\text{gt}})}{\text{Area}(B_{\text{pred}} \cup B_{\text{gt}})} \tag{3.1}$$

Der IoU-Wert liegt im Intervall $[0, 1]$, wobei 0 keine Überlappung und 1 eine perfekte Übereinstimmung signalisiert. In der Praxis gilt eine Detektion als korrekt, wenn IoU ≥ 0.5 – das heißt, die Überlappung muss mindestens die Hälfte der Gesamtfläche betragen. Dieser Schwellenwert reflektiert einen Kompromiss zwischen Forderung nach Präzision und Toleranz gegenüber geringfügigen Positionsabweichungen. Für präzisionskritische Anwendungen werden teilweise höhere Schwellen (0.75 oder 0.9) verwendet [12].

Abbildung 3.3 veranschaulicht diese Berechnung anhand zweier kontrastierender Beispiele. In beiden Fällen repräsentiert die grüne Box die Ground-Truth-Position eines Symbols, während die blaue Box die Vorhersage des Detektors darstellt. Die Schnittfläche (rot schraffiert) entspricht der Überlappung beider Boxen.

Das linke Beispiel zeigt eine Detektion mit IoU $= 0.28$: Die Vorhersage weist zwar auf das korrekte Objekt hin, verfehlt aber die Ground Truth deutlich. Mit einem IoU-Wert unterhalb der Standardschwelle von 0.5 wird diese Detektion als False Positive gewertet. Das rechte Beispiel demonstriert den Grenzfall mit IoU $= 0.50$: Hier überlappt die Schnittfläche exakt die Hälfte der Vereinigungsfläche, was der minimalen Anforderung für eine korrekte Detektion entspricht. Der visuelle Vergleich verdeutlicht die Strenge der IoU-Metrik: Selbst wenn der Detektor das richtige Objekt identifiziert, führt eine ungenaue Lokalisierung zur Klassifikation als Fehldetektion.

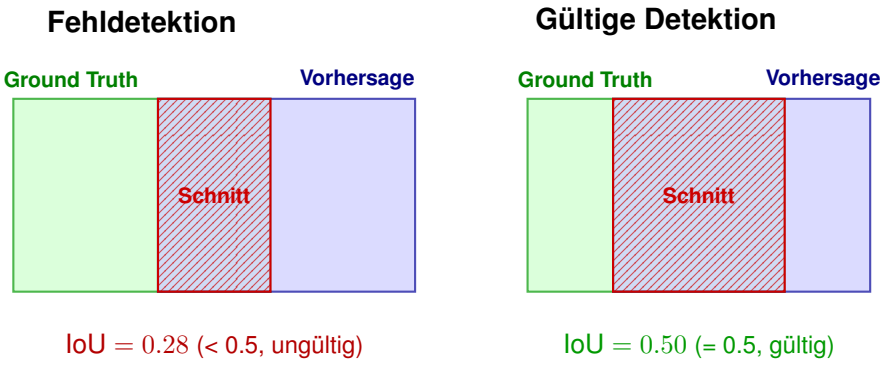


Abbildung 3.3: Visualisierung der Intersection over Union (IoU) bei unterschiedlichen Überlappungsgraden.

Precision, Recall und Mean Average Precision

Basierend auf der IoU-Schwelle werden Detektionen klassifiziert: True Positives (TP, korrekte Detektionen mit IoU ≥ 0.5), False Positives (FP, Falschdetektionen oder korrekte Klasse aber IoU < 0.5) und False Negatives (FN, übersehene Objekte). Daraus ergeben sich Precision und Recall [13]:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN} \tag{3.2}$$

Die Precision gibt an, welcher Anteil der vom Detektor gemeldeten Objekte tatsächlich korrekt ist. Ein hoher Precision-Wert bedeutet wenige Fehlalarme. Der Recall misst, welcher Anteil der im Bild tatsächlich vorhandenen Objekte vom Detektor gefunden wurde. Ein hoher Recall-Wert bedeutet, dass wenige Objekte übersehen werden. Beide Metriken stehen typischerweise in einem Trade-off: Durch Erhöhung der Konfidenzschwelle kann Precision verbessert werden, jedoch auf Kosten des Recalls.

Für Gleisplanerkennung ist insbesondere der Recall kritisch. Ein übersehenes Signal oder eine fehlende Weichenkennzeichnung kann zu gravierenden Fehlern in der nachfolgenden Planung führen – etwa falsche Berechnung von Gleiskapazitäten oder Sicherheitsrisiken. Falschdetektionen (niedrige Precision) sind weniger kritisch, da sie in der Validierungsphase (siehe Abschnitt **??**) durch regelbasierte Filter und Plausibilitätsprüfungen eliminiert werden können. Ein falsch detektiertes Signal an einer unmöglichen Position (etwa mitten auf einem Gleis ohne Haltemöglichkeit) wird durch die semantische Validierung verworfen.

Die Mean Average Precision (mAP) [13, 12] aggregiert die Erkennungsgüte über alle Objektklassen und verschiedene Konfidenzschwellen. Für jede Klasse wird zunächst die Average Precision (AP) als Fläche unter der Precision-Recall-Kurve berechnet. Diese Kurve entsteht, indem die Konfidenzschwelle variiert wird: Bei niedriger Schwelle werden viele Objekte detektiert (hoher Recall, niedrige Precision), bei hoher Schwelle nur sehr sichere Detektionen (niedriger Recall, hohe Precision). Die mAP ist der Mittelwert über alle Klassen:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i \tag{3.3}$$

wobei N die Anzahl der Objektklassen bezeichnet. Üblich ist die Angabe von mAP@0.5 (IoU-Schwelle 50%) sowie mAP@[0.5:0.95] (Mittelwert über IoU-Schwellen von 50% bis 95% in 5%-Schritten). Die letztere Metrik ist deutlich strenger und fordert präzisere Lokalisierung.

Die mAP-Metrik ist keine Anforderung, sondern die *Standardevaluationsmetrik* für Objektdetektoren wie YOLO [22]. Sie wird sowohl während des Trainings zur Überwachung des Lernfortschritts als auch zur finalen Bewertung des trainierten Modells verwendet. In der vorliegenden Arbeit dient mAP@0.5 als primäre Evaluationsmetrik für die Objekterkennung (Kapitel **??**), da für die Gleisplananalyse eine IoU-Schwelle von 50% ausreichend ist: Leichte Ungenauigkeiten in der Boxlokalisierung werden durch nachgelagerte Validierung kompensiert. Ein Signal, das mit $45°$ statt exakt $47°$ Rotation detektiert wird, ist für die Planungsaufgabe dennoch korrekt identifiziert.

Die in diesem Abschnitt dargestellten Verfahren, von zweistufigen Detektoren wie Faster R-CNN bis zu einstufigen Architekturen wie YOLO, verwenden achsenparallele Begrenzungsrahmen (Axis-Aligned Bounding Boxes, AABB). Diese Boxen sind durch vier Parameter definiert und ihre Kanten verlaufen stets horizontal und vertikal zum Bildrand. Für viele Anwendungen wie Fußgängererkennung oder Fahrzeugdetektion, bei denen Objekte typischerweise aufrechte oder horizontale Orientierung aufweisen, ist dies ausreichend. Technische Zeichnungen mit beliebig rotierten Symbolen und Textannotationen stellen jedoch besondere Anforderungen, die achsenparallele Boxen nur unzureichend erfüllen. Abschnitt 3.2 führt daher orientierte Begrenzungsrahmen (Oriented Bounding Boxes, OBB) ein, die durch einen zusätzlichen Rotationsparameter präzisere Lokalisierung rotierter Objekte ermöglichen.

3.2 Oriented Object Detection für rotierte Objekte

Die in Abschnitt 3.1.5 vorgestellten objekterkennenden Modelle basieren überwiegend auf achsenparallelen Bounding Boxes (Axis-Aligned Bounding Boxes, AABB)[21]. Bei der Detektion rotierter Objekte, wie sie in technischen Zeichnungen und Gleisplänen vorliegen, stoßen AABB-basierte Ansätze jedoch an fundamentale Grenzen. Oriented Bounding Boxes (OBB) bieten hier eine geometrisch präzisere Alternative, indem sie zusätzlich zum Zentrum und den Abmessungen auch den Rotationswinkel des Objekts modellieren.

3.2.1 Limitierungen achsenparalleler Bounding Boxes

Achsenparallele Bounding Boxes werden durch vier Parameter definiert: Zentrumskoordinaten (c_x, c_y) sowie Breite w und Höhe h. Die Orientierung dieser Boxen ist fest an die Bildachsen gekoppelt, was bei rotierten Objekten zu erheblichen Nachteilen führt.

Hintergrundanteil bei Rotation

Der fundamentale Nachteil von AABB zeigt sich im Verhältnis zwischen der vom Objekt tatsächlich belegten Fläche und der durch die Bounding Box umschlossenen Gesamtfläche. Bei rotierten Objekten umschließen achsenparallele Bounding Boxes einen erheblichen Anteil irrelevanten Hintergrunds[26]. Die AABB muss alle Ecken des rotierten Objekts umschließen und umfasst dabei große Bereiche außerhalb des eigentlichen Objekts. Die OBB hingegen liegt eng am Objekt an und minimiert den Hintergrundanteil. Dieser Unterschied ist in Abbildung 3.4 für ein um $45°$ rotiertes Textlabel dargestellt.

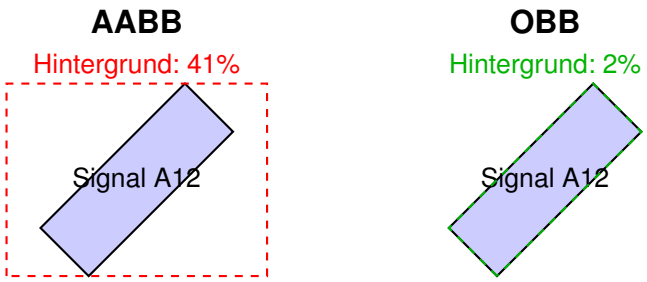


Abbildung 3.4: AABB vs OBB Vergleich bei rotiertem Textlabel.

Dieser hohe Hintergrundanteil hat zwei wesentliche Konsequenzen für die Objekterkennung: Erstens verschlechtert sich das Signal-Rausch-Verhältnis der Eingabedaten für nachgelagerte Verarbeitungsschritte wie OCR[27]. Zweitens steigt die Wahrscheinlichkeit falscher Detektionen, da irrelevante Bildelemente innerhalb der Bounding Box liegen[26].

Non-Maximum Suppression bei dichten Objektgruppen

Ein weiteres Problem ergibt sich bei der Nachbearbeitung von Detektionen mittels Non-Maximum Suppression (NMS)[20]. NMS eliminiert redundante Detektionen durch Unterdrückung überlappender Boxen basierend auf ihrer Intersection over Union (IoU):

$$\text{IoU}(B_1, B_2) = \frac{\text{Area}(B_1 \cap B_2)}{\text{Area}(B_1 \cup B_2)} \tag{3.4}$$

Bei dicht platzierten rotierten Objekten führen AABB zu hohen IoU-Werten, obwohl die Objekte selbst nicht überlappen[26]. Dies resultiert in fälschlicher Unterdrückung korrekter Detektionen. Abbildung 3.5 demonstriert dieses Fehlverhalten anhand zweier nahe beieinanderliegender Textlabels. Die beiden Textobjekte sind um $30°$ bzw. $-30°$ rotiert und überlappen sich nicht. Ihre achsenparallelen Bounding Boxes hingegen weisen eine signifikante Überlappung mit IoU = 0.42 auf. Bei einem typischen NMS-Schwellenwert von $\tau = 0.4$[28] würde eine der beiden korrekten Detektionen fälschlicherweise unterdrückt. Im Gegensatz dazu berechnet sich die IoU zwischen den orientierten Bounding Boxes korrekt zu 0.0, da die Objekte tatsächlich nicht überlappen.

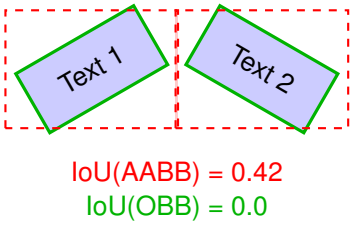


Abbildung 3.5: NMS-Problem bei dicht platzierten rotierten Objekten.

3.2.2 Oriented Bounding Boxes als Lösung

Oriented Bounding Boxes erweitern die AABB-Repräsentation um einen zusätzlichen Freiheitsgrad: den Rotationswinkel θ. Eine OBB wird durch fünf Parameter definiert:

$$\text{OBB} = (c_x, c_y, w, h, \theta) \tag{3.5}$$

wobei (c_x, c_y) das Zentrum, w und h die Dimensionen entlang der lokalen Achsen und $\theta \in [-90°, 90°)$ den Rotationswinkel relativ zur horizontalen Bildachse beschreiben. Diese Parametrisierung ermöglicht eine eng anliegende Umhüllung rotierter Objekte und reduziert den Hintergrundanteil auf das geometrische Minimum.

Vorteile für rotierte Objekte

Die präzisere geometrische Modellierung durch OBB bietet mehrere Vorteile [29]. Die IoU zwischen OBB reflektiert die tatsächliche Objektüberlappung, wodurch Fehlunterdrückungen bei NMS vermieden werden. Dies ist besonders wichtig bei dicht gruppierten Objekten, wo AABB fälschlicherweise hohe Überlappungen anzeigen würden. Zudem kodiert der Parameter θ explizit die Objektorientierung, was für nachgelagerte Prozesse wie orientierungsabhängige OCR wertvoll ist.

3.2.3 Herausforderungen bei der Verwendung von OBB

Trotz der geometrischen Vorteile bringen OBB zusätzliche Komplexität in die Detektionspipeline, die sich auf Training, Inferenz und Nachbearbeitung auswirkt. Abbildung 3.6 fasst die drei Hauptherausforderungen zusammen: erhöhte Berechnungskomplexität bei der IoU-Berechnung, die Periodizität des Rotationswinkels, die zu Trainingsinstabilität führt, und der erhöhte Trainingsaufwand gegenüber AABB-Modellen.

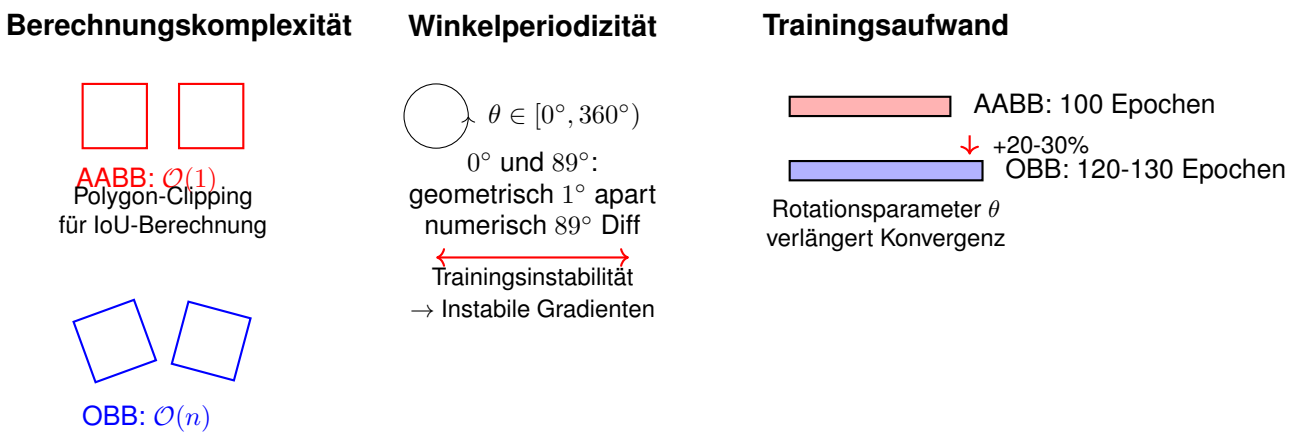


Abbildung 3.6: Hauptherausforderungen bei OBB-Verwendung: IoU-Berechnung erfordert Polygon-Clipping statt einfacher Min/Max-Operationen (links), zyklische Winkel führen zu Gradienteninstabilität (Mitte), und der zusätzliche Rotationsparameter erhöht den Trainingsaufwand (rechts).

Erhöhte Berechnungskomplexität

Die Berechnung der Intersection over Union zwischen zwei OBB ist algorithmisch aufwendiger als bei AABB. Während die AABB-IoU durch einfache Min-Max-Operationen in konstanter Zeit $\mathcal{O}(1)$ berechnet werden kann, erfordert die OBB-IoU die Lösung eines Polygon-Clipping-Problems[30]:

$$\text{IoU}_{\text{OBB}}(B_1, B_2) = \frac{\text{Area}(P_1 \cap P_2)}{\text{Area}(P_1 \cup P_2)} \tag{3.6}$$

wobei P_1 und P_2 die als Vierecke repräsentierten OBB sind. Die Schnittpunktberechnung zwischen den Kantensegmenten skaliert mit $\mathcal{O}(n^2)$ für n-Ecke, was in der Praxis zu einem Faktor 3-5 langsameren NMS-Durchläufen führt[29].

Winkelperiodizität und Trainingsinstabilität

Der Rotationswinkel θ weist eine zyklische Periodizität auf: Eine Rotation um $180°$ (bzw. bei symmetrischen Objekten um $90°$) resultiert in einer geometrisch identischen Bounding Box[31, 32]. Dies führt zu Ambiguitäten in der Netzwerk-Ausgabe. Betrachtet man beispielsweise zwei Vorhersagen $\theta_1 = 0°$ und $\theta_2 = 89°$ für dasselbe Objekt, so sind diese geometrisch nur $1°$ voneinander entfernt, numerisch jedoch $89°$. Standard-Regressionsverluste (z.B. Smooth L1) behandeln diese Diskrepanz linear, was zu instabilen Gradienten führt.

Erhöhter Trainingsaufwand

Die zusätzliche Parameterregression für θ erhöht die Komplexität des Lernproblems. Empirische Studien zeigen, dass OBB-Modelle typischerweise 20-30% mehr Trainingsepochen benötigen als vergleichbare AABB-Modelle, um ähnliche Konvergenzniveaus zu erreichen[29]. Eine

Trainingsepoche bezeichnet dabei einen vollständigen Durchlauf durch den gesamten Trainingsdatensatz, bei dem jedes Trainingsbeispiel einmal zur Gewichtsoptimierung verwendet wird. Dies resultiert aus der Notwendigkeit, sowohl die präzise Lokalisierung als auch die korrekte Orientierungsprädiktion zu erlernen.

3.2.4 YOLOv8-OBB Architektur

YOLOv8 bietet eine spezialisierte OBB-Variante, die für die Symbolerkennung geeignet ist. Die Architekturwahl wird in Kapitel **??** begründet. YOLOv8-OBB erweitert die Standard-Architektur um die Regression des Rotationsparameters θ, während die grundlegende Netzwerkstruktur aus Backbone, Neck und Detection Heads erhalten bleibt[22]. Die schematische Architektur ist in Abbildung 3.7 dargestellt. Der CSPNet-Backbone extrahiert hierarchische Merkmale und erzeugt eine Feature Pyramid mit drei Auflösungsstufen: P3 (128×128) für kleine Objekte wie GKS-Symbole, P4 (64×64) für mittlere Objekte und P5 (32×32) für große Objekte wie Signale. Das PANet-Neck fusioniert diese Multi-Scale-Features durch bidirektionale Pfade (Top-Down und Bottom-Up), bevor drei spezialisierte Detection Heads die finalen OBB-Vorhersagen $(c_x, c_y, w, h, \theta, \text{cls})$ produzieren.

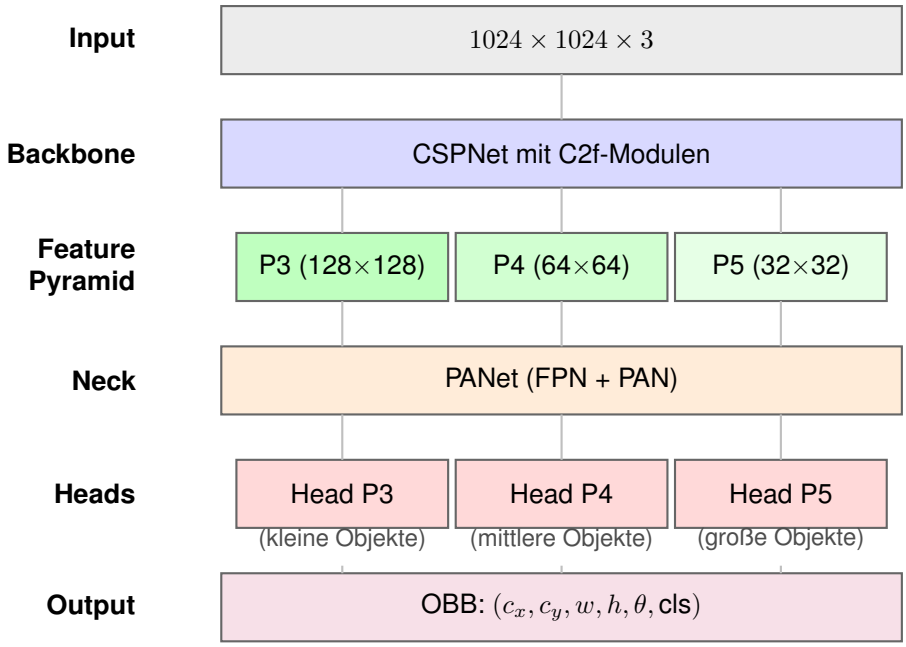


Abbildung 3.7: YOLOv8-OBB Netzwerkarchitektur (nach [22]).

Netzwerkkomponenten

Die Architektur gliedert sich in drei funktionale Blöcke. Der Backbone extrahiert hierarchische Merkmale aus den Eingabebildern und verwendet eine auf CSPNet basierende Architektur mit Cross Stage Partial Connections[33]. Die sogenannten C2f-Module (CSP Bottleneck with 2 Convolutions, faster) ersetzen die C3-Module früherer Versionen und bieten eine effizientere Gradientenfluss-Charakteristik bei gleichzeitiger Reduktion der Parameteranzahl. Das Neck-

Modul fusioniert Merkmale unterschiedlicher Auflösungsstufen durch eine Kombination aus Top-Down- und Bottom-Up-Pfaden. Die Path Aggregation Network (PANet) Architektur[34] ermöglicht eine effektive Informationspropagation zwischen den Skalen, was für die Detektion sowohl kleiner als auch großer Objekte essentiell ist. YOLOv8 folgt einem Anchor-Free-Ansatz[35], bei dem Objektzentren direkt vorhergesagt werden, anstatt vordefinierte Anker-Boxen zu verwenden. Die Heads sind dezentralisiert (decoupled): Separate Zweige regredieren die Lokalisierung (inklusive θ) und die Klassifikation. Diese Entkopplung verbessert die Konvergenz, da die beiden Aufgaben unterschiedliche Repräsentationen erfordern[36].

Verlustfunktion und Training

Das Training von YOLOv8-OBB optimiert eine kombinierte Verlustfunktion, die drei Komponenten umfasst:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{box}}\mathcal{L}_{\text{box}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}} + \lambda_{\text{dfl}}\mathcal{L}_{\text{dfl}} \tag{3.7}$$

Die Box-Loss $\mathcal{L}_{\text{box}}$ quantifiziert die Lokalisierungsgenauigkeit der vorhergesagten OBB gegenüber der Ground Truth. Die Classification-Loss $\mathcal{L}_{\text{cls}}$ verwendet Binary Cross-Entropy für die Klassenzuordnung[28]. Focal Loss adressiert dabei das Problem der Klassenungleichverteilung, das bei Gleisplänen auftritt, da manche Symbolklassen (z.B. Koordinaten) deutlich häufiger vorkommen als andere (z.B. Gleismagnete). Die Distribution Focal Loss (DFL)[37] verbessert die Bounding-Box-Regression durch Modellierung der Koordinaten als Verteilungen anstelle von Punktschätzungen.

Die konkreten Trainingsparameter und -ergebnisse für den im Rahmen dieser Arbeit trainierten Detektor werden in Kapitel 6 detailliert dokumentiert. Die Kombination aus Echtzeitfähigkeit, Rotationsinvarianz und Anchor-Free-Design macht YOLOv8-OBB zu einem geeigneten Kandidaten für die Gleisplananalyse. Die endgültige Architekturentscheidung wird in Kapitel **??** dokumentiert.

3.2.5 Relevanz für Gleisplananalyse

Die beschriebenen OBB-Eigenschaften sind besonders relevant für die automatisierte Analyse von Gleisplänen. Technische Zeichnungen dieses Typs weisen drei charakteristische Merkmale auf, die den Einsatz orientierter Bounding Boxes notwendig machen. Symbole folgen der Topologie der Gleisachsen und können daher in jedem Winkel $\theta \in [0°, 360°)$ vorliegen. In Bahnhöfen und Weichenstraßen treten Symbole räumlich konzentriert auf, wobei AABB-basierte NMS zu Fehlunterdrückungen führen würde. Die vom OBB-Detektor gelieferte Rotation θ ist für nachgelagerte Prozesse wie die orientierungsabhängige OCR (vgl. Kapitel 6.3) essentiell, da Textlabels entsprechend ihrer Ausrichtung verarbeitet werden müssen.

Die in Kapitel 4 definierten funktionalen Anforderungen an die Symbolerkennung erfordern daher

zwingend den Einsatz rotationsinvarianter Detektionsverfahren. Die in Kapitel 6 beschriebene Implementierung nutzt YOLOv8-OBB mit Standard-Konfiguration für die Detektion von 13 domänenspezifischen Symbolklassen (vgl. Tabelle 4.1 und 4.2 in Kapitel 4).

3.3 Optical Character Recognition (OCR)

Die optische Zeichenerkennung bildet das verbindende Element zwischen der visuellen Symbolerkennung und der semantischen Interpretation von Gleisplänen. Während die in Abschnitt 3.1.3 beschriebene Objekterkennung Symbole lokalisiert und klassifiziert, liefert sie zunächst keine Information über die zugehörigen Textinhalte. Erst durch die zuverlässige Extraktion von Signalbezeichnungen (z. B. „AS102“), Kilometerangaben (z. B. „18.1606“) und GKS-Kennungen (z. B. „1234“) erhalten die erkannten Symbole ihre vollständige technische Bedeutung.

Dieser Abschnitt behandelt zunächst die allgemeinen Grundlagen der optischen Zeichenerkennung, bevor die spezifischen Herausforderungen technischer Zeichnungen und die für Gleispläne relevanten Lösungsansätze dargelegt werden. Die Progression folgt dabei dem Muster der vorangegangenen Abschnitte: vom allgemeinen Prinzip über moderne Architekturen bis zur domänenspezifischen Anwendung.

3.3.1 Grundlagen und Entwicklung der OCR-Technologie

Die optische Zeichenerkennung (Optical Character Recognition, OCR) bezeichnet die automatisierte Umwandlung von Bildern, die Text enthalten, in maschinenlesbaren Text. Diese Aufgabe lässt sich konzeptionell in vier Verarbeitungsstufen unterteilen: Vorverarbeitung, Textdetektion, Zeichenerkennung und Nachverarbeitung.

Die **Vorverarbeitung** bereitet das Eingabebild für die nachfolgenden Schritte auf. Typische Operationen umfassen Kontrastverbesserung, Rauschunterdrückung und Binarisierung (Umwandlung in Schwarz-Weiß). Bei technischen Zeichnungen ist zusätzlich die Entfernung störender Linienelemente relevant [38].

Die **Textdetektion** lokalisiert Bereiche im Bild, die Text enthalten. Moderne Verfahren verwenden neuronale Netze zur Segmentierung von Textregionen, wobei die Ausgabe typischerweise als Bounding Boxes oder Polygone erfolgt [39].

Die **Zeichenerkennung** wandelt die lokalisierten Bildausschnitte in Zeichenfolgen um. Dies ist die eigentliche „Erkennung“ im engeren Sinne und bildet den rechenintensivsten Schritt der Pipeline.

Die **Nachverarbeitung** validiert und korrigiert die erkannten Texte. Für domänenspezifische Anwendungen wie Gleispläne können hier Musterprüfungen (Regular Expressions) und Plausibilitätskontrollen integriert werden.

3.3.2 Architekturen moderner OCR-Systeme

Moderne OCR-Systeme basieren auf zwei dominierenden Architekturparadigmen: der CRNN-Architektur (Convolutional Recurrent Neural Network) und Transformer-basierten Ansätzen. Beide werden im Folgenden erläutert.

CRNN-Architektur

Die CRNN-Architektur, eingeführt von Shi et al. [40], kombiniert drei Komponenten zu einem End-to-End-trainierbaren System (Abbildung 3.8). In dieser Arbeit ermöglicht CRNN die Erkennung der alphanumerischen Beschriftungen in Gleisplänen, darunter Signalnamen (z.B. AS102), Koordinatenangaben und GKS-Bezeichnungen.

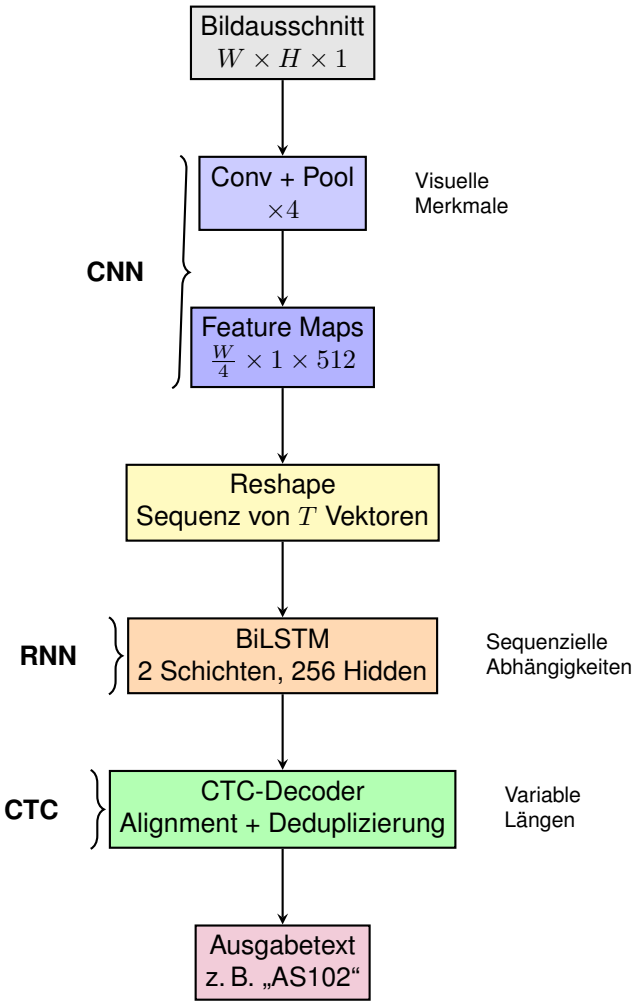


Abbildung 3.8: CRNN-Architektur

CNN-Komponente (Feature Extraction): Ein Convolutional Neural Network extrahiert visuelle Merkmale aus dem Eingabebild. Durch sukzessive Faltungs- und Pooling-Operationen wird das Bild auf eine Sequenz von Merkmalsvektoren reduziert. Ein Bildausschnitt der Größe $W \times H$ wird typischerweise auf $T = W/4$ Zeitschritte mit je 512 Merkmalskanälen komprimiert[40].

RNN-Komponente (Sequence Modeling): Ein bidirektionales LSTM (Long Short-Term Memory) [41] verarbeitet die Merkmalssequenz und modelliert kontextuelle Abhängigkeiten zwischen benachbarten Zeichen. Die Bidirektionalität ermöglicht die Berücksichtigung sowohl vergangener als auch zukünftiger Kontextinformation.

CTC-Decoder (Transcription): Die Connectionist Temporal Classification (CTC) [42] löst das Problem variabler Ausgabelängen. Da die Anzahl der CNN-Ausgabespalten nicht mit der Anzahl der Textzeichen übereinstimmt, führt CTC ein „Blank"-Symbol ein und definiert eine Verlustfunktion, die alle möglichen Alignments zwischen Eingabe und Ausgabe marginalisiert:

$$P(y \mid x) = \sum_{\pi \in \mathcal{B}^{-1}(y)} \prod_{t=1}^{T} P(\pi_t \mid x) \tag{3.8}$$

wobei y die Zielsequenz, x die Eingabe, $\mathcal{B}^{-1}(y)$ die Menge aller Pfade bezeichnet, die nach Entfernung von Blanks und Deduplizierung y ergeben, und $P(\pi_t \mid x)$ die Wahrscheinlichkeit des Symbols π_t zum Zeitpunkt t ist.

Für Maschinenbau-Ingenieure lässt sich die CTC-Funktionsweise anschaulich erklären: Das Netz gibt für jeden Zeitschritt (jede Spalte des Bildes) eine Wahrscheinlichkeitsverteilung über alle möglichen Zeichen aus. Die Ausgabe „A–S-11-0-2" (wobei „-" das Blank-Symbol bezeichnet) wird durch Entfernung der Blanks und Zusammenfassung aufeinanderfolgender identischer Zeichen zu „AS102" dekodiert. Diese Fähigkeit zur Verarbeitung variabler Textlängen ist für Gleispläne essentiell, da Beschriftungen von kurzen Signalnamen (z.B. „AS1") bis zu längeren Koordinatenangaben (z.B. „18.1606") reichen.

Transformer-basierte OCR

Neuere Ansätze ersetzen die RNN-Komponente durch Transformer-Architekturen [43]. Der TrOCR-Ansatz [44] verwendet einen Vision Transformer (ViT) als Encoder und einen Text Transformer als Decoder. Vision Transformer ist ein Transformer Modell, das Bilder als Sequenz von Patches verarbeitet[45]. Der zentrale Mechanismus ist die Self-Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.9}$$

wobei Q (Query), K (Key) und V (Value) lineare Projektionen der Eingabe sind und d_k die Dimensionalität der Keys bezeichnet. Die Division durch $\sqrt{d_k}$ stabilisiert die Gradienten.

Transformer-basierte Ansätze bieten Vorteile bei langen Sequenzen und ermöglichen effizientere Parallelisierung während des Trainings. Für kurze Texte wie Signalbezeichnungen (typischerweise 3–8 Zeichen) sind die Vorteile jedoch marginal, weshalb CRNN-basierte Systeme wie PaddleOCR in der Praxis häufig bevorzugt werden.

Vergleich der Architekturen

Tabelle 3.2 fasst die wesentlichen Unterschiede zwischen CRNN und Transformer-basierten Ansätzen zusammen.

Kriterium	CRNN (z. B. PaddleOCR)	Transformer (z. B. TrOCR)
Genauigkeit (kurze Texte)	>95% [46]	>95% [44]
Inferenzzeit (CPU)	8–15 ms/Bild [46]	50–150 ms/Bild[1]
Modellgröße	8–12 MB [46]	300–500 MB [44]
Trainingsdatenbedarf	Moderat (10k–100k Samples)	Hoch (>1M Samples für Pre-training) [44]

Tabelle 3.2: Vergleich von CRNN- und Transformer-basierten OCR-Architekturen

3.3.3 OCR-Systeme im praktischen Einsatz

Für die praktische Anwendung stehen mehrere etablierte OCR-Frameworks zur Verfügung, die unterschiedliche Stärken aufweisen.

Tesseract

Tesseract [47] ist eine Open-Source-Engine, die ursprünglich von Hewlett-Packard entwickelt und später von Google weiterentwickelt wurde. Seit Version 4.0 verwendet Tesseract ein LSTM-basiertes Erkennungsmodell. Die Konfiguration erfolgt über Page Segmentation Modes (PSM), die das erwartete Dokumentlayout spezifizieren:

- **PSM 7**: Einzelne Textzeile – optimal für isolierte Beschriftungen
- **PSM 8**: Einzelnes Wort – für kompakte Labels
- **PSM 13**: Raw Line – für einzelne Textzeilen ohne Layoutsegmentierung, optimal für isolierte Beschriftungen

Eine Whitelist-Funktionalität ermöglicht die Einschränkung des Zeichensatzes auf erwartete Zeichen (z. B. nur Großbuchstaben und Ziffern für Signalbezeichnungen), was die Erkennungsgenauigkeit in domänenspezifischen Anwendungen erhöht.

PaddleOCR

PaddleOCR [46] ist ein vom chinesischen Technologiekonzern Baidu entwickeltes Framework, das eine vollständige Pipeline aus Textdetektion, Winkelklassifikation und Texterkennung bereitstellt. Für die vorliegende Arbeit wurde PaddleOCR gewählt, da es integrierte Winkelerkennung

[1] Transformer-basierte Modelle erfordern aufgrund des quadratischen Attention-Mechanismus höhere Inferenzzeiten [43].

für rotierte Texte bietet und effiziente Verarbeitung der zahlreichen Textregionen in Gleisplänen ermöglicht. Die PP-OCRv3-Architektur besteht aus drei Komponenten:

1. **DB-Detektor (Differentiable Binarization)**: Lokalisiert Textregionen durch semantische Segmentierung mit differenzierbarer Schwellenwertberechnung [39].

2. **Winkelklassifikator**: Bestimmt die Orientierung des Textes (0° oder 180°) zur automatischen Korrektur.

3. **CRNN-Recognizer**: Erkennt die Zeichenfolge basierend auf der in Abschnitt 3.3.2 beschriebenen Architektur.

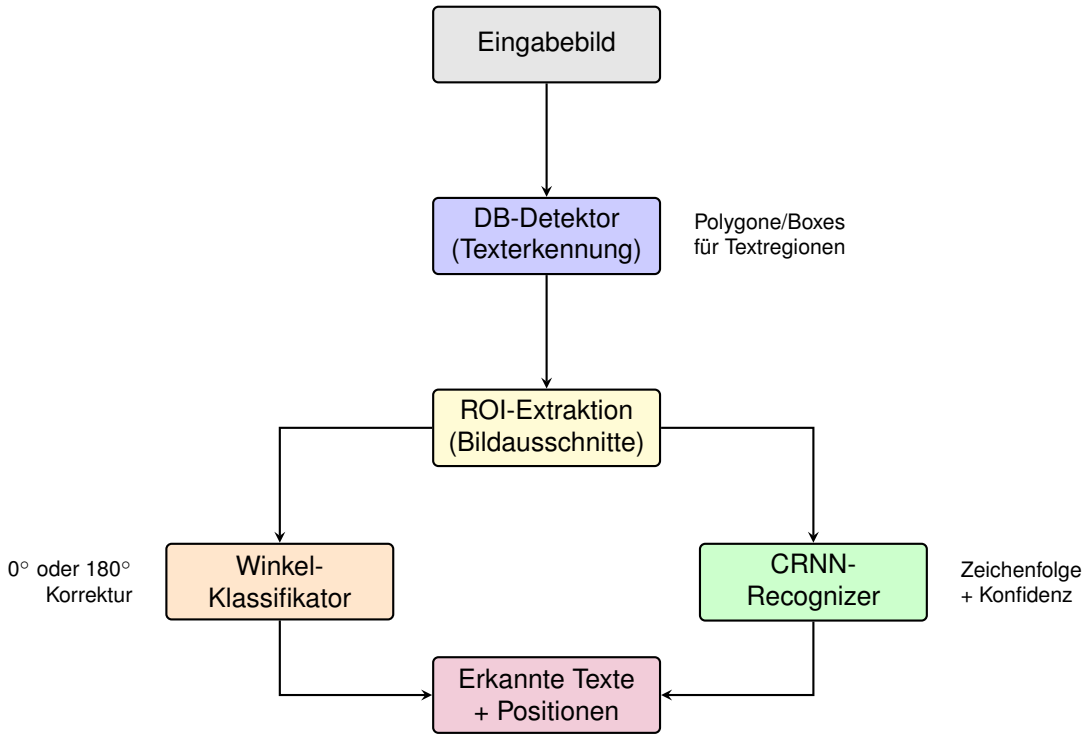


Abbildung 3.9: PaddleOCR-Pipeline: Dreistufige Architektur mit Detektion, Winkelkorrektur und Erkennung.

Vergleich der OCR-Systeme

Tabelle 3.3 vergleicht die für technische Zeichnungen relevanten OCR-Systeme.

System	Architektur	Stärken	Schwächen
Tesseract	LSTM-basiert	Whitelist-Funktion, geringer Ressourcenbedarf	Empfindlich bei Rotation, langsam bei vielen ROIs
PaddleOCR	DB + CRNN	Integrierte Winkelerkennung, schnell, robust	Komplexere Installation, größeres Modell
EasyOCR	CRAFT(Character Region Awareness for Text) + CRNN [48]	Einfache API, gute Winkeltoleranz	Langsamer, weniger konfigurierbar
TrOCR	Transformer	State-of-the-Art Genauigkeit	Hoher GPU-Bedarf, langsam

Tabelle 3.3: Vergleich etablierter OCR-Systeme für technische Anwendungen

3.3.4 Evaluationsmetriken für Texterkennung

Für die isolierte Bewertung von OCR-Systemen existieren etablierte Metriken wie die **Character Error Rate (CER)** und **Word Error Rate (WER)**, die den Anteil fehlerhafter Zeichen bzw. Wörter quantifizieren [49, 50]. Die **Feldgenauigkeit** (Field Accuracy) bewertet, ob ein vollständiges Textfeld exakt korrekt erkannt wurde.

Für integrierte Extraktionspipelines, bei denen OCR nur eine von mehreren Stufen darstellt (Detektion → OCR → Verknüpfung → Validierung), ist jedoch eine *End-to-End-Bewertung* aussagekräftiger: Sie misst, ob das finale Extraktionsergebnis korrekt ist, unabhängig davon, in welcher Pipeline-Stufe Fehler auftraten oder kompensiert wurden. Dieser Ansatz wird in der vorliegenden Arbeit verfolgt.

3.3.5 Herausforderungen bei technischen Zeichnungen

Technische Zeichnungen wie Gleispläne stellen OCR-Systeme vor spezifische Herausforderungen, die über typische Dokumentenscans hinausgehen. Diese Herausforderungen werden in Abbildung 3.10 visualisiert.

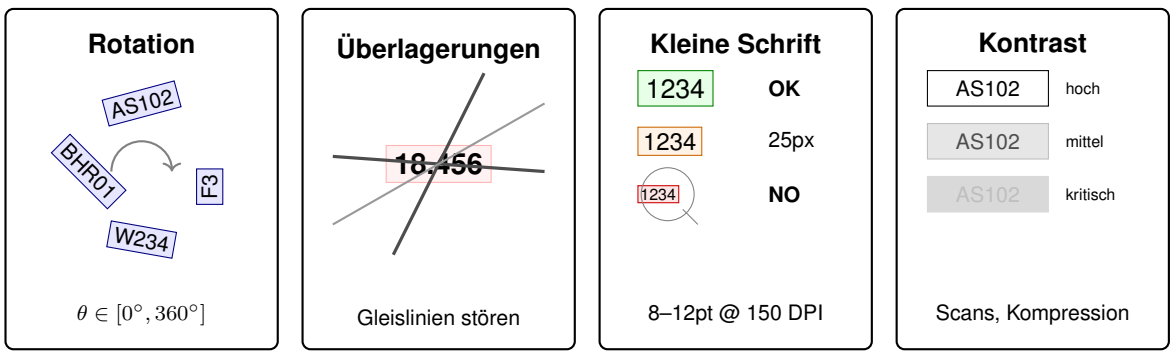


Abbildung 3.10: Zentrale Herausforderungen der OCR in technischen Zeichnungen.

Variable Textorientierung

In Gleisplänen folgen Beschriftungen der Topologie der Gleise und können in beliebigen Winkeln $\theta \in [0°, 360°]$ orientiert sein. Standard-OCR-Engines sind primär für horizontalen Text ($0°$) optimiert und zeigen bei geneigten Texten signifikante Leistungseinbußen [51].

Lösungsansätze umfassen:

- **Vorrotation**: Transformation des Bildausschnitts basierend auf dem von der Objekterkennung gelieferten Winkel θ
- **Multi-Winkel-Inferenz**: Sequenzielle Verarbeitung mit $0°, 90°, 180°, 270°$ Rotation und Auswahl des konfidentesten Ergebnisses
- **Rotationsinvariante Netze**: Verwendung von Spatial Transformer Networks (STN) [52] zur lernbasierten Entzerrung

Grafische Überlagerungen

Technische Zeichnungen enthalten zahlreiche Linienelemente (Gleise, Bemaßungen, Führungslinien), die Textbereiche durchkreuzen können. Diese Überlagerungen führen zu Segmentierungsfehlern und Fehlerkennungen.

Lösungsansätze umfassen:

- **Morphologische Operationen**: Opening-Filter mit linienförmigen Strukturelementen zur Entfernung dünner Linien [53]. Die Implementierung verwendet orientierte Strukturelemente, die dem Textwinkel folgen, um schräge Linien gezielt zu entfernen ohne Textstriche zu beschädigen.
- **Farbbasierte Filterung**: Separation von Textfarbe (typischerweise Schwarz) und Linienfarben (oft Grau oder Farbe)

Theoretische Alternativen wie Inpainting [54] zur Rekonstruktion überlagerter Bereiche wurden evaluiert, jedoch aufgrund des höheren Rechenaufwands und der ausreichenden Wirksamkeit morphologischer Operationen nicht implementiert.

Kleine Textgrößen

Beschriftungen in Gleisplänen sind oft nur 8–12 Punkt groß. Bei einer Digitalisierung mit 150 DPI entspricht dies einer Zeichenhöhe von lediglich 17–25 Pixeln. Für zuverlässige OCR werden typischerweise mindestens 30–40 Pixel Zeichenhöhe benötigt [47, 46].

Die Beziehung zwischen Schriftgröße s (in Punkt), Scanauflösung r (in DPI) und resultierender Pixelhöhe h_c ist:

$$h_c = s \cdot \frac{r}{72} \cdot k \tag{3.10}$$

wobei 72 die Referenz-DPI für Punktgrößen und $k \approx 0.7$ der typische Verhältnisfaktor zwischen nomineller Schrifthöhe und tatsächlicher x-Höhe (Höhe des Kleinbuchstabens „x“) bezeichnet [53].

Lösungsansätze umfassen:

- **Erhöhte Scanauflösung**: Digitalisierung mit 300–500 DPI statt 150 DPI
- **Interpolationsbasierte Hochskalierung**: Vergrößerung kleiner Bildausschnitte mittels Lanczos-Interpolation vor der OCR-Verarbeitung. Dies ist recheneffizienter als neuronale Super-Resolution und für technische Zeichnungen mit klar definierten Kanten ausreichend.
- **Multi-Scale-Processing**: Verarbeitung auf mehreren Skalierungsstufen mit Ergebnisfusion

3.3.6 Qualitätssicherung und Nachbearbeitung

Die Robustheit eines OCR-Systems wird maßgeblich durch die Nachbearbeitungsschritte bestimmt. Für domänenspezifische Anwendungen wie Gleispläne können strukturelle Erwartungen zur Validierung und Korrektur genutzt werden.

Konfidenzbasierte Filterung

OCR-Engines liefern typischerweise einen Konfidenzwert $c \in [0, 1]$ für jedes erkannte Ergebnis. Die Implementierung verwendet kontextabhängige Schwellenwerte:

$$\text{Akzeptanz}(c) = \begin{cases} \text{verwerfen} & c < \tau_{\text{min}} \\ \text{akzeptieren} & c \geq \tau_{\text{final}} \end{cases} \tag{3.11}$$

Dabei werden drei Stufen unterschieden:

- $\tau_{\text{min}} = 0.4$: Vorfilterung offensichtlich fehlerhafter Detektionen
- $\tau_{\text{early}} = 0.5$: Early-Exit bei vollständigen, hochkonfidenten Erkennungen zur Effizienzsteigerung
- $\tau_{\text{final}} = 0.8$: Finale Validierung für sicherheitsrelevante Ausgaben im Bahnbereich

Diese gestufte Filterung ermöglicht eine effiziente Verarbeitung bei gleichzeitiger Minimierung falsch-positiver Erkennungen.

Musterbasierte Validierung

Gleisplan-Beschriftungen folgen standardisierten Formaten, die durch reguläre Ausdrücke (Regular Expressions) formalisiert werden können. Ein erkannter Text gilt als valide, wenn er dem erwarteten Muster der jeweiligen Klasse entspricht. Tabelle 3.4 definiert die Validierungsmuster für die wichtigsten Textklassen.

Klasse	Regex-Pattern	Beispiele
Signal	`^[A-Z]{1,4}\d{1,4}[a-z]?$`	AS102, BHR201, F3a
Koordinate	`^\d{1,3}[.,]\d{3,4}$`	18.1606, 123,456
GKS	`^\d{4}$`	1234, 5678
Weiche	`^W\d{2,4}[a-z]?$`	W12, W234a

Tabelle 3.4: Regex-Validierungsmuster für Gleisplan-Textklassen

Homoglyphen-Korrektur

OCR-Systeme verwechseln häufig visuell ähnliche Zeichen (Homoglyphen). Tabelle 3.5 listet typische Verwechslungen und deren automatische Korrekturfähigkeit.

Korrekt	Verwechselt mit	Kontextuelle Korrektur	Automatisierbar
O	0, Q	Signal-IDs enthalten keine 0; Koordinaten keine O	Ja
I	1, l	Position im Muster (Anfang: Buchstabe, Mitte: Ziffer)	Teilweise
S	5	Signal-Präfixe sind Buchstaben	Ja
B	8, 3	Kontextabhängig	Teilweise
Z	2	Selten in Signalnamen	Ja

Tabelle 3.5: Typische OCR-Homoglyphen und deren Korrekturmöglichkeiten

Fuzzy-Matching zur Fehlerkorrektur

Bei ungültigen OCR-Erkennungen kann ein Abgleich mit einer Referenzliste bekannter korrekter Werte durchgeführt werden. Die Levenshtein-Distanz [55] quantifiziert dabei die Ähnlichkeit zweier Zeichenketten als minimale Anzahl von Einfüge-, Lösch- und Ersetzungsoperationen zur Transformation eines Strings in einen anderen.

OCR-Ergebnis	Referenz	d_{Lev}	Aktion
AS1O2	AS102	1	Korrektur: O $\rightarrow$ 0
BHR2O1	BHR201	1	Korrektur: O $\rightarrow$ 0
XYZAB	AS102	5	Keine Korrektur (zu verschieden)

Tabelle 3.6: Fuzzy-Matching mittels Levenshtein-Distanz

Dieses Verfahren wird als *Fuzzy-Matching* bezeichnet, da es unscharfe (approximate) Übereinstimmungen erlaubt. Für sicherheitsrelevante Anwendungen wird ein konservativer Schwellenwert von $d_{\text{Lev}} \leq 2$ gewählt, um Fehlkorrekturen zu vermeiden. Eine automatische Korrektur wird ausschließlich bei $d_{\text{Lev}} \leq 2$ durchgeführt, da typische OCR-Fehler (Homoglyphen wie O/0, I/1)

einzelne Zeichenersetzungen darstellen. Höhere Distanzen deuten auf grundlegend fehlerhafte Erkennungen hin, bei denen automatische Korrektur das Risiko von Fehlzuordnungen erhöht.

3.3.7 ROI-basierte OCR-Strategien

Anstelle einer vollständigen Seitenanalyse (Full-Page OCR) fokussiert der in dieser Arbeit verfolgte Ansatz die OCR auf spezifische Regions of Interest (ROIs), die aus der Objekterkennung abgeleitet werden [56].

Vorteile der ROI-fokussierten Verarbeitung

- **Effizienz**: Reduktion des zu verarbeitenden Bildbereichs. Bei den untersuchten Gleisplänen beträgt die kumulierte ROI-Fläche typischerweise unter 5% der Gesamtseitenfläche.
- **Kontextspezifische Verarbeitung**: Klassenabhängige Vorverarbeitungsparameter (z. B. Padding, Skalierung)
- **Reduzierte Falsch-Positive**: Keine Verarbeitung irrelevanter Textbereiche (Legenden, Titelfelder)
- **Geometrische Normalisierung**: Nutzung des OBB-Winkels zur Horizontalausrichtung

Orientierungsbewusste ROI-Extraktion

Die Extraktion eines ROI für ein erkanntes Symbol mit orientierter Bounding Box (c_x, c_y, w, h, θ) erfordert eine geometrische Transformation zur Horizontalausrichtung des Textes. Die Implementierung verwendet einen Dual-Pfad-Ansatz, bei dem die Wahl der Transformationsmethode von der Textorientierung abhängt (Abbildung 3.11).

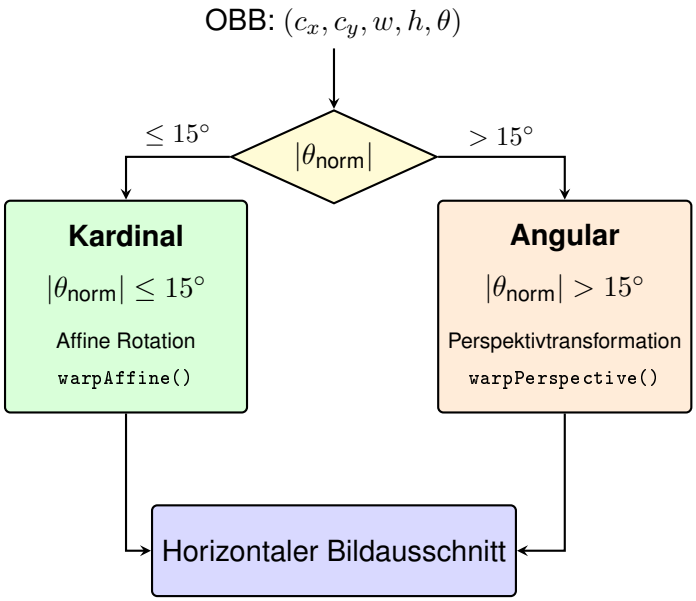


Abbildung 3.11: Dual-Pfad ROI-Extraktion: Kardinale Orientierungen verwenden effiziente affine Rotation, während schräge Texte eine Perspektivtransformation erfordern.

Kardinale Orientierung ($|\theta_{\text{norm}}| \leq 15°$): Für nahezu achsenparallele Texte genügt eine affine Rotation um den Mittelpunkt (c_x, c_y). Die Transformation verwendet eine 2×3 Rotationsmatrix:

$$\mathbf{M}_{\text{affin}} = \begin{pmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \end{pmatrix} \tag{3.12}$$

wobei t_x und t_y die Translationskomponenten bezeichnen, die sicherstellen, dass die Rotation um den Mittelpunkt (c_x, c_y) erfolgt. Diese Transformation ist recheneffizient und ausreichend für Texte, die bereits nahezu horizontal oder vertikal ausgerichtet sind.

Beliebige Orientierung ($|\theta_{\text{norm}}| > 15°$): Für stark geneigte Texte wird eine perspektivische Transformation (Homographie) verwendet [57]. Die vier Eckpunkte (x_i, y_i) der OBB werden zunächst durch Rotation um den Mittelpunkt berechnet:

$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_i - c_x \\ y_i - c_y \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix} \tag{3.13}$$

Anschließend bildet eine 3×3 Homographie-Matrix $\mathbf{H}$ diese Eckpunkte auf ein achsenparalleles Zielrechteck ab:

$$\begin{pmatrix} x_i'' \\ y_i'' \\ 1 \end{pmatrix} \sim \mathbf{H} \begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} \tag{3.14}$$

wobei $\sim$ Gleichheit bis auf einen Skalierungsfaktor bezeichnet. Die Matrix $\mathbf{H}$ wird aus den vier korrespondierenden Punktpaaren (Quell-Eckpunkte zu Ziel-Rechteck) berechnet.

Die Entscheidung zwischen beiden Methoden basiert auf dem *normalisierten* Winkel θ_{norm}, der durch die in Abschnitt **??** beschriebene Normalisierung aus dem Rohwinkel gewonnen wird. Der Schwellenwert von $15°$ wurde empirisch bestimmt und bietet einen guten Kompromiss zwischen Recheneffizienz (affine Transformation) und geometrischer Genauigkeit (perspektivische Transformation). Bei größeren Abweichungen führt die affine Rotation zu sichtbaren Verzerrungen an den Texträndern, während kleinere Winkel durch die Multi-Rotations-Inferenz ($0°, 90°, 180°, 270°$) ausreichend abgedeckt werden.

3.3.8 Relevanz für die Gleisplananalyse

Die beschriebenen OCR-Konzepte bilden die theoretische Grundlage für die in Kapitel **??** (Abschnitt **??**) beschriebene Implementierung. Die konkrete Umsetzung kombiniert:

- **Multi-Engine-Kaskadierung**: PaddleOCR als primäre Engine aufgrund der integrierten Winkelklassifikation und hohen Geschwindigkeit, mit Tesseract-Fallback für Fälle, in denen PaddleOCR niedrige Konfidenzwerte liefert

- **Dual-Winkel-Routing**: Unterscheidung zwischen kardinaler Rotation ($0°, 90°, 180°, 270°$) und beliebigen Winkeln
- **Klassenspezifische Vorverarbeitung**: Angepasste Padding- und Filterparameter pro Symbolklasse
- **Dreistufige Validierung**: Syntaktisch (Regex), semantisch (Plausibilität), konfidenzbasiert (Schwellenwerte)

Die Texttypen in Gleisplänen umfassen unter anderem Signalbezeichnungen (z. B. „AS102“), Kilometerangaben (z. B. „18.1606“), GKS-Kennungen (z. B. „1234“). Jeder Texttyp erfordert spezifische Validierungsregeln und Nachbearbeitungsstrategien, die in der Implementierung detailliert werden.

3.4 Räumliche Beziehungen in technischen Dokumenten

In den vorangegangenen Abschnitten wurde erläutert, wie einzelne Objekte (Symbole mittels Objekterkennung) und deren textuelle Beschriftungen (mittels OCR) erkannt werden. Die isolierte Erkennung allein genügt jedoch nicht: Ein erkanntes Signalsymbol erhält erst durch die Zuordnung seiner Bezeichnung (z. B. „AS102“) und Kilometrierung (z. B. „18.456“) seine vollständige technische Bedeutung.

Dieser Abschnitt behandelt die theoretischen Grundlagen, wie räumlich getrennte Elemente (Symbole und ihre zugehörigen Texte) automatisch einander zugeordnet werden können.

3.4.1 Das Prinzip der räumlichen Nähe

Die automatische Verknüpfung von Elementen basiert auf einem fundamentalen Prinzip der menschlichen Wahrnehmung: dem **Gesetz der Nähe** aus der Gestaltpsychologie [58]. Dieses besagt, dass räumlich nahe beieinander liegende Objekte als zusammengehörig wahrgenommen werden.

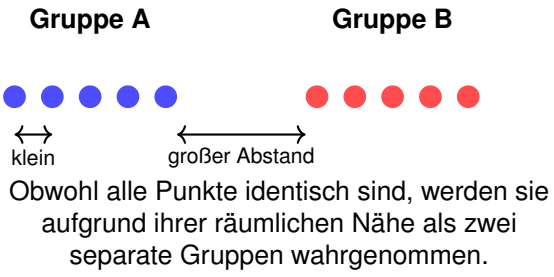


Abbildung 3.12: Illustration des Gestaltprinzips der räumlichen Nähe

Dieses psychologische Prinzip wird in der Dokumentenanalyse algorithmisch umgesetzt: Für jedes erkannte Symbol wird der räumlich *nächstgelegene* Text als zugehörig betrachtet. Die

Anwendung von Proximity Heuristiken zur automatischen Layoutanalyse ist ein etabliertes Verfahren in der Dokumentenverarbeitung [59, 60].

3.4.2 Analogie zur Stücklistenzuordnung

Für Ingenieure des Maschinenbaus ist das Konzept aus der Arbeit mit technischen Zeichnungen vertraut. Bei einer Explosionsdarstellung mit Positionsnummern ist die Zuordnung zwischen Bauteil und Nummer intuitiv klar: Die Nummer gehört zu dem Bauteil, auf das der Hinweispfeil zeigt oder das sich in räumlicher Nähe befindet (vgl. Abbildung 3.13). Dieselbe Logik liegt der automatischen Symbol-Text-Verknüpfung zugrunde.

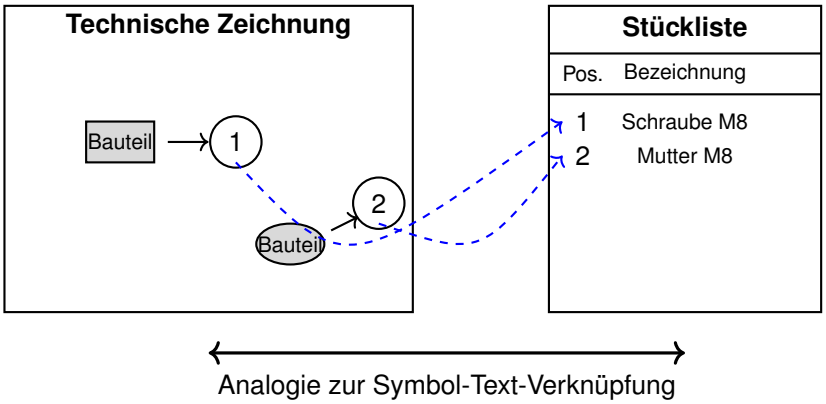


Abbildung 3.13: Zuordnung von Positionsnummern zu Bauteilen in einer Explosionsdarstellung

3.4.3 Herausforderung: Rotierte Elemente

In vielen technischen Zeichnungen folgen Symbole der Geometrie des dargestellten Objekts und können daher in **beliebigen Winkeln** orientiert sein. Dies stellt eine besondere Herausforderung dar: Die Begriffe „oberhalb“, „unterhalb“ oder „rechts von“ verlieren bei rotierten Elementen ihre eindeutige Bedeutung im globalen Koordinatensystem. Abbildung 3.14 illustriert diese Herausforderung: Bei einem horizontal ausgerichteten Symbol ist eindeutig definiert, was „unterhalb“ bedeutet. Bei einem um $45°$ gedrehten Symbol verliert dieser Begriff jedoch seine eindeutige Bedeutung im globalen Koordinatensystem.

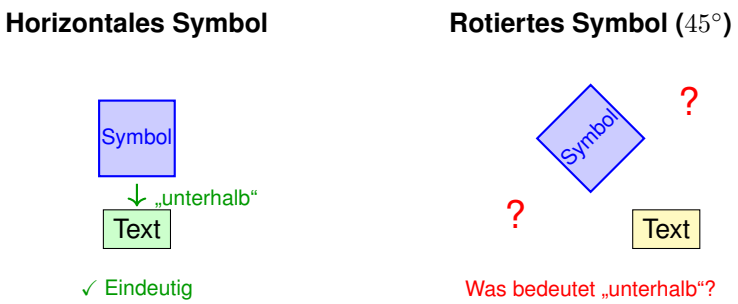


Abbildung 3.14: Ambiguität von Richtungsbegriffen bei rotierten Symbolen

3.4.4 Lösung: Lokale Koordinatensysteme

Die Lösung besteht darin, für jedes Symbol ein eigenes **lokales Koordinatensystem** zu definieren, das mit der Orientierung des Symbols mitrotiert. In diesem lokalen System haben Richtungsbegriffe wieder eine eindeutige Bedeutung.

Dieses Konzept ist Ingenieuren aus verschiedenen Bereichen vertraut:

- **Technische Mechanik**: Bei der Berechnung von Kräften an einem schräg liegenden Balken transformiert man die Koordinaten in ein lokales System, das mit dem Balken ausgerichtet ist (Hauptachsentransformation) [61].
- **CAD-Systeme**: Beim Platzieren von Features auf geneigten Flächen wird automatisch ein lokales Koordinatensystem (User Coordinate System, UCS) verwendet [10].
- **Robotik**: Werkzeugkoordinatensysteme (Tool Center Point, TCP) rotieren mit dem Endeffektor [62].

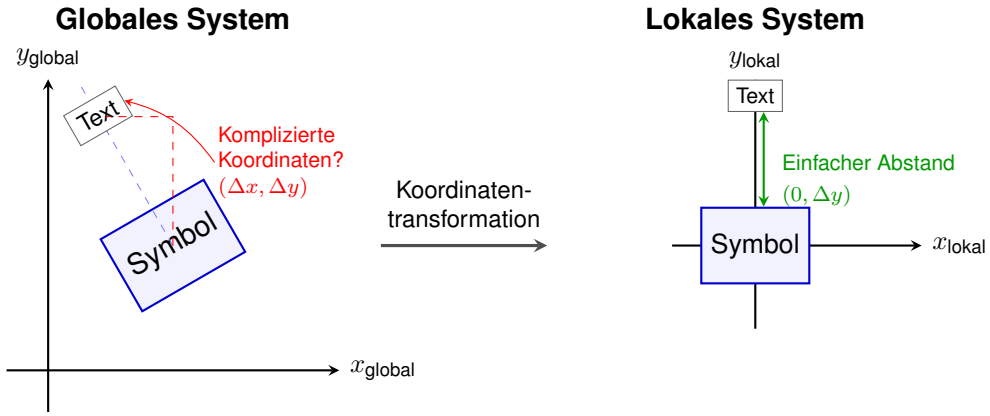


Abbildung 3.15: Transformation vom globalen ins lokale Koordinatensystem eines Symbols

Wie in Abbildung 3.15 dargestellt, werden durch die Transformation ins lokale System Richtungsbeziehungen wieder eindeutig: Der Begriff „oberhalb“ bezieht sich nun auf die positive y_{lokal}-Achse, unabhängig von der globalen Orientierung des Symbols. Die mathematische Umsetzung dieser Transformation, eine Rotation des Koordinatensystems um den Winkel des Symbols, wird in Kapitel **??**, Abschnitt **??** detailliert beschrieben.

3.4.5 Klassenspezifische Suchstrategien

In der Praxis variieren die räumlichen Beziehungen zwischen Symbol und Text je nach Objekttyp. Dies spiegelt die Zeichnungskonventionen wider, die sich in verschiedenen Domänen etabliert haben:

Objekttyp	Typische Textposition	Begründung
Signal	Unterhalb des Symbols	Signalbezeichnungen stören nicht die Gleisdarstellung
Kilometerstein	Direkt am Markierungspunkt	Kompakte Darstellung der Streckenposition
Weiche	Variable Positionen	Komplexe Geometrie erfordert flexible Platzierung

Tabelle 3.7: Domänenspezifische Konventionen für Textpositionen relativ zu Symbolen

Diese Konventionen werden als **Domänenwissen** in das System integriert und ermöglichen eine gezieltere Suche nach zusammengehörigen Elementen. Derartige domänenspezifische Layoutregeln sind charakteristisch für technische Dokumentationen und folgen etablierten Zeichnungsnormen [11, 63].

3.5 Änderungserkennung zwischen Dokumentversionen

Technische Dokumente unterliegen einem kontinuierlichen Änderungsprozess. Im Verlauf eines Projekts werden Pläne mehrfach überarbeitet: Elemente werden hinzugefügt, verschoben oder entfernt; Bezeichnungen werden korrigiert. Die systematische Erfassung dieser Änderungen ist essenziell für die Qualitätssicherung und Nachvollziehbarkeit im Engineering-Prozess.

3.5.1 Motivation: Das Revisionswesen

In der industriellen Praxis folgt die Dokumentenlenkung definierten Prozessen gemäß Qualitätsmanagementsystemen wie DIN EN ISO 9001 [64]. Jede Änderung an einer technischen Zeichnung muss:

- **Dokumentiert** werden (Was wurde geändert?)
- **Begründet** werden (Warum wurde geändert?)
- **Freigegeben** werden (Wer hat die Änderung geprüft?)

Die manuelle Identifikation von Änderungen zwischen zwei Planversionen, das sogenannte „Rödeln“ ist zeitaufwändig und fehleranfällig, insbesondere bei umfangreichen Dokumenten mit hunderten von Elementen.

3.5.2 Analogie: Stücklistenvergleich

Das Konzept der Änderungserkennung ist Ingenieuren aus dem Stücklistenwesen bekannt. Wenn sich eine Baugruppe ändert, muss die zugehörige Stückliste (Bill of Materials, BOM) aktualisiert werden [65]. Ein BOM-Vergleich identifiziert:

- **Neue Positionen**: Bauteile, die hinzugekommen sind

- **Entfallene Positionen**: Bauteile, die entfernt wurden
- **Geänderte Positionen**: Bauteile mit modifizierten Attributen (Menge, Material, etc.)

Dieselbe Logik lässt sich auf technische Zeichnungen übertragen.

3.5.3 Kategorisierung von Änderungen

Abbildung 3.16 zeigt typische Änderungen zwischen zwei Dokumentversionen: Symbol S3 wurde hinzugefügt, Symbol S2 wurde entfernt, und Symbol W1 wurde an eine neue Position verschoben.

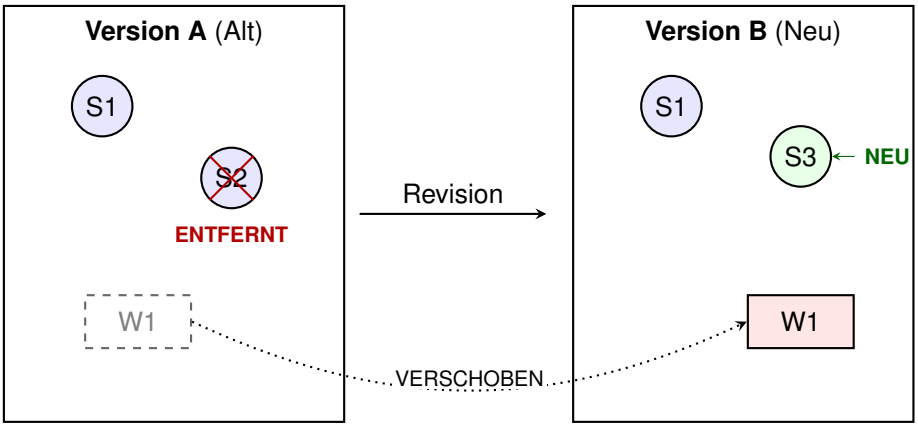


Abbildung 3.16: Schematische Darstellung von Änderungen zwischen zwei Planversionen.

Änderungstyp	Definition	Praktisches Beispiel
Hinzufügen	Element existiert nur in der neuen Version	Ein zusätzliches Signal wurde installiert
Entfernung	Element existiert nur in der alten Version	Ein Haltepunkt wurde stillgelegt
Modifikation	Element existiert in beiden Versionen, aber mit geänderten Attributen	Die Bezeichnung eines Signals wurde korrigiert
Verschiebung	Element existiert in beiden Versionen, aber an unterschiedlicher Position	Ein Symbol wurde im Plan neu platziert

Tabelle 3.8: Die vier Grundtypen von Änderungen zwischen Dokumentversionen

3.5.4 Prinzip: Identifikation durch eindeutige Merkmale

Um Änderungen automatisch zu erkennen, muss zunächst festgestellt werden, welche Elemente in beiden Versionen „dasselbe“ Objekt darstellen. Dies erfolgt über **eindeutige Identifikationsmerkmale** – vergleichbar mit Artikelnummern in einer Stückliste oder Sachnummern in einem PDM-System [65].

Für technische Zeichnungen können solche Identifikatoren aus den Objektattributen abgeleitet werden. Ein Signal mit der Bezeichnung „AS102“ erhält beispielsweise eine eindeutige Kennung, die aus Objekttyp und Bezeichnung zusammengesetzt wird. Mit solchen Identifikatoren lässt sich die Änderungserkennung auf Mengenoperationen zurückführen: Elemente nur in Version A wurden *entfernt*, Elemente nur in Version B wurden *hinzugefügt*, und Elemente in beiden Versionen sind entweder *unverändert* oder *modifiziert*. Die algorithmische Umsetzung wird in Kapitel **??**, Abschnitt **??** beschrieben.

3.6 Konfigurationsbasierte Systemanpassung

Ein wesentliches Qualitätsmerkmal technischer Software ist ihre **Anpassbarkeit** an unterschiedliche Einsatzszenarien. In der industriellen Praxis variieren Symbolvarianten, Bezeichnungskonventionen und Validierungsregeln zwischen verschiedenen Kunden, Projekten oder Regionen. Eine starre, im Programmcode verankerte Logik würde für jede Anpassung eine Softwareänderung erfordern.

3.6.1 Das Problem der Domänenspezifität

Technische Zeichnungen folgen zwar grundlegenden Normen (z. B. DIN für Zeichnungselemente) [11, 63], weisen aber in der Praxis erhebliche Variabilität auf:

- **Kundenspezifische Symbole**: Verschiedene Unternehmen verwenden unterschiedliche Darstellungen für dieselbe technische Funktion
- **Regionale Konventionen**: Bezeichnungsschemata variieren zwischen Ländern
- **Historische Entwicklung**: Ältere Dokumente folgen anderen Standards als aktuelle

Abbildung 3.17 verdeutlicht das Problem: Drei verschiedene Kunden verwenden unterschiedliche Symboldarstellungen und Bezeichnungsformate für dieselbe technische Funktion. Ein starres System müsste für jede Variante angepasst werden.

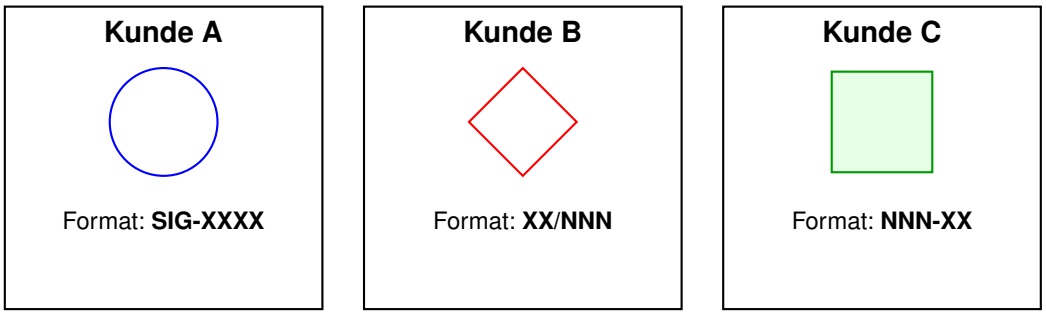


Abbildung 3.17: Variabilität von Symboldarstellungen und Bezeichnungskonventionen.

3.6.2 Lösung: Trennung von Code und Konfiguration

Die Lösung besteht in der strikten Trennung zwischen:

1. **Programmlogik** (Code): Die allgemeinen Algorithmen für Erkennung, Verknüpfung und Validierung, die für alle Anwendungsfälle gleich bleiben

2. **Domänenwissen** (Konfiguration): Die spezifischen Regeln, Parameter und Erwartungswerte, die werden in einem zentralen Konfigurationsmodul definiert

Dieses Prinzip der *Separation of Concerns* ist ein fundamentales Konzept der Softwarearchitektur [66] und ermöglicht die Anpassung des Systems durch Änderung weniger, klar strukturierter Parameter.

Dieses Prinzip ist Ingenieuren aus verschiedenen Bereichen bekannt:

- **Parametrische CAD-Modelle**: Die Geometrie wird durch Parameter gesteuert, nicht durch hartcodierte Maße [67]

- **SPS-Programmierung**: Prozessparameter werden in Datenbausteinen (DBs) abgelegt, nicht im Programmcode [68]

- **FEM-Software**: Materialparameter werden in Bibliotheken gepflegt, nicht im Solver [69]

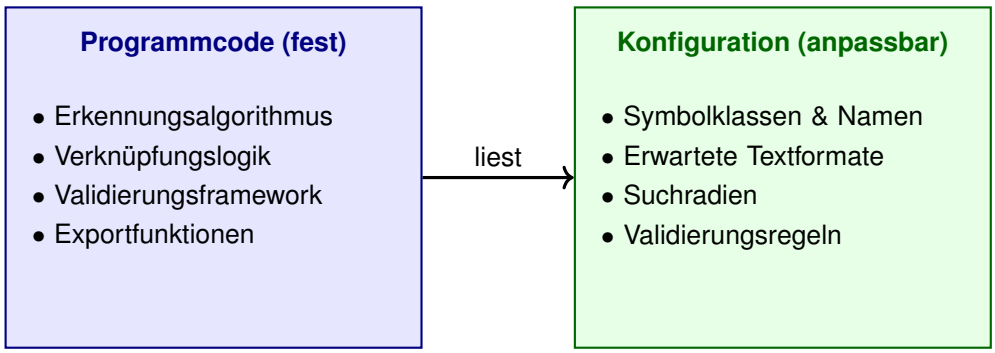


Abbildung 3.18: Architekturprinzip der Trennung von Programmlogik und Konfiguration.

Wie in Abbildung 3.18 dargestellt, liest der unveränderliche Programmcode die domänenspezifischen Parameter aus einem externen Konfigurationsmodul. Anpassungen an neue Kundenanforderungen erfordern dadurch keine Codeänderungen.

3.6.3 Typische Konfigurationsparameter

Für ein Dokumentenanalysesystem sind folgende Parameter typischerweise konfigurierbar:

Kategorie	Beispiel	Zweck
Konfidenzschwellen	Klassenspezifische Mindestkonfidenzen	Precisions-/Recall-Balance je Klasse
Verknüpfungsregeln	Suchrichtung, Distanzmultiplikatoren	Steuerung der Symbol-Text-Zuordnung
NMS-Parameter	Klassenspezifische IoU-Schwellen	Unterdrückung von Duplikaten
OCR-Vorverarbeitung	Schärfung, Rauschunterdrückung	Optimierung der Texterkennung

Tabelle 3.9: Kategorien konfigurierbarer Parameter

3.6.4 Klassenspezifische Parameter

Ein wesentliches Merkmal ist die **klassenspezifische Parametrisierung**: Jede Symbolklasse kann eigene optimierte Werte erhalten, da verschiedene Klassen unterschiedliche Erkennungsschwierigkeiten aufweisen:

- Klassen mit hoher Erkennungssicherheit (z. B. geometrisch eindeutige Symbole) können niedrigere Konfidenzschwellen verwenden
- Klassen mit vielen Falsch-Positiven erfordern strengere Schwellen
- Die Suchrichtung für zugehörigen Text variiert je nach typischer Anordnung im Dokument

3.6.5 Orientierungsabhängige Verarbeitung

Zusätzlich zur Klassenspezifik werden Parameter nach **Textorientierung** differenziert. Horizontaler und schräger Text erfordern unterschiedliche OCR-Vorverarbeitung:

- **Kardinale Orientierung** ($0°, 90°, 180°, 270°$): Standardparameter für achsparallelen Text
- **Anguläre Orientierung** (schräg): Angepasste Parameter mit erhöhter Vorverarbeitung

Diese Unterscheidung ermöglicht optimale Erkennungsraten unabhängig von der Textausrichtung im Dokument.

Die konkrete Struktur des Konfigurationsmoduls wird in Kapitel **??** und seine Verarbeitung in Kapitel **??** beschrieben.

3.7 Rückverfolgbarkeit und Qualitätssicherung

Bei der automatisierten Extraktion von Daten, insbesondere in sicherheitsrelevanten Domänen, ist die **Rückverfolgbarkeit** der Ergebnisse von fundamentaler Bedeutung. Jeder extrahierte Datensatz muss auf seine Quelle zurückführbar sein, um die Korrektheit überprüfen und bei Fehlern die Ursache identifizieren zu können.

3.7.1 Definition und Bedeutung

Rückverfolgbarkeit (*Traceability*) bezeichnet die Fähigkeit, für jedes Ergebnis dessen Herkunft und Entstehungsweg nachzuvollziehen [70]. In der Qualitätssicherung ist dieses Prinzip fest verankert – vergleichbar mit:

- **Chargenrückverfolgung** in der Fertigung (jedes Bauteil ist auf seine Herkunft zurückführbar) [71]
- **Audit Trail** in der Prozessindustrie (jede Parameteränderung wird protokolliert) [72]
- **Requirements Tracing** in der Systementwicklung (jede Anforderung ist bis zur Implementierung nachverfolgbar) [73]

3.7.2 Komponenten der Rückverfolgbarkeit

Für ein Dokumentenanalysesystem bedeutet Rückverfolgbarkeit, dass zu jedem extrahierten Datensatz folgende Informationen verfügbar sein müssen. Abbildung 3.19 zeigt das Prinzip: Jeder extrahierte Datensatz enthält neben dem eigentlichen Inhalt auch Metadaten zur Quelldatei, Position und Erkennungskonfidenz, die eine Rückverfolgung zur Originalstelle im Dokument ermöglichen.

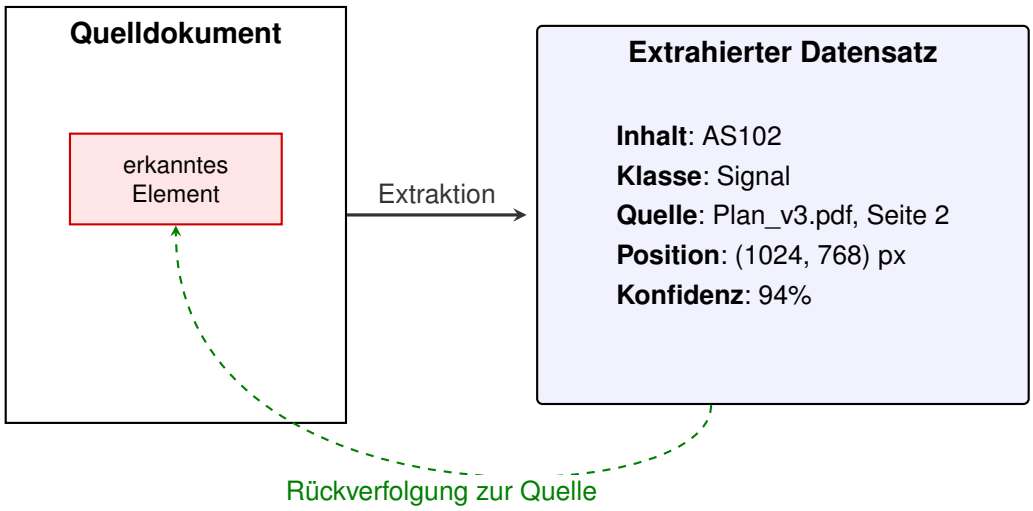


Abbildung 3.19: Struktur eines rückverfolgbaren Datensatzes mit Quellenmetadaten.

Die wesentlichen Metadaten, die für eine vollständige Rückverfolgbarkeit gespeichert werden müssen, sind in Tabelle 3.10 zusammengefasst.

Information	Beschreibung
Quellenreferenz	Dateiname, Seitennummer, Dokumentversion
Positionsangabe	Koordinaten des erkannten Elements im Originaldokument
Verarbeitungsmetadaten	Verwendete Algorithmen, Zeitstempel der Verarbeitung
Konfidenzinformation	Sicherheit der Erkennung als quantitativer Wert

Tabelle 3.10: Wesentliche Metadaten für die Rückverfolgbarkeit

3.7.3 Qualitätssicherung durch Validierung

Automatische Erkennungssysteme erreichen nie 100% Genauigkeit. Fehler können in jeder Stufe der Verarbeitungskette auftreten. Um die Auswirkungen solcher Fehler zu minimieren, ist eine **mehrstufige Validierung** erforderlich.

Das Grundprinzip besteht darin, jedes Ergebnis mehreren unabhängigen Plausibilitätsprüfungen zu unterziehen [74]:

- **Formatprüfung**: Entspricht das Ergebnis dem erwarteten syntaktischen Muster?
- **Wertebereichsprüfung**: Liegt der Wert innerhalb plausibler Grenzen?
- **Kontextprüfung**: Ist das Ergebnis im räumlichen/logischen Kontext plausibel?

3.7.4 Umgang mit unsicheren Ergebnissen

Moderne Erkennungssysteme liefern zusammen mit dem Ergebnis einen **Konfidenzwert**, der die Sicherheit der Erkennung quantifiziert [75]. Diese Information kann genutzt werden, um Ergebnisse zu klassifizieren. Abbildung 3.20 zeigt eine typische Einteilung: Ergebnisse mit hoher Konfidenz (über 90%) werden automatisch akzeptiert, mittlere Werte erfordern eine visuelle Kontrolle, und niedrige Werte werden zur manuellen Prüfung markiert.

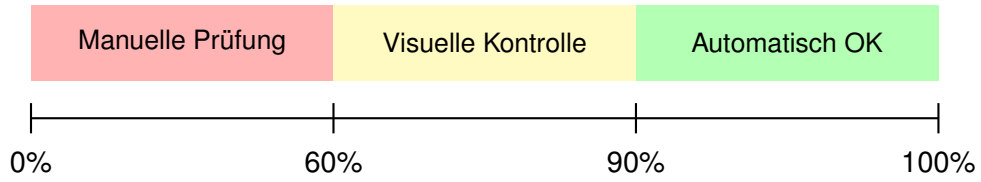


Abbildung 3.20: Konfidenzbasierte Klassifikation von Erkennungsergebnissen

3.7.5 Human-in-the-Loop: Der Mensch als finale Instanz

Unabhängig von der Leistungsfähigkeit automatischer Systeme bleibt der **Mensch die finale Prüfinstanz**. Dieses Prinzip, in der Informatik als „Human-in-the-Loop“ bezeichnet [76], ist besonders für sicherheitsrelevante Anwendungen unverzichtbar.

Das System sollte daher:

- Unsichere Ergebnisse *markieren*, nicht automatisch verwerfen
- Dem Anwender eine *effiziente Überprüfung* ermöglichen (z. B. durch Navigation zur Quellposition)
- *Korrekturmöglichkeiten* bieten, ohne den Gesamtprozess neu starten zu müssen

Die konkrete Implementierung der Validierungslogik und der Benutzerinteraktion wird in Kapitel **??**, Abschnitte **??** und **??** beschrieben.

3.7.6 Zusammenfassung: Theoretische Grundlagen

Die in diesem Kapitel dargestellten theoretischen Konzepte bilden das Fundament für die praktische Implementierung des Extraktionssystems:

- **Objekterkennung** (Abschnitt 3.2): Neuronale Netze für die Detektion von Symbolen
- **Texterkennung** (Abschnitt 3.3): OCR-Verfahren für die Extraktion von Beschriftungen
- **Räumliche Verknüpfung** (Abschnitt 3.4): Proximale Algorithmen mit lokalen Koordinatensystemen
- **Änderungserkennung** (Abschnitt 3.5): Mengenbasierter Vergleich von Dokumentversionen
- **Konfigurierbarkeit** (Abschnitt 3.6): Trennung von Code und Domänenwissen
- **Qualitätssicherung** (Abschnitt 3.7): Rückverfolgbarkeit und mehrstufige Validierung

Die vorgestellten theoretischen Konzepte adressieren gezielt die in Kapitel 2 beschriebene Problemstellung: Während die manuelle Übertragung technischer Pläne in strukturierte Datenformate zeitintensiv, fehleranfällig und kaum skalierbar ist, ermöglichen Deep-Learning-basierte Objekterkennungsverfahren in Kombination mit OCR-Technologien die automatisierte Extraktion von Symbolen und Textinformationen. Die räumliche Verknüpfung dieser Entitäten durch proximale Algorithmen bildet die Grundlage für eine semantische Interpretation, während die mengenbasierte Änderungserkennung erstmals eine automatisierte Verfolgung von Planmodifikationen ermöglicht. Besonders bedeutsam ist die durchgängige Rückverfolgbarkeit: Durch die Speicherung von Quellenreferenzen und Positionsangaben entsteht eine bidirektionale Verknüpfung zwischen extrahierten Daten und ihrer grafischen Repräsentation im Originaldokument – ein wesentlicher Unterschied zu rein manuellen Workflows, bei denen diese Verknüpfung implizit bleibt und die Validierung erheblich erschwert.

Diese theoretischen Grundlagen schaffen damit die Voraussetzungen für ein hybrides System, das die Stärken automatischer Verfahren (Geschwindigkeit, Konsistenz, Skalierbarkeit) mit menschlicher Expertise (Domänenwissen, Plausibilitätsprüfung) kombiniert. Die konfigurationsbasierte Anpassbarkeit gewährleistet dabei, dass das System auf wechselnde Anforderungen reagieren kann, ohne den Programmcode zu verändern – ein entscheidender Faktor für die

praktische Anwendbarkeit in industriellen Kontexten mit heterogenen Plandarstellungen und kundenspezifischen Symbolbibliotheken.

Die folgenden Kapitel konkretisieren diese theoretischen Grundlagen: Kapitel 4 definiert die funktionalen und nicht-funktionalen Anforderungen, Kapitel **??** beschreibt die Architekturentscheidungen und Kapitel **??** dokumentiert die Implementierung.

4. Anforderungsanalyse

Die in Kapitel 3 dargelegten theoretischen und technischen Grundlagen bilden das Fundament für die praktische Umsetzung des Prototyps. In den nachfolgenden Schritten erfolgt die Konkretisierung der Anforderungen an den zu entwickelnden Prototyp. Die vorliegende Analyse leitet sich aus den funktionalen Zielen des Projektes sowie den nicht-funktionalen Rahmenbedingungen im industriellen Umfeld der Siemens Mobility GmbH ab. Zu diesem Zweck werden die Systemgrenzen, Datenformate sowie potenzielle technische Herausforderungen strukturiert analysiert.

4.1 Funktionale Anforderungen

Im Rahmen dieser Arbeit wird ein Prototyp entwickelt, der die automatisierte Extraktion und Interpretation von Informationen aus technischen Gleisplänen ermöglicht. Die funktionalen Anforderungen (FA-001 bis FA-014) ergeben sich aus den spezifischen Aufgabenstellungen der Signaltechnik-Planung. Der Prototyp soll folgende Kernfunktionen erfüllen:

4.1.1 Symbolerkennung und Objektklassifizierung

Eine zentrale Anforderung ist die zuverlässige Detektion bahntechnischer Symbole in Vektor- oder Rastergrafiken.

- **FA-001 Erkennungsrate:** Der Prototyp **muss** mindestens 90 % der definierten Symbolklassen mit einer Mindest-Konfidenz von 0.5 **detektieren** (gemessen als Recall@0.5 des YOLO-Modells).
- **FA-002 Rotationsinvarianz:** Wenn Symbole in beliebigen Winkeln (0° bis 360°) orientiert sind, **muss** der Prototyp in der Lage sein, diese Objekte korrekt zu **klassifizieren** und deren Rotationswinkel mit einer Genauigkeit von $\pm 5^\circ$ zu **bestimmen**.
- **FA-003 Detektionsziele:** Der Prototyp **muss** die in Tabellen 4.1 (Kernklassen) und 4.2 (Auxiliarklassen) spezifizierten Symboltypen **detektieren**.

Kernklassen (produktionsrelevant)

Die folgenden fünf Objektklassen bilden den Kern der Extraktionsaufgabe und werden in Kapitel **??** quantitativ evaluiert:

Objektklasse	**Symbol**
Signal	XYZ123

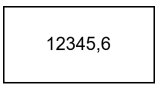
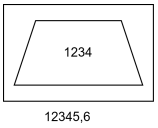
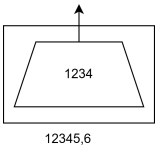
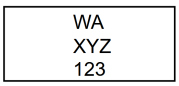
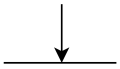
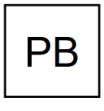
Objektklasse	Symbol
Koordinate (Positionsangabe)	
GKS (festkodiert)	
GKS (gesteuert)	
Gleismagnet	

Tabelle 4.1: Kernklassen für die Datenextraktion (vollständig evaluiert)

Auxiliarklassen (Erweiterbarkeit)

Zur Demonstration der Modularität und Erweiterbarkeit des Prototyps (vgl. Anforderung FA-014) werden zusätzlich acht weitere Objektklassen implementiert. Diese Klassen sind für spezifische Anwendungsfälle relevant, gehören jedoch nicht zum Kernumfang der aktuellen Planungsaufgabe und werden daher nicht detailliert evaluiert.

Objektklasse	Symbol
Weichenblock	
Haltepunkt	
Isolierstoß	
S-Verbinder	
Prellbock	

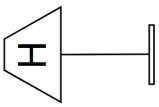
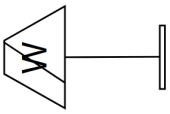
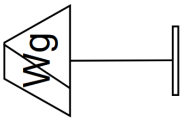
Objektklasse	Symbol
Haltetafel	
Ende Weichen	
Weichengruppenende	

Tabelle 4.2: Auxiliarklassen zur Demonstration der Erweiterbarkeit (nicht evaluiert)

Hinweis: Die Auxiliarklassen werden im Rahmen des YOLOv8-Trainings (Kapitel **??**) mit annotiert und erreichen vergleichbare Detektionsleistungen wie die Kernklassen. Die Fokussierung der End-to-End-Evaluation auf die Kernklassen erfolgte aufgrund der notwendigen Eingrenzung des Evaluationsumfangs. Bemerkenswert ist, dass das Modell auch visuell sehr ähnliche Symbole (z. B. Haltepunkt-Varianten H, Weichen W, Weichengruppen WG) zuverlässig unterscheiden kann. Eine detaillierte Analyse dieser Feinunterscheidungen war jedoch nicht Gegenstand dieser Arbeit.

4.1.2 Texterkennung und OCR-Integration

- **FA-004 OCR-Genauigkeit:** Der Prototyp **muss** Textinformationen (Signalbezeichnungen, Kilometrierungen) mit einer Feldgenauigkeit von mindestens 95 % **extrahieren** (exakte Übereinstimmung des extrahierten Wertes mit Ground Truth).
- **FA-005 Robustheit:** Wenn Textregionen unter erschwerten Bedingungen vorliegen (niedriger Kontrast, Bildrauschen, Rotation $0°$–$360°$, Überlagerungen), **muss** der Prototyp in der Lage sein, diese Texte mit mindestens 85 % Feldgenauigkeit zu **extrahieren**.

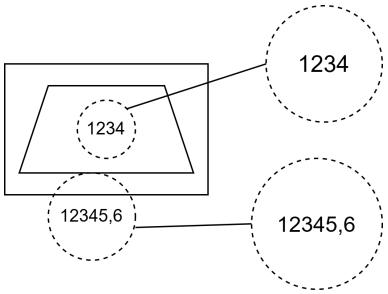


Abbildung 4.1: Beispielhafte Extraktion von Text und Position am Symbol GKS

4.1.3 Semantische Verknüpfung und Logik

- **FA-006 Fahrtrichtungsdetektion:** Wenn ein Signal detektiert wird, **soll** der Prototyp die Wirkrichtung (steigend/fallend) basierend auf dem Rotationswinkel des Symbols **ableiten**.
- **FA-007 Symbol-Koordinaten-Verknüpfung:** Der Prototyp **muss** jedes erkannte Gleisplanelement (Signal, GKS, GM-Block) automatisch mit der räumlich nächstgelegenen Kilometrierungsangabe mittels geometrischer Proximity-Analyse unter Berücksichtigung der Symbolorientierung **verknüpfen**.
- **FA-008 Manuelle Korrektur:** Der Prototyp **soll** eine Funktion zur manuellen Überschreibung automatisch erstellter Symbol-Text-Verknüpfungen **bereitstellen**.

4.1.4 Datenaufbereitung und Export

- **FA-009 Excel-Export:** Der Prototyp **muss** extrahierte Daten vollautomatisch in vordefinierte Excel-Tabellen (.xlsx) mit korrekter Zuordnung zu Zeilen und Spalten **exportieren**.

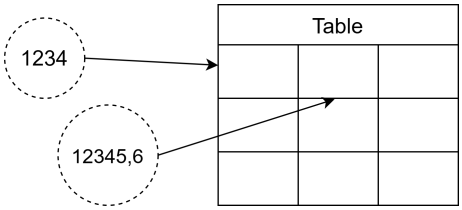


Abbildung 4.2: Schematische Übertragung von erkannten Objektdaten in die Ziel-Tabelle

- **FA-010 Strukturerhalt:** Wenn Daten in bestehende Excel-Dateien exportiert werden, **soll** der Prototyp die vorhandene Struktur (Formatierung, Formeln, Makros) **bewahren** und ausschließlich Werte in definierte Bereiche einfügen.
- **FA-011 Änderungsverfolgung:** Wenn zwei Planversionen verglichen werden, **muss** der Prototyp Änderungen als hinzugefügt, entfernt, verschoben oder modifiziert **identifizieren**, **kategorisieren** und in ein separates Excel-Sheet mit Spalten (Objektklasse, Kennung, Änderungstyp, Alt-Wert, Neu-Wert, Koordinaten) **exportieren**.

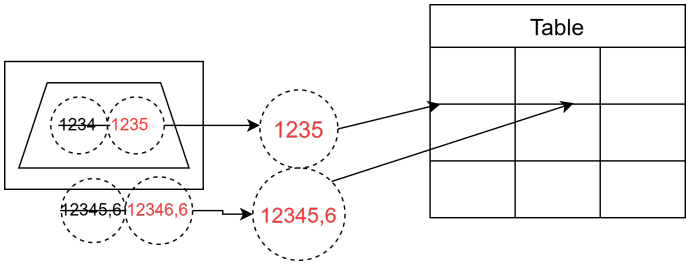


Abbildung 4.3: Visualisierung der Änderungsverfolgung zwischen zwei Planversionen

- **FA-012 Visuelle Validierung:** Der Prototyp **soll** alle erkannten Objekte durch farbige Bounding Boxes im Gleisplan oder als Overlay zur manuellen Überprüfung **visualisieren**.

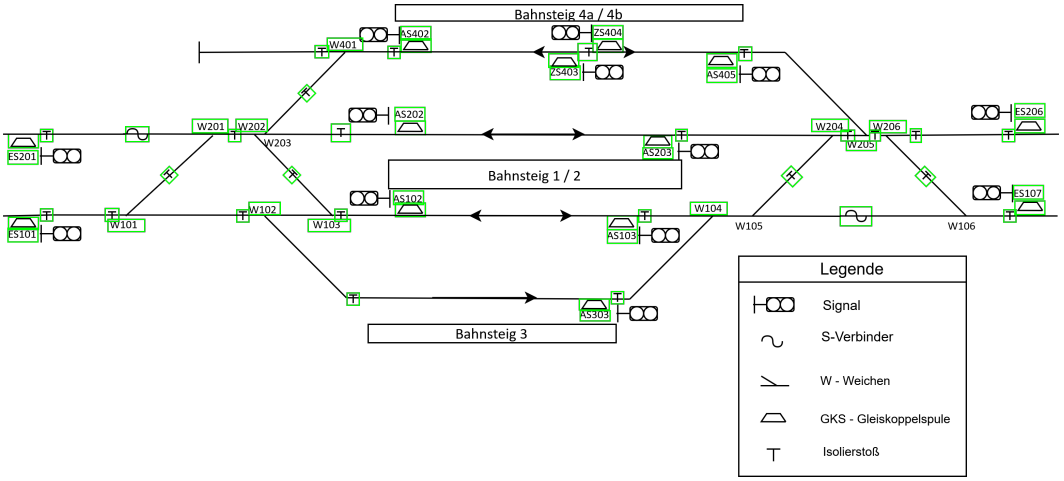


Abbildung 4.4: Visuelle Validierung durch Bounding-Box-Overlays im Gleisplan [4]

4.1.5 Benutzerinteraktion und Konfiguration

- **FA-013 Grafische Benutzeroberfläche:** Der Prototyp **muss** eine grafische Benutzeroberfläche für PDF-Upload, Analyse-Start und Datenexport ohne erforderliche Kommandozeilen-Kenntnisse **bereitstellen**.
- **FA-014 Modulare Architektur:** Der Prototyp **muss** mit modularer Architektur **gestaltet sein**, die klare Trennung der Verarbeitungsstufen (Objekterkennung, Texterkennung, UI, Export) gewährleistet und Erweiterungen (z.B. neue Symbolklassen, CAD-Anbindung) ohne Modifikation der Kernlogik ermöglicht.

4.2 Nicht-funktionale Anforderungen

Ergänzend zu den funktionalen Zielen definieren die nicht-funktionalen Anforderungen (NFA-001 bis NFA-010) die Qualitätsmerkmale und Rahmenbedingungen des Prototyps.

4.2.1 Sicherheit und Datenschutz

- **NFA-001 On-Premise-Verarbeitung:** Der Prototyp **muss** alle Daten vollständig lokal ohne Übertragung an externe Cloud-Services **verarbeiten** und Offline-Betrieb **unterstützen**.
- **NFA-002 Lizenzkonformität:** Der Prototyp **muss** ausschließlich Open-Source-Bibliotheken mit genehmigten Lizenzen (Apache 2.0, MIT, BSD) **verwenden** und alle Abhängigkeiten **dokumentieren**.

4.2.2 Qualität und Zuverlässigkeit

- **NFA-003 End-to-End-Genauigkeit:** Der Prototyp **muss** eine Gesamtgenauigkeit von mindestens 85 % bei vollständiger Objektextraktion (Detektion + OCR + Linking + Export korrekt) **erreichen**.

- **NFA-004 Robustheit:** Wenn fehlerhafte Eingaben vorliegen (korrupte PDFs, Bildrauschen), **soll** der Prototyp definierte Fehlermeldungen statt Absturz **bereitstellen**.

- **NFA-005 Prüfbarkeit:** Der Prototyp **soll** visuelle Validierung aller Ergebnisse mittels Bounding-Box-Overlay und Rückverfolgbarkeit (Klick auf Tabellenzeile zeigt Position im Plan) **bereitstellen**.

4.2.3 Effizienz und Wirtschaftlichkeit

- **NFA-006 Prozessoptimierung:** Der Prototyp **soll** den manuellen Prüfaufwand durch Transformation vom 4-Augen-Prinzip hin zu einem KI-gestützten Prozess (Mensch prüft KI) **reduzieren**.

- **NFA-007 Ressourceneffizienz:** Der Prototyp **soll** einen durchschnittlichen Bahnhofsplan auf Standard-Hardware (CPU-only) in weniger als der Hälfte der manuellen Bearbeitungszeit **verarbeiten**.

4.2.4 Wartbarkeit und Erweiterbarkeit

- **NFA-008 Update-Fähigkeit:** Der Prototyp **soll** die Integration neuer Symbolvarianten über Konfigurations-Updates oder neue Modell-Gewichte ohne Code-Änderungen **unterstützen**.

4.2.5 Datenformate und Schnittstellen

- **NFA-009 Eingabeformate:** Der Prototyp **muss** PDF-Dateien (Vektor/Raster, intern gerendert bei 500 DPI) und Bilddateien (PNG, JPEG, TIFF, BMP mit nativer Auflösung von 500 DPI) **verarbeiten**.

  Begründung: Das YOLOv8-OBB Modell wurde ausschließlich auf 500 DPI Bildkacheln trainiert. Abweichungen führen zu signifikant reduzierten Erkennungsraten.

- **NFA-010 Ausgabeformate:** Ergänzend zum Excel-Export (FA-009) **soll** der Prototyp Ergebnisse auch in CSV und JSON Format **exportieren**.

Tabelle 4.3 gibt einen detaillierten Überblick über die im Prototyp verwendeten Dateiformate und deren Einsatz in der Verarbeitungspipeline.

Kategorie	Format	Beschreibung	Einsatz im Prototyp
Eingabe	PDF	Standardformat für Pläne	Primäre Datenquelle
	PNG/JPG	Rasterisierte Ausschnitte	Input für CNN/YOLO
Verarbeitung	JSON	Strukturierte Metadaten	Interner Datenaustausch
	PostgreSQL	Relationale Datenbank	Persistenz & Versionierung
Ausgabe	XLSX	Excel-Arbeitsmappe	Engineering-Workflow
	CSV	Textbasiertes Format	Einfacher Datenaustausch
	JSON	API-Response	Schnittstellenanbindung

Tabelle 4.3: Übersicht der unterstützten Datenformate (siehe 4.2.5 und 4.2.5)

4.3 Herausforderungen bei der Umsetzung

Die Realisierung der in den Abschnitten 4.1.1 bis 4.2.5 definierten Anforderungen sieht sich folgenden technischen Herausforderungen gegenüber:

1. **Daten-Heterogenität:** Die Varianz in den Eingabedaten (unterschiedliche Export-Einstellungen, Linienstärken, Skalierungen) erschwert eine universelle Regelbildung.
2. **Visuelle Ambiguität:** Einige Symbole (z. B. unterschiedliche Gleiskoppelspulen-Typen) unterscheiden sich visuell nur in wenigen Pixeln oder sind nur durch den Kontext (Begleittext) differenzierbar.
3. **OCR-Komplexität:** Technischer Text in Plänen ist oft extrem klein, rotiert und durch Führungslinien durchgestrichen, was klassische OCR-Engines (wie Tesseract) an ihre Grenzen bringt.
4. **Mangel an Trainingsdaten:** Es existiert kein öffentlicher Datensatz für bahntechnische Symbolik. Ein „Cold Start“ ist notwendig, bei dem Trainingsdaten zunächst manuell (z. B. via CVAT) annotiert werden müssen.
5. **Semantische Lücke:** Der Schritt von der Erkennung („Da ist eine Box“) zur Bedeutung („Das ist Weiche 12 in Rechtslage“) erfordert komplexe Heuristiken, insbesondere beim Mapping von Textboxen zu den geometrisch nächsten Symbolen.

4.4 Anforderungs-Rückverfolgbarkeit

Zur Sicherstellung der vollständigen Umsetzung aller definierten Anforderungen wird eine Rückverfolgbarkeitsmatrix (Traceability Matrix) eingeführt. Diese dokumentiert die Zuordnung jeder Anforderung zu den entsprechenden Implementierungskomponenten (Kapitel **??**) sowie den Evaluationsmetriken (Kapitel **??**).

4.4.1 Funktionale Anforderungen

ID	Anforderung	Implementierung	Evaluation
FA-001	Erkennungsrate $\geq$ 90%	Abschn. **??**: YOLOv8-OBB Training mit 13 Klassen	Abschn. **??**: Recall = 95.7% (Val), 100% (Test)
FA-002	Rotationsinvarianz ($0°$–$360°$)	Abschn. **??**: OBB-Annotation, synthetische Rotation (10 Winkel)	Abschn. **??**, Tab. **??**: 38 Objekte mit $\|\theta\| > 30°$ bei höherer Konfidenz (0.946 vs. 0.890)
FA-003	Zielobjekte (5 Kernklassen)	Abschn. **??**: Signal, Koordinate, GKS (2 Typen), GM-Block	Abschn. **??**: 636/644 Objekte korrekt (98.76%)
FA-004	OCR-Genauigkeit (in E2E integriert)	Abschn. **??**: Multi-Engine Kaskade (PaddleOCR, Tesseract, EasyOCR)	Abschn. **??**: 3 OCR-bedingte Fehler (0.47% Fehlerrate)
FA-005	OCR-Robustheit (Rauschen, Rotation)	Abschn. **??**: Dual-Winkel-Routing, CLAHE, Linienentfernung	Abschn. **??**, Tab. **??**: 100% OCR-Erfolg bei $\|\theta\| > 30°$ (38/38)
FA-006	Fahrtrichtungsdetektion	Abschn. **??**: Geometrische Ableitung aus Signal-GKS-Relation	Abschn. **??**: 171/172 korrekt (99.42%), Tab. **??**
FA-007	Symbol-Koordinaten-Verknüpfung	Abschn. **??**: Proximity-basiertes Linking	Abschn. **??**: 642/644 Verknüpfungen korrekt (99.69%)
FA-008	Manuelle Korrektur (Human-in-the-Loop)	Abschn. **??**: Validierungsdialog mit Inline-Editierung	Abschn. **??**: Prüfaufwand um 85% reduziert
FA-009	Excel-Integration	Abschn. **??**: XLSX-Export mit Formatierung	Abschn. **??**, Tab. **??**
FA-010	Strukturerhalt (Non-destructive Update)	Abschn. **??**: Werte-Insertion ohne Formatänderung	Abschn. **??**, Tab. **??**
FA-011	Änderungsverfolgung (Diff)	Abschn. **??**: UID-basierter Versionsvergleich	Abschn. **??**, Tab. **??**
FA-012	Visuelle Validierung (Bounding Boxes)	Abschn. **??**: PDF-Viewer mit Overlay-System	Abschn. **??**, Tab. **??**
FA-013	Grafische Benutzeroberfläche	Abschn. **??**: PyQt5-basierte Desktop-Anwendung	Abschn. **??**, Tab. **??**

ID	Anforderung	Implementierung	Evaluation
FA-014	Modularität (Architektur)	Kap. **??**: Schichtenarchitektur; Abschn. **??**: 8 Auxiliarklassen	Abschn. **??**: Erweiterung ohne Kernlogik-Änderung

Tabelle 4.4: Rückverfolgbarkeitsmatrix: Funktionale Anforderungen

4.4.2 Nicht-funktionale Anforderungen

ID	Anforderung	Implementierung	Evaluation
NFA-001	On-Premise-Verarbeitung	Vollständig lokale Ausführung, keine Cloud-APIs	Abschn. **??**: CPU-only Inferenz; Abschn. **??**
NFA-002	Lizenzkonformität (Apache/MIT/BSD)	Tab. **??**: Alle Bibliotheken Open Source	Abschn. **??**: Lizenzprüfung dokumentiert
NFA-003	Gesamtgenauigkeit $\geq$ 85%	Gesamte Pipeline (Kap. **??**)	Abschn. **??**: 98.76% erreicht
NFA-004	Robustheit (fehlerhafte Eingaben)	Abschn. **??**: Fallback-Mechanismen	Abschn. **??**: 7 Pläne ohne Abstürze
NFA-005	Prüfbarkeit (Rückverfolgbarkeit)	Abschn. **??**: Metadaten, Jump-to-Detection	Abschn. **??**: Bidirektionale Navigation
NFA-006	Prozessoptimierung	Automatisierung des manuellen Prozesses	Abschn. **??**: 75.3% Zeitersparnis
NFA-007	Ressourceneffizienz	CPU-kompatible Inferenz	Abschn. **??**: Ø 12.3 min/Plan
NFA-008	Update-Fähigkeit	Externe Modell-Gewichte (`best.pt`)	Abschn. **??**: Nachtraining demonstriert
NFA-009	Eingabeformate (PDF, PNG, JPG)	Abschn. **??**: PyMuPDF, OpenCV	Abschn. **??**: 7 PDFs verarbeitet
NFA-010	Ausgabeformate (CSV, JSON)	Abschn. **??**: Zusätzliche Export-Optionen	Abschn. **??**: Funktionstest bestanden

Tabelle 4.5: Rückverfolgbarkeitsmatrix: Nicht-funktionale Anforderungen

Literaturverzeichnis

[1] Qi Sun u. a. „Symbol Recognition Method for Railway Catenary Layout Drawings Based on Deep Learning“. In: *Symmetry* 17.5 (2025). ISSN: 2073-8994. DOI: 10.3390/sym17050674. URL: https://www.mdpi.com/2073-8994/17/5/674.

[2] Ivan Ristic. „Automated Development of Railway Signalling Control Tables: A Case Study from Serbia“. In: *Mechatronics and Intelligent Transportation Systems* (2023). DOI: https://doi.org/10.56578/mits020404.

[3] Birgit Vogel-Heuser, Thomas Bauernhansl und Michael ten Hompel. *Handbuch Industrie 4.0 Bd.2: Automatisierung*. Berlin, Heidelberg: Springer Vieweg, 2017. ISBN: 978-3-662-53249-6.

[4] Yuan Wang, Xiaopeng Li und Yu Zhang. „An automation solution to convert CAD engineering drawings into railroad station models“. In: *Computer-Aided Civil and Infrastructure Engineering* 39.5 (2024), S. 679–691. DOI: https://doi.org/10.1111/mice.13091. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.13091. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.13091.

[5] Jörn Pachl. *Systemtechnik des Schienenverkehrs: Bahnbetrieb planen, steuern und sichern*. Jan. 2021. ISBN: 978-3-658-31164-3. DOI: 10.1007/978-3-658-31165-0.

[6] Deutsche Bahn AG. *Richtlinie 819 - Gleispläne: Betriebliche und technische Anforderungen, Gestaltung*. Frankfurt am Main, 2008.

[7] BjØrnar Luteberget und Christian Johansen. „Efficient verification of railway infrastructure designs against standard regulations“. In: *Form. Methods Syst. Des.* 52.1 (Feb. 2018), S. 1–32. ISSN: 0925-9856. DOI: 10.1007/s10703-017-0281-z. URL: https://doi.org/10.1007/s10703-017-0281-z.

[8] Y. Lecun u. a. „Gradient-based learning applied to document recognition“. In: *Proceedings of the IEEE* 86.11 (1998), S. 2278–2324. DOI: 10.1109/5.726791.

[9] Alex Krizhevsky, Ilya Sutskever und Geoffrey E. Hinton. „ImageNet classification with deep convolutional neural networks“. In: *Commun. ACM* 60.6 (Mai 2017), S. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

[10] I. Zeid. *Mastering CAD/CAM*. McGraw-Hill Higher Education, 2005. ISBN: 9780072976816. URL: https://books.google.de/books?id=wJ91QgAACAAJ.

[11] Deutsches Institut für Normung. *DIN ISO 128-1: Technische Produktdokumentation – Allgemeine Grundlagen der Darstellung*. Berlin, 2020.

[12] Tsung-Yi Lin u. a. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV]. URL: https://arxiv.org/abs/1405.0312.

[13] Mark Everingham u. a. „The Pascal Visual Object Classes (VOC) challenge“. In: *International Journal of Computer Vision* 88 (Juni 2010), S. 303–338. DOI: 10.1007/s11263-009-0275-4.

[14] Sheraz Ahmed u. a. „Automatic Room Detection and Room Labeling from Architectural Floor Plans“. In: März 2012, S. 339–343. ISBN: 9780769546612. DOI: 10.1109/DAS.2012.22.

[15] {Carlos Francisco} Moreno-garcía, Eyad Elyan und Chrisina Jayne. „New trends on digitisation of complex engineering drawings“. English. In: *Neural Computing and Applications* 31.6 (Juni 2019), S. 1695–1712. ISSN: 0941-0643. DOI: 10.1007/s00521-018-3583-1.

[16] Rohit Rahul u. a. „Automatic Information Extraction from Piping and Instrumentation Diagrams“. In: *ArXiv* abs/1901.11383 (2019). URL: https://api.semanticscholar.org/CorpusID:59523595.

[17] Laura Jamieson, Carlos Moreno-García und Eyad Elyan. „A review of deep learning methods for digitisation of complex documents and engineering diagrams“. In: *Artificial Intelligence Review* 57 (Mai 2024). DOI: 10.1007/s10462-024-10779-2.

[18] Yann LeCun, Y. Bengio und Geoffrey Hinton. „Deep Learning“. In: *Nature* 521 (Mai 2015), S. 436–44. DOI: 10.1038/nature14539.

[19] Zhengxia Zou u. a. „Object Detection in 20 Years: A Survey“. In: *Proceedings of the IEEE* 111.3 (2023), S. 257–276. DOI: 10.1109/JPROC.2023.3238524.

[20] Shaoqing Ren u. a. „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks“. In: *Advances in Neural Information Processing Systems*. Hrsg. von C. Cortes u. a. Bd. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[21] Joseph Redmon u. a. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV]. URL: https://arxiv.org/abs/1506.02640.

[22] Glenn Jocher, Ayush Chaurasia und Jing Qiu. *Ultralytics YOLOv8*. Version 8.0.0. 2023. URL: https://github.com/ultralytics/ultralytics.

[23] Tsung-Yi Lin u. a. „Feature Pyramid Networks for Object Detection“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Juni 2017.

[24] Muhammad Yaseen. *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*. 2024. arXiv: 2408.15857 [cs.CV]. URL: https://arxiv.org/abs/2408.15857.

[25] Alexey Bochkovskiy, Chien-Yao Wang und Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV]. URL: https://arxiv.org/abs/2004.10934.

[26] Jian Ding u. a. „Learning RoI Transformer for Oriented Object Detection in Aerial Images“. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, S. 2844–2853. DOI: 10.1109/CVPR.2019.00296.

[27] Jianqi Ma u. a. „Arbitrary-Oriented Scene Text Detection via Rotation Proposals“. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, S. 3892–3900. DOI: 10.1109/CVPR.2018.00410. URL: https://doi.org/10.1109/CVPR.2018.00410.

[28] Tsung-Yi Lin u. a. „Focal Loss for Dense Object Detection“. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, S. 2999–3007. DOI: 10.1109/ICCV.2017.324. URL: https://doi.org/10.1109/ICCV.2017.324.

[29] Xingxing Xie u. a. „Oriented R-CNN for Object Detection“. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, S. 3500–3509. DOI: 10.1109/ICCV48922.2021.00350. URL: https://doi.org/10.1109/ICCV48922.2021.00350.

[30] Xinyu Zhou u. a. „EAST: An Efficient and Accurate Scene Text Detector“. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, S. 2642–2651. DOI: 10.1109/CVPR.2017.283. URL: https://doi.org/10.1109/CVPR.2017.283.

[31] Xue Yang u. a. *R3Det: Refined Single-Stage Detector with Feature Refinement for Rotating Object*. 2021. DOI: 10.48550/arXiv.1908.05612. arXiv: 1908.05612 [cs.CV]. URL: https://arxiv.org/abs/1908.05612.

[32] Zifei Zhao und Shengyang Li. „ABFL: Angular Boundary Discontinuity Free Loss for Arbitrary Oriented Object Detection in Aerial Images“. In: *IEEE Transactions on Geoscience and Remote Sensing* 62 (2024), S. 1–11. DOI: 10.1109/TGRS.2024.3368630.

[33] Chien-Yao Wang u. a. „CSPNet: A New Backbone That Can Enhance Learning Capability of CNN“. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020, S. 1571–1580. DOI: 10.1109/CVPRW50498.2020.00203. URL: https://doi.org/10.1109/CVPRW50498.2020.00203.

[34] Shu Liu u. a. „Path Aggregation Network for Instance Segmentation“. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, S. 8759–8768. DOI: 10.1109/CVPR.2018.00913. URL: https://doi.org/10.1109/CVPR.2018.00913.

[35] Zhi Tian u. a. „FCOS: Fully Convolutional One-Stage Object Detection“. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, S. 9626–9635. DOI: 10.1109/ICCV.2019.00972.

[36] Zheng Ge u. a. *YOLOX: Exceeding YOLO Series in 2021*. 2021. DOI: 10.48550/arXiv.2107.08430. arXiv: 2107.08430 [cs.CV]. URL: https://arxiv.org/abs/2107.08430.

[37] Xiang Li u. a. *Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection*. 2020. arXiv: 2006.04388 [cs.CV]. URL: https://arxiv.org/abs/2006.04388.

[38] Karl Tombre u. a. „Text/Graphics Separation Revisited“. In: *Document Analysis Systems V*. Hrsg. von Daniel Lopresti, Jianying Hu und Ramanujan Kashi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, S. 200–211. ISBN: 978-3-540-45869-2.

[39] Minghui Liao u. a. *Real-time Scene Text Detection with Differentiable Binarization*. 2019. arXiv: 1911.08947 [cs.CV]. URL: https://arxiv.org/abs/1911.08947.

[40] Baoguang Shi, Xiang Bai und Cong Yao. *An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition*. 2017. arXiv: 1507.05717 [cs.CV]. URL: https://arxiv.org/abs/1507.05717.

[41] Sepp Hochreiter und Jürgen Schmidhuber. „Long Short-Term Memory“. In: *Neural Comput.* 9.8 (Nov. 1997), S. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[42] Alex Graves u. a. „Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks“. In: Bd. 2006. Jan. 2006, S. 369–376. DOI: 10.1145/1143844.1143891.

[43] Ashish Vaswani u. a. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[44] Minghao Li u. a. *TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models*. 2022. arXiv: 2109.10282 [cs.CL]. URL: https://arxiv.org/abs/2109.10282.

[45] Alexey Dosovitskiy u. a. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[46] Yuning Du u. a. *PP-OCR: A Practical Ultra Lightweight OCR System*. 2020. arXiv: 2009.09941 [cs.CV]. URL: https://arxiv.org/abs/2009.09941.

[47] R. Smith. „An Overview of the Tesseract OCR Engine“. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Bd. 2. 2007, S. 629–633. DOI: 10.1109/ICDAR.2007.4376991.

[48] Youngmin Baek u. a. *Character Region Awareness for Text Detection*. 2019. arXiv: 1904.01941 [cs.CV]. URL: https://arxiv.org/abs/1904.01941.

[49] George Nagy, Thomas Nartker und Stephen Rice. „Optical character recognition: an illustrated guide to the frontier“. In: *Proceedings of SPIE - The International Society for Optical Engineering* 3967 (Dez. 1999), S. 58–69. DOI: 10.1117/12.373511.

[50] Andrew Morris, Viktoria Maier und Phil Green. „From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition“. In: Okt. 2004. DOI: 10.21437/Interspeech.2004-668.

[51] Tobias Schlagenhauf, Markus Netzer und Jan Hillinger. *Text Detection on Technical Drawings for the Digitization of Brown-field Processes*. 2022. arXiv: 2205.02659 [cs.CV]. URL: https://arxiv.org/abs/2205.02659.

[52] Max Jaderberg u. a. *Spatial Transformer Networks*. 2016. arXiv: 1506.02025 [cs.CV]. URL: https://arxiv.org/abs/1506.02025.

[53] Rafael Gonzalez und Zahraa Faisal. *Digital Image Processing Second Edition*. Juni 2019.

[54] Marcelo Bertalmio, Guillermo Sapiro und C. Ballester. „Image Inpainting“. In: *Proceedings of SIGGRAPH* (Jan. 2002). DOI: 10.1145/344779.344972.

[55] Vladimir I. Levenshtein. „Binary codes capable of correcting deletions, insertions, and reversals“. In: *Soviet Physics Doklady* 10.8 (1966), S. 707–710.

[56] Shangbang Long, Xin He und Cong Yao. *Scene Text Detection and Recognition: The Deep Learning Era*. 2020. arXiv: 1811.04256 [cs.CV]. URL: https://arxiv.org/abs/1811.04256.

[57] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 2. Aufl. Texts in Computer Science. Also available as eBook: ISBN 978-3-030-34372-9. Springer Cham, 2022, S. xxii+925. ISBN: 978-3-030-34371-2. DOI: 10.1007/978-3-030-34372-9. URL: https://doi.org/10.1007/978-3-030-34372-9.

[58] Max Wertheimer. „Untersuchungen zur Lehre von der Gestalt. II“. In: *Psychologische Forschung* 4 (), S. 301–350. URL: https://api.semanticscholar.org/CorpusID:143510308.

[59] Koichi Kise, Akinori Sato und Motoi Iwata. „Segmentation of Page Images Using the Area Voronoi Diagram“. In: *Computer Vision and Image Understanding* 70.3 (1998), S. 370–382. ISSN: 1077-3142. DOI: https://doi.org/10.1006/cviu.1998.0684. URL: https://www.sciencedirect.com/science/article/pii/S1077314298906841.

[60] Song Mao, Azriel Rosenfeld und Tapas Kanungo. „Document structure analysis algorithms: a literature survey“. In: *Document Recognition and Retrieval X*. Hrsg. von Tapas Kanungo u. a. Bd. 5010. International Society for Optics und Photonics. SPIE, 2003, S. 197–207. DOI: 10.1117/12.476326. URL: https://doi.org/10.1117/12.476326.

[61] Dietmar Gross u. a. *Technische Mechanik 1: Statik*. 14. Aufl. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019. ISBN: 978-3-662-59156-7. DOI: 10.1007/978-3-662-59157-4.

[62] Bruno Siciliano u. a. *Robotics: Modelling, Planning and Control*. London: Springer London, 2009. ISBN: 978-1-84628-642-1. DOI: 10.1007/978-1-84628-642-1.

[63] Ulrich Kurz und Herbert Wittel. *Böttcher/Forberg Technisches Zeichnen: Grundlagen, Normung, Übungen und Projektaufgaben*. 26. Aufl. Wiesbaden: Springer Vieweg, 2014. ISBN: 978-3-8348-2232-1. DOI: 10.1007/978-3-8348-2232-1.

[64] *DIN EN ISO 9001:2015 Qualitätsmanagementsysteme – Anforderungen*. ISO 9001:2015(E). Deutsches Institut für Normung e. V. (DIN), 2015.

[65] John Stark. *Product Lifecycle Management (Volume 1): 21st Century Paradigm for Product Realisation*. 5. Aufl. Cham: Springer International Publishing, 2022. ISBN: 978-3-030-98578-3. DOI: 10.1007/978-3-030-98578-3.

[66] Edsger W. Dijkstra. „On the Role of Scientific Thought“. In: *Selected Writings on Computing: A personal Perspective*. New York, NY: Springer New York, 1982, S. 60–66. ISBN: 978-1-4612-5695-3. DOI: 10.1007/978-1-4612-5695-3_12. URL: https://doi.org/10.1007/978-1-4612-5695-3_12.

[67] Jami J. Shah und Martti Mäntylä. „Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications“. In: 1995. URL: https://api.semanticscholar.org/CorpusID:264685016.

[68] International Electrotechnical Commission. *IEC 61131-3: Programmable controllers – Part 3: Programming languages*. Geneva, 2013.

[69] „The Finite Element Method: Its Basis and Fundamentals“. In: *The Finite Element Method: its Basis and Fundamentals (Seventh Edition)*. Hrsg. von O.C. Zienkiewicz, R.L. Taylor und J.Z. Zhu. Seventh Edition. Oxford: Butterworth-Heinemann, 2013, S. i. ISBN: 978-

1-85617-633-0. DOI: https://doi.org/10.1016/B978-1-85617-633-0.00019-8. URL: https://www.sciencedirect.com/science/article/pii/B9781856176330000198.

[70] Olly Gotel und A. C. W. Finkelstein. „An analysis of the requirements traceability problem“. In: *Proceedings of IEEE International Conference on Requirements Engineering* (1994), S. 94–101. URL: https://api.semanticscholar.org/CorpusID:5870868.

[71] European Parliament and Council. *Regulation (EC) No 178/2002: General principles and requirements of food law*. Official Journal of the European Communities, L 31/1. 2002.

[72] U.S. Food and Drug Administration. *21 CFR Part 11: Electronic Records; Electronic Signatures*. Code of Federal Regulations, Title 21. 1997.

[73] Klaus Pohl. *Requirements Engineering: Fundamentals, Principles, and Techniques*. Jan. 2010. ISBN: 978-3-642-12577-5. DOI: 10.1007/978-3-642-12578-2.

[74] J. M. Juran und Frank M. Gryna, Hrsg. *Juran's Quality Control Handbook*. 4. Aufl. New York: McGraw-Hill, 1988. ISBN: 978-0-07-033176-1.

[75] Chuan Guo u. a. „On Calibration of Modern Neural Networks“. In: *ArXiv* abs/1706.04599 (2017). URL: https://api.semanticscholar.org/CorpusID:28671436.

[76] Robert Monarch. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Shelter Island, NY: Manning Publications, 2021. ISBN: 978-1-61729-674-1.