Hochschule für Technik, Wirtschaft und Kultur Leipzig (FH)
Fachbereich Informatik, Mathematik und Naturwissenschaften

Bachelorarbeit

Einsatz von Algorithmen der Photogrammetrie und Bildverarbeitung zur Einblendung spezifischer Lichtraumprofile in Videosequenzen

Vorgelegt von

Matthias Jauernig

Leipzig, September 2006

Betreuer der Firma CCC GmbH: Dipl.-Inf. Markus Maspfuhl
Betreuender Professor: Prof. Dr.rer.nat.habil. Karl-Udo Jahn

„Eine Abschlussarbeit ist wie ein Marathon.
Übermotiviert düst man los, schleppt sich dann weiter und
begegnet im letzten Drittel dem Mann mit dem Hammer.
Doch hat man sein Ziel erreicht, so lässt
der Stolz alle Schmerzen vergessen.“

Matthias Jauernig, September 2006

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die Bildverarbeitung als Teilbereich der Informatik profitiert in den letzten Jahren zunehmend von immer schnelleren Rechnern. Kommerziell werden ihre Algorithmen z.B. bei der Fingerabdruck- oder Gesichtserkennung, in der Robotik und bei der Überwachung von Produktionsprozessen eingesetzt. Als aktuelles Anwendungsgebiet sei hier der Robocup-Wettbewerb erwähnt, bei dem die Bildverarbeitung einen fundamentalen Bestandteil darstellt, auf dem die Interpretation der jeweiligen Situation und die Reaktion im entsprechenden Fall aufbaut. Sehr viel Aufmerksamkeit erhält zur Zeit zudem die Entwicklung von Systemen zur Fahrerunterstützung im Straßenverkehr, z.B. die *Spurwechselwarnung*, bei der ein Fahrer im Falle des unbeabsichtigten Verlassens der Fahrbahn ab einer gewissen Geschwindigkeit gewarnt wird. Weitere aktuelle Forschungsgebiete in diesem Bereich sind Fußgänger-, Hindernis- und Verkehrszeichenerkennung sowie autonome Fahrzeuge, welche durch ihre Echtzeitanforderungen hohe Ansprüche an die zugrunde liegenden Algorithmen stellen. Ein anderes benötigtes Hilfsmittel hierbei stellt die Photogrammetrie dar, durch welche ein mathematischer Zusammenhang zwischen Punkten in der realen Welt und deren perspektivischer Abbildung in einem aufgenommenen Bild hergestellt werden kann. Dadurch lassen sich beispielsweise Entfernungen in einer aufgenommenen Szene messen, was entscheidend ist für die situationsabhängige Reaktion eines Systems.

Auch im Bereich der Überwachung und Instandhaltung technischer Systeme kommen verstärkt rechnerbasierte Lösungen zum Einsatz, die Teilbereiche der Sensorik und Bildverarbeitung umfassen und eine effiziente Kontrolle ermöglichen. So hat sich die europaweit agierende Firma *Eurailscout* auf die Instandhaltung von Bahnanlagen (speziell von Fahrwegen) spezialisiert. Dies stellt eine wichtige Aufgabe dar, wird dadurch doch eine frühzeitige Erkennung von Problemen und die Vermeidung von Folgeschäden erreicht. Nur so kann die Sicherheit der Fahrgäste und die Konkurrenzfähigkeit der Bahn mit anderen Verkehrssystemen gewährleistet werden. Eurailscout deckt bei der Instandhaltung viele Bereiche ab, angefangen bei der Fahrdrahtgeometrie von Oberleitungen über die Beschaffenheit der Gleise bis hin zur Analyse des Gleisumfeldes und der Vegetationskontrolle. In den letzten Bereich fällt vor allem die Auswertung von Messvideos, um in den Fahrweg hineinragende Objekte wie z.B. umgeknickte Äste erkennen zu können. Dazu wird eine Art Maske, das so genannte *Lichtraumprofil*, in das Bild eingezeichnet. Dieses stellt in groben Umrissen die Höchstmaße eines Zuges dar und gibt somit den Bereich an, der von anderen Objekten nicht geschnitten werden darf.

1.1 Zielsetzung der Arbeit

Das Ziel dieser Arbeit stellt die Entwicklung eines Softwaremoduls sowie einer darauf aufbauenden Demonstrationsapplikation dar, mit denen die Instandhaltungsaufgabe von Fahrwegen der Bahn unterstützt werden soll. Dazu sind Messvideos von Inspektionsfahrten auszuwerten, um in einer definierten Entfernung ein Lichtraumprofil einzuzeichnen. Solch ein Video wird durch eine Kamera aufgenommen, die im Führerhaus des Inspektionszuges angebracht ist und den vorausliegenden Fahrweg aufnimmt. Abbildung 1.1 zeigt eine typische Szene. Das Lichtraumprofil soll maßstäblich eingezeichnet werden und direkt auf der aktuellen Fahrspur aufsetzen, sodass ein Inspekteur leicht eine Entscheidung darüber treffen kann, ob ein Objekt zu weit in den Fahrweg hinein ragt. Dabei wird keine Forderung nach Echtzeitfähigkeit des zu entwickelnden Systems gestellt: die Videos sollen entweder nach der Fahrt Bild für Bild analysiert werden oder noch während der Fahrt, wobei dann aber in Abhängigkeit der vorhandenen Rechnerleistung nur solche Bilder ausgewertet werden sollen, die direkt nach Abschluss der vorherigen Berechnung im laufenden Video anliegen.

Abbildung 1.1: Typische Aufnahme im Messvideo

1.2 Kernprobleme und Zerlegung der Arbeit

Die generelle Vorgehensweise zur Lösung des Problems gestaltet sich wie folgt: zunächst ist eine *Kamerakalibrierung* zu erstellen, um zum einen die Entfernungsmessung im Bild realisieren zu können und zum anderen den Bildbereich effizient zu begrenzen. Diese Kalibrierung kann einmal zu Beginn der Auswertung durchgeführt und dann für den Rest des Videos benutzt werden. Ebenfalls nur einmal wird eine Bildraumbeschränkung anhand von A-priori-Informationen

wie minimaler Kurvenradius, maximale Entfernung des Lichtraumprofils etc. durchgeführt. Für jedes auszuwertende Bild muss dann nur noch dieser Bereich betrachtet werden. Um das Ziel der Einzeichnung des Lichtraumprofils auf der aktuellen Fahrspur zu erreichen, muss für jedes Videobild anschließend die aktuelle Fahrspur extrahiert werden, was das Kernproblem dieser Arbeit darstellt. Dazu ist ein geeigneter Algorithmus zu entwickeln und auf seine Tauglichkeit hin zu überprüfen. Die dabei möglichen Bildverarbeitungstechniken sind gegeneinander abzuwägen und zu beschreiben. Weiterhin gilt es worst-case-Szenarien zu erläutern und Bedingungen an das aufgenommene Videobild zu formulieren.

Aus dieser Überlegung ergeben sich folgende Kernfragen:

1. Wie lässt sich eine geeignete Kamerakalibrierung durchführen?
2. Wie kann ein Algorithmus zur Schienenerkennung aussehen?
3. Wie lässt sich die Berechnungszeit verkürzen?
4. Welche Bildverarbeitungs-Techniken sind sinnvoll, welche nicht?
5. Welche Voraussetzungen werden an das aufgenommene Bild und die zugrunde liegende Rechnerarchitektur gestellt?

Frage 1 wird in Kapitel 2 abgehandelt, welches die Photogrammetrie und damit verbundene Algorithmen zur Messung in Bildern beinhaltet. Die hier vorgestellte Methode zur Kamerakalibrierung stellt zudem die Basis für den entwickelten Algorithmus zur Schienenerkennung dar.

Die Fragen 2 und 3 betrachten sowohl den Algorithmus zur Schienenerkennung als auch dessen effiziente Realisierung. Die in Kapitel 3 behandelte Begrenzung des Suchbereichs stellt dabei zusätzlich eine Grundvoraussetzung für den in Kapitel 4 vorgestellten Algorithmus dar. Dieser stützt sich zudem wesentlich auf Bildverarbeitungsalgorithmen wie optimale Kantendetektion und die Hough-Transformation, welche ebenfalls dort beschrieben werden. Zudem findet eine Abgrenzung nützlicher und überflüssiger Algorithmen statt, womit auf die 4. Frage eingegangen wird. Weiterhin werden Testfälle vorgestellt und Bedingungen an die Videobilder formuliert, wie sie von Frage 5 verlangt werden.

Kapitel 5 betrachtet die Implementierung der vorgestellten Algorithmen und geht kurz auf die einzelnen Komponenten des entwickelten Systems ein.

Mit Kapitel 6 soll ein abschließender Rahmen gegeben werden, der Schlussbemerkungen sowie einen Ausblick enthält.

1.3 Verwandte Problemstellungen in der Literatur

Eine Recherche nach Literatur direkt zum Problem der Fahrwegerkennung bei schienenbasierten Verkehrsmitteln wie Zügen oder auch Straßenbahnen hatte leider keinen Erfolg, allerdings sind

Parallelen mit der Spurerkennung und -verfolgung im Straßenverkehr offensichtlich. Hier wurden in den letzten Jahren große Fortschritte erzielt, sodass einige Systeme wie die Spurwechselwarnung sogar schon in serienreifen Fahrzeugen Einzug gefunden haben. Abschnitt 4.1 geht auf einzelne Algorithmen und die entsprechenden Literaturquellen ein. Hier soll zunächst ein Vergleich mit der Spurerkennung im Automobilbereich Aufschluss darüber geben, inwieweit beide Probleme Äquivalenzen aufweisen und sich publizierte Ergebnisse auf diese Arbeit übertragen lassen.

1.3.1 Vergleich zur Fahrspurerkennung im Straßenverkehr

Zunächst gilt es die Gemeinsamkeiten zu ermitteln. Beide Probleme beinhalten die Extraktion der Fahrspur aus einem aufgenommenen Videobild, welches den Bereich vor dem Fahrzeug zeigt. Dabei müssen nicht nur Geraden, sondern auch Kurven erkannt und mathematisch formuliert werden können. Hierbei auftretende Probleme können beispielsweise schlechte Sichtverhältnisse (z.B. bei Regen, Nebel oder Schnee) und diffuser Lichteinfall (Sonnenreflexionen, Schatten auf der Fahrspur etc.) sein. Auch die perspektivische Abbildung muss modelliert werden können, um in der Lage zu sein Entfernungen im Bild zu messen. Ebenso wird in beiden Fällen die Fahrspur links und rechts begrenzt, einerseits durch Schienen im vorliegenden Problem, andererseits durch Fahrbahnmarkierungen auf Straßen. Die Grauwert-Kontraste, welche sich dadurch ergeben, sind die wichtigsten Hilfsmittel zur Extraktion der Fahrspur.

Auf der anderen Seite lassen sich auch viele Unterschiede festhalten. Straßen stellen prinzipiell eine Verallgemeinerung von Schienen dar und so sind die Anforderungen an die Fahrspurerkennung bei Straßen allgemein als höher anzusehen. Viele Probleme die im Straßenverkehr auftreten, können bei Schienensystemen ausgeschlossen werden. Vor allem die unterschiedlichen Ausprägungen von Fahrbahnen verkomplizieren eine robuste Erkennung. Stadt- und Landstraßen sowie Autobahnen stellen jeweils spezielle Anforderungen an das Erkennungssystem, variable Straßenbreiten verhindern zudem eine strenge Begrenzung des zu untersuchenden Fahrbahnbereiches. Weiterhin werden zur Verkehrskontrolle oftmals spezielle Fahrbahnmarkierungen eingesetzt, welche von den üblichen Begrenzungen abweichen. Nicht zuletzt sind es auch mögliche Spurwechsel und Hindernisse im Fahrweg (z.B. andere Autos), die bei einer effizienten Erkennung berücksichtigt werden müssen. Fahrerassistenzsysteme und autonome Fahrzeuge stellen außerdem die Forderung nach einer Echtzeitverarbeitung, welche dort ein unerlässliches Kriterium darstellt (schließlich möchte niemand erst dann über das Verlassen der Fahrspur informiert werden, wenn er im nächsten Graben oder am übernächsten Baum gelandet ist - selbt ein autonomes Vehikel nicht).

Andererseits lässt die Erkennung von Straßensystemen auch einige Vereinfachungen zu, die bei Schienen nicht gegeben sind. So sind heuristische Algorithmen denkbar, die sich auf die Annahme stützen, dass eine Straße einen relativ homogenen Farbbereich darstellt und sich Fahrbahnmarkierungen als helle Stellen abheben. Schienen haben keine solche Eigenschaft, sie können

sich je nach Lichtverhältnissen heller oder dunkler von der Umgebung abheben. Bei der vorliegenden Problemstellung ist zudem die Genauigkeit der Erkennung hinsichtlich einer exakten Einzeichnung des Lichtraumprofils von großer Bedeutung. Für autonome Fahrzeuge und weitere Anwendungen im Automobilbereich genügt hier meist eine Ausgleichslösung, durch die lediglich eine grobe Extraktion der Fahrspur stattfindet. Eines der größten Probleme stellt allerdings das Störverhalten der Umgebung auf die bekannten Kantendetektoren dar, wie noch zu sehen sein wird. Dies verhindert den Einsatz vieler guter Algorithmen, wie sie z.B. in [6, 27] vorgestellt werden (mehr dazu in Abschnitt 4.1).

Das Fazit des Vergleichs lautet, dass sich bereits entwickelte Algorithmen nur partiell auf diese Arbeit übertragen lassen. Teile wie die mathematische Modellierung des Fahrweges (siehe auch 4.6.1) sind zwar übertragbar, bei den konkreten Algorithmen machen sich dann jedoch die Unterschiede zwischen den beiden Problemen stark bemerkbar. Deshalb wurde in dieser Arbeit ein eigener Algorithmus entwickelt, der berücksichtigt, dass viel A-priori-Wissen ausgenutzt werden kann, um den tatsächlichen Schienenverlauf zu extrahieren. Kapitel 4 befasst sich ausführlich damit.

2 Einsatz von Photogrammetrie

Das Fachgebiet der Photogrammetrie beschäftigt sich vorrangig mit der Rekonstruktion von Form bzw. Lage räumlicher Objekte aus perspektivischen Abbildungen wie Fotografien, also *ohne* direkte Berührung der betreffenden Gegenstände. Deshalb spricht man bei dieser Art von Informationsgewinnung häufig auch von *Fernerkundung*. Hauptanwendungsgebiet der Photogrammetrie stellt das Vermessen von Objekten dar, so z.B. in der Luftbildphotogrammetrie bei der Erzeugung topographischer Karten oder in der Nahbereichsphotogrammetrie bei der Präzisionsvermessung von Bauten, Deformationsmessungen oder auch Bauüberwachungssystemen. In den letzten Jahren kamen (bedingt durch die Entwicklung im Digitalrechner-Bereich) vermehrt auch Anwendungsgebiete wie die situationsbezogene kriminaltechnische Rekonstruktion von Unfällen oder Tatvorgängen sowie Anwendungen in Medizin, industrieller Messtechnik und sogar Kinematographie hinzu. Ebenso ist eine immer ausgeprägtere Verschmelzung mit Bildverarbeitungs-Algorithmen erkennbar. Eine Übersicht schon länger typischer Anwendungen liefert z.B. [13].

Photogrammetrie im Sinne der 3D-Rekonstruktion spielt in diesem Kontext keine Rolle, vielmehr sollen Algorithmen der Photogrammetrie dazu eingesetzt werden, um einen Zusammenhang zwischen 3D-Weltkoordinaten und 2D-Bildkoordinaten herzustellen. So wird die Umrechnung von Welt- in Bildpunkte benötigt, um den zu untersuchenden Bildbereich weitestgehend eingrenzen zu können und Vorwissen für den entwickelten Algorithmus auszunutzen (minimaler Kurvenradius, Parallelität und Abstand der Schienen, ...). Eine Umrechnung in Weltkoordinaten (bei Festlegung einer Bezugsebene) ist weiterhin notwendig, um das Lichtraumprofil in der definierten Entfernung maßstäblich einzeichnen zu können.

In diesem Kapitel soll zunächst das allgemein verwendete Kameramodell dargestellt werden. Auf diesem aufbauend wird dann der Abbildungsvorgang beschrieben, auf welchem viele photogrammetrische Verfahren basieren. Schließlich wird mit der Direkten Linearen Transformation (DLT) eine Kamera-Kalibrierungs-Technik eingeführt, welche am häufigsten in Messsystemen wieder zu finden ist.

2.1 Verwendete Symbole

X_w, Y_w, Z_w	Weltkoordinaten
X_{w0}, Y_{w0}, Z_{w0}	Ursprung des Weltkoordinatensystems
P_w	Punkt im Weltkoordinatensystem

X_k, Y_k, Z_k	Kamerakoordinaten
P_k	Punkt im Kamerakoordinatensystem
x, y	Bildkoordinaten
x_0, y_0	Koordinaten des Bildhauptpunktes
P	Punkt im Bildkoordinatensystem
$\mathcal{P}_w^i$	Lage des i. Passpunktes im Weltkoordinatensystem
$\mathcal{P}^i$	Abbildung des i. Passpunktes im Bild
P'	Punkt im Bild in homogenen Koordinaten
P'_w	Weltpunkt in homogenen Koordinaten
P'_k	Kamerapunkt in homogenen Koordinaten
M	Projektionsmatrix von Kamera- in Bildkoordinaten
T	Translationsmatrix zur Verschiebung des Weltkoordinatensystems
R	Rotationsmatrix mit den Winkeln ω, ϕ und κ zur Rotation um die x-, y- bzw. z-Achse (Primär-, Sekundär-, Tertiärdrehung)
M'	M in homogenen Koordinaten
T'	T in homogenen Koordinaten
R'	R in homogenen Koordinaten

2.2 Kameramodell

Zur Beschreibung der Kamera und des darauf aufbauenden Abbildungsvorganges wird das Lochkameramodell (*Camera obscura*) verwendet. Dieses stellt eine Abstraktion genau der Klasse realer Kameras dar, welche ein punktförmiges Projektionszentrum besitzen und das Bild in eine Ebene abbilden. Hierzu gehören alle handelsüblichen analogen sowie digitalen Kameras. Ausgeschlossen sind z.B. 360°-Kameras oder Panoramakameras, bei denen die Bildebene einen Zylinder darstellt.

Eine Lochkamera besteht physisch lediglich aus einem Gehäuse, in welches durch eine lochförmige Öffnung Licht einfällt. Auf der gegenüber liegenden Seite der Öffnung – der Projektionsfläche – entsteht dadurch ein Abbild der angestrahlten Objekte. Abbildung 2.1 zeigt das grundlegende Modell des Projektionsvorgangs. Dabei wird angenommen, dass die Richtungen der x- und y-Achsen des Weltkoordinatensystems mit denen des Bildkoordinatensystems übereinstimmen und es sich um kartesische Koordinatensysteme handelt, die Z_w-Achse also mit der optischen Achse zusammenfällt. Das *optische Zentrum* entspricht der Öffnung in der Lochkamera, c ist die so genannte *Kamerakonstante*. Durch die Abbildung von $P_w(X_w, Y_w, Z_w)$ entsteht in der Projektionsebene (auch *Retina-Ebene* genannt) der Punkt $P(x, y)$.

Bei realen Kameras findet ein mit dem Lochkameramodell vergleichbarer Abbildungsvorgang statt, nur dass Objekte durch eine Linse bzw. ein Linsensystem auf die Bildebene projiziert werden. Hier sei angenommen, dass eine ideale dünne Linse zur Verfügung steht, wodurch sich

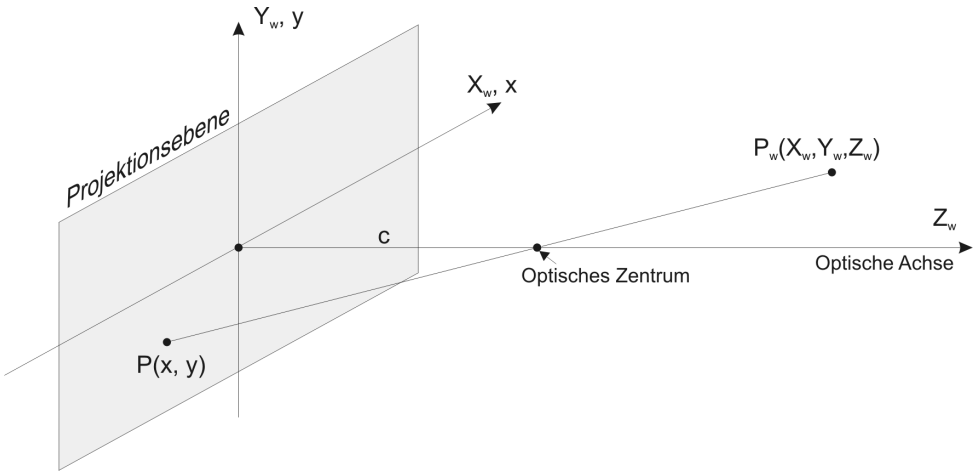

Abbildung 2.1: Grundlegendes Modell des Abbildungsvorgangs (nach [18])

das Lochkameramodell verwenden lässt. Der Linsenmittelpunkt stimmt mit dem optischen Zentrum überein und die Brennweite f der Linse gibt den Abstand vom Linsenmittelpunkt zum Brennpunkt an, wie Abbildung 2.2 verdeutlicht.

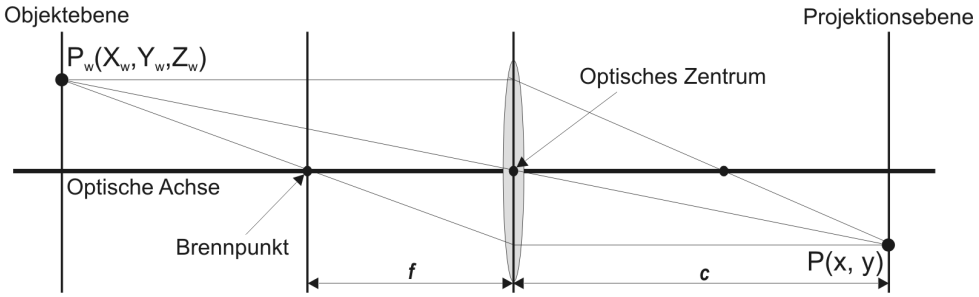

Abbildung 2.2: Abbildungsvorgang bei einer dünnen Linse

Nun handelt es sich bei dem Lochkameramodell um eine idealisierte Darstellung der realen Abbildungsvorgänge. Daher wird das Modell oft auch nicht 1:1 übernommen, sondern teilweise angepasst. Diese Anpassung spiegelt sich in einer Erweiterung der Anzahl an Parametern der inneren Orientierung bzw. deren Kalibrierung wider. Sie bezieht sich beispielsweise auf den CCD-Chip von Digitalkameras, dessen Zellen nicht immer quadratisch sind. Dies lässt sich jedoch durch Aufspaltung der Kamerakonstante c in zwei Teile c_x (für die horizontale Richtung) sowie c_y (für die vertikale Richtung) bereinigen. Weiterhin stimmt durch ungenaue Platzierung der Mittelpunkt des CCD-Chips i. Allg. nicht exakt mit dem Bildhauptpunkt (also dem Durchstoßpunkt der optischen Achse in der Bildebene) überein. Ferner sind die Koordinatenachsen der Welt und des Bildes jeweils unterschiedlich skaliert (metrisches System kontra Pixelkoordinaten). Die Berücksichtigung dessen stellt durch entsprechende Kalibrierung der inneren Orientierung ebenfalls kein Problem dar.

Eine weitere Eigenschaft realer Linsen bzw. Linsensysteme sind mögliche Verzerrungen, welche die Projektion beeinflussen. Häufig auch als *Linsenfehler* bezeichnet, wirken sich diese auf den

Abbildungsvorgang derart aus, dass Punkte im Bild an anderer Stelle erscheinen als durch das Lochkameramodell angenommen. Einige Kalibrierungsverfahren beziehen diese Linsenverzerrungen (und hier insbesondere *radiale Linsenverzerrungen*) mit in die Berechnung ein, andere nicht (siehe Abschnitt 2.4).

2.3 Abbildungsvorgang

Aufbauend auf dem in 2.2 spezifizierten Kameramodell lässt sich nun der Abbildungsvorgang eines realen Objekts in ein Bild beschreiben.

Zunächst soll der *interne Abbildungsvorgang* dargestellt werden. Dieser wurde bereits in Abbildung 2.1 illustriert und beschreibt die Projektion eines Punktes P_w in die Bildebene. Dabei wird stillschweigend angenommen, dass das Kamerakoordinatensystem mit dem Weltkoordinatensystem übereinstimmt. Diese Einschränkung soll in 2.3.1 zur Veranschaulichung des Abbildungsvorganges gelten, wird in 2.3.2 aber aufgehoben.

2.3.1 Interne Projektion

Aus der angesprochenen Einschränkung der Übereinstimmung von Welt- und Kamerakoordinatensystem resultiert, dass ein Punkt P_w nicht erst in das Kamerakoordinatensystem überführt werden muss. Somit gilt $P_w(X_w, Y_w, Z_w) = P_k(X_k, Y_k, Z_k)$. Aus Abbildung 2.1 folgt nun über die Strahlensätze direkt eine Zentralprojektion des Punktes $P_w(X_w, Y_w, Z_w)$ (der P_k entspricht) in den Punkt $P(x, y)$ mit folgenden Zusammenhängen:

$$\begin{aligned} \frac{x}{c} = -\frac{X_k}{Z_k} \quad &\equiv \quad x = -\frac{c \cdot X_k}{Z_k} \\ \frac{y}{c} = -\frac{Y_k}{Z_k} \quad &\equiv \quad y = -\frac{c \cdot Y_k}{Z_k} \end{aligned} \tag{2.1}$$

Die negativen Vorzeichen auf der rechten Seite der Gleichungen zeigen, dass die Koordinaten des projizierten Bildes zu denen des Welt- bzw. Kamera-Punktes invertiert sind, das Bild also „auf-dem-Kopf-stehend“ dargestellt wird.

Löst man die Gleichungen von 2.1 nach X_k bzw. Y_k auf, erhält man:

$$\begin{aligned} X_k &= -\frac{x \cdot Z_k}{c} \\ Y_k &= -\frac{y \cdot Z_k}{c} \end{aligned} \tag{2.2}$$

Diese Gleichungen lassen sich ohne Kenntnis von Z_k nicht lösen. Das Problem dabei ist die Transformation von 3D- auf 2D-Koordinaten, welche durch die Gleichungen 2.1 gegeben ist. Der

so berechnete Bildpunkt $P(x, y)$ entspricht genau der Menge von 3D-Weltpunkten, welche durch $P_w(x, y, 0)$ sowie den Punkt $(0, 0, c)$ auf der optischen Achse führen. Die obigen Gleichungen 2.2 definieren diese Gerade im Kamerakoordinatensystem (siehe auch Abbildung 2.1).

Die Zentralprojektions-Gleichungen 2.1 gelten zunächst nur für das Lochkameramodell. In der Realität möchte man den Bildursprung jedoch meist in der linken oberen Bildecke haben. Dies entspricht einer Translation um die Koordinaten des Bildhauptpunktes (x_0, y_0), womit sich die Gleichungen 2.1 erweitern zu:

$$\begin{aligned} x &= x_0 - \frac{c \cdot X_k}{Z_k} \\ y &= y_0 - \frac{c \cdot Y_k}{Z_k} \end{aligned} \tag{2.3}$$

Daraus ergibt sich direkt die zugehörige homogene Projektionsmatrix M':

$$M' = \begin{pmatrix} 1 & 0 & -\frac{x_0}{c} & 0 \\ 0 & 1 & -\frac{y_0}{c} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-1}{c} & 0 \end{pmatrix} \tag{2.4}$$

Stellt man den Punkt P_k in homogenen Koordinaten als P'_k dar, so ergibt sich für die Projektion folgende Gleichung:

$$P' = M' * P'_k = \begin{pmatrix} 1 & 0 & -\frac{x_0}{c} & 0 \\ 0 & 1 & -\frac{y_0}{c} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-1}{c} & 0 \end{pmatrix} * \begin{pmatrix} X_k \\ Y_k \\ Z_k \\ 1 \end{pmatrix} = \begin{pmatrix} X_k - \frac{x_0 \cdot Z_k}{c} \\ Y_k - \frac{y_0 \cdot Z_k}{c} \\ 0 \\ \frac{-Z_k}{c} \end{pmatrix} \tag{2.5}$$

Die kartesischen Koordinaten von P' lassen sich leicht durch Division des vierten Elements erhalten, womit man zu den Gleichungen 2.3 gelangt.

2.3.2 Externe Transformation

Da nun der interne Abbildungsvorgang bekannt ist, kann eine Erweiterung dessen erfolgen, indem die Einschränkung aufgehoben wird, dass das Kamerakoordinatensystem mit dem Weltkoordinatensystem übereinstimmt. Abbildung 2.3 zeigt das so modifizierte Kameramodell.

Wie leicht ersichtlich ist, geht das Weltkoordinatensystem durch Translation und Rotation in das Kamerakoordinatensystem über. Der Ursprung 0_k des Kamerakoordinatensystems stimmt dabei mit dem Hauptpunkt der projektiven Abbildung überein. Die Translation findet im Raum durch den Vektor t statt und besitzt 3 Freiheitsgrade. Diese sind definiert durch den Ursprung

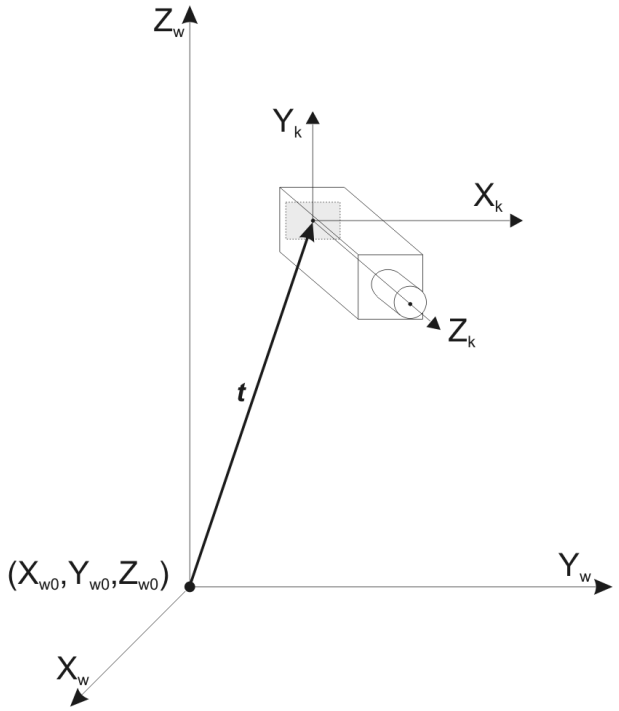

Abbildung 2.3: Beziehung zwischen Welt- und Kamerakoordinatensystem

$0_w(X_{w0}, Y_{w0}, Z_{w0})$ des Weltkoordinatensystems, der bezüglich des Punktes 0_k angegeben ist. Die Translationsmatrix T (bzw. T' für homogene Koordinaten) definiert sich so wie folgt:

$$T = \begin{pmatrix} 1 & 0 & 0 & -X_{w0} \\ 0 & 1 & 0 & -Y_{w0} \\ 0 & 0 & 1 & -Z_{w0} \end{pmatrix}, \qquad T' = \begin{pmatrix} 1 & 0 & 0 & -X_{w0} \\ 0 & 1 & 0 & -Y_{w0} \\ 0 & 0 & 1 & -Z_{w0} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.6}$$

Auch der Rotation lassen sich 3 Freiheitsgrade zuordnen: ω, ϕ und κ geben den jeweiligen Rotationswinkel um die X-, Y- bzw. Z-Achse an. Es sei dabei ein *Rechtssystem* angenommen, also Rotationen gegen den Uhrzeigersinn, wenn man von der Spitze einer Koordinatenachse Richtung Ursprung schaut. Die so definierten Rotationsmatrizen R_ω, R_ϕ sowie R_κ und die Matrix der Gesamtrotation R als Verkettung der Einzelrotationen ergeben sich dann in kartesischen Koordinaten aus:

$$R_\omega = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{pmatrix}, \qquad R_\phi = \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix}, \qquad R_\kappa = \begin{pmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R = R_\omega R_\phi R_\kappa = \begin{pmatrix} \cos\phi\cos\kappa & -\cos\phi\sin\kappa & \sin\phi \\ \cos\omega\sin\kappa + \sin\omega\sin\phi\cos\kappa & \cos\omega\cos\kappa - \sin\omega\sin\phi\sin\kappa & -\sin\omega\cos\phi \\ \sin\omega\sin\kappa - \cos\omega\sin\phi\cos\kappa & \sin\omega\cos\kappa + \cos\omega\sin\phi\sin\kappa & \cos\omega\cos\phi \end{pmatrix} \tag{2.7}$$

Zur vereinfachten Schreibweise sollen die Elemente der Gesamtrotations-Matrix substituiert werden. Dadurch ergibt sich für die Matrix R (bzw. R' für homogene Koordinaten):

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}, \qquad R' = \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.8}$$

In homogenen Kamerakoordinaten ergibt sich ein Weltpunkt $P'_w(X_w, Y_w, Z_w, 1)$ dann als:

$$P'_k = R'^T T' P'_w = \begin{pmatrix} X_k \\ Y_k \\ Z_k \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11}(X_w - X_{w0}) + r_{21}(Y_w - Y_{w0}) + r_{31}(Z_w - Z_{w0}) \\ r_{12}(X_w - X_{w0}) + r_{22}(Y_w - Y_{w0}) + r_{32}(Z_w - Z_{w0}) \\ r_{13}(X_w - X_{w0}) + r_{23}(Y_w - Y_{w0}) + r_{33}(Z_w - Z_{w0}) \\ 1 \end{pmatrix} \tag{2.9}$$

2.3.3 Gesamter Abbildungsvorgang

Der vollständige Abbildungsvorgang setzt sich zusammen aus der Translation und Rotation des Weltkoordinatensystems nach Gleichung 2.9, welches damit in das Kamerakoordinatensystem übergeht, sowie der internen Projektion mittels Gleichung 2.4 in das Bildkoordinatensystem. Fasst man beide Gleichungen zusammen, ergibt sich die Gesamtgleichung für die Projektion eines homogenen Weltpunktes P'_w in einen Bildpunkt P' aus:

$$P' = M' R'^T T' P'_w \tag{2.10}$$

Dies führt zu den folgenden Projektionsgleichungen:

$$\begin{aligned} x &= x_0 - c\frac{r_{11}(X_w - X_{w0}) + r_{21}(Y_w - Y_{w0}) + r_{31}(Z_w - Z_{w0})}{r_{13}(X_w - X_{w0}) + r_{23}(Y_w - Y_{w0}) + r_{33}(Z_w - Z_{w0})} \\ y &= y_0 - c\frac{r_{12}(X_w - X_{w0}) + r_{22}(Y_w - Y_{w0}) + r_{32}(Z_w - Z_{w0})}{r_{13}(X_w - X_{w0}) + r_{23}(Y_w - Y_{w0}) + r_{33}(Z_w - Z_{w0})} \end{aligned} \tag{2.11}$$

2.4 Kamerakalibrierung

Unter der Kamerakalibrierung versteht man auf Basis des vorgestellten Kameramodells die Bestimmung der Parameter der inneren sowie (in den meisten Fällen) der äußeren Orientierung der Kamera, welche die Abbildung von Objekt- in Bildkoordinaten beschreiben und somit für photogrammetrische Messungen benötigt werden. Für diese Zwecke entstanden über die Jahre eine Vielzahl von Algorithmen, die sich bezüglich der Art der Parameterbestimmung, von ihnen benötigter Informationen, Einbeziehung realer physikalischer Parameter, Laufzeitkomplexität und Genauigkeit unterscheiden.

Per se werden nach [19] drei Kalibrierungsarten unterschieden:

1. **Laborkalibrierung**: Hierbei werden die Parameter der inneren Orientierung bei meist bekannter äußerer Orientierung durch den Hersteller in dessen Laboratorien physikalisch gemessen. Dies hat eine hohe Genauigkeit zur Folge, ist aber aufgrund des Aufwandes und damit entstehender Kosten für Endanwender meist nicht von Interesse.

2. **Testfeldkalibrierung**: Mit dem zu kalibrierenden System werden aus verschiedenen Positionen Punkte aufgenommen, die im Objektraum verteilt sind (*Testfeld*) und deren Lage bekannt ist. Mit diesen können die Parameter der inneren sowie äußeren Orientierung bestimmt werden.

3. **Simultankalibrierung**: Die Kamera wird anhand einer Menge von bekannten Passpunkten direkt während der eigentlichen Messung kalibriert, d.h. dass Mess- und Kalibrierungsaufnahmen identisch sind. Damit wird bei einer Kamera, die statisch bezüglich des verwendeten Weltkoordinatensystems ist, garantiert, dass sie stets kalibriert ist. Dies wird für den betrachteten Fall angenommen, womit diese Art der Kalibrierung hier die günstigste darstellt und nachfolgend näher betrachtet werden soll.

2.4.1 Direkte Lineare Transformation (DLT)

Die Direkte Lineare Transformation (DLT) wurde 1971 von Y.I. Abdel-Aziz und H.M. Karara vorgeschlagen und stellt ein Simultankalibrierungs-Verfahren dar, welches die Besonderheit aufweist, dass zur Bestimmung der Parameter der inneren und äußeren Orientierung ausschließlich lineare Gleichungen verwendet werden.

Für die Linearisierung werden im ersten Schritt die Gleichungen aus 2.11 so umgeformt, dass die Hauptpunktkoordinaten x_0 und y_0 mit in die Brüche einbezogen werden. Anschließend werden die Unbekannten beider Gleichungen durch Variablen L_i substituiert und es wird so transformiert,

dass folgende Form ensteht:

$$\begin{aligned} x &= \frac{L_1 X_w + L_2 Y_w + L_3 Z_w + L_4}{L_9 X_w + L_{10} Y_w + L_{11} Z_w + 1} \\ y &= \frac{L_5 X_w + L_6 Y_w + L_7 Z_w + L_8}{L_9 X_w + L_{10} Y_w + L_{11} Z_w + 1} \end{aligned} \tag{2.12}$$

Es ist ersichtlich, dass die 9 Unbekannten der Gleichungen 2.11 durch 11 Unbekannte in 2.12 ersetzt wurden. Diese Ersetzung ist so jedoch nicht zulässig, da bei einer Substitution die Anzahl der Parameter erhalten bleiben muss. Die Lösung dieses Problems liegt in der Einführung von 2 zusätzlichen Parametern in den Ursprungsgleichungen. Hier soll den Ausführungen von [12] gefolgt werden, wo für die beiden Parameter einerseits ein *unterschiedlicher Maßstab m in einer der beiden Achsen* und andererseits eine *Scherung s* der x- und y-Achse angegeben wird. Ist die Lösung der DLT mathematisch exakt, betragen $m = 1$ und $s = 0$, womit sich ein kartesisches Koordinatensystem ergibt. Dies bedeutet allerdings auch, dass das für die DLT verwendete Koordinatensystem nicht mehr zwangsläufig orthogonal sein muss. In der Praxis weichen m und s allerdings so geringfügig von ihren Idealwerten ab, dass $m = 1$ und $s = 0$ angenommen werden können.

Nach Umstellung ergeben sich die Gleichungen 2.12 zu:

$$\begin{aligned} x &= L_1 X_w + L_2 Y_w + L_3 Z_w + L_4 - x L_9 X_w - x L_{10} Y_w - x L_{11} Z_w \\ y &= L_5 X_w + L_6 Y_w + L_7 Z_w + L_8 - x L_9 X_w - x L_{10} Y_w - x L_{11} Z_w \end{aligned} \tag{2.13}$$

Die 11 unbekannten Parameter lassen sich mit einem linearen Gleichungssystem bestehend aus 11 Gleichungen ermitteln. Demzufolge benötigt man mindestens $5\frac{1}{2}$ Passpunkte (1 Passpunkt jeweils für die Berechnung von x und y). Da in der Praxis nur ganze Punkte gemessen werden, nimmt man in der Regel 6 oder mehr Passpunkte, womit man ein überbestimmtes Gleichungssystem erhält, welches sich mit gängigen Methoden der Ausgleichsrechnung (Methode kleinster Quadrate etc.) lösen lässt.

Bei der Wahl der Passpunkte sollte berücksichtigt werden, dass die Punkte ein Volumen aufspannen, d.h. nicht bzw. auch nicht annähernd in ein und derselben Ebene liegen. Eine Vergrößerung dieses Volumens bewirkt eine Erhöhung der Genauigkeit.

Um von den DLT-Gleichungen die innere und äußere Orientierung bestimmen zu können, muss eine Rücksubstitution der Parameter $L_1, \cdots, L_{11}$ vorgenommen werden. [16, 25] geben diese an, hier wird solch eine Rücksubstitution hingegen nicht benötigt, weswegen nicht näher darauf eingegangen werden soll.

2.4.2 Weitere Methoden der Simultankalibrierung

Der große Vorteil der DLT liegt wie schon erwähnt im ausschließlichen Lösen linearer Gleichungssysteme. Bezüglich des idealisierten Modells der Lochkamera ist dieses Vorgehen exakt und somit problemlos anwendbar. Jedoch hat die DLT auch einige Nachteile. So werden mindestens 6 Passpunkte benötigt, die im Raum verteilt sein müssen und nicht in einer Ebene liegen dürfen. Das mithin größte Problem ist allerdings, dass es bei realen Kameras wie schon erwähnt zu Linsenfehlern kommen kann, die z.B. bei einer radialen Linsenverzerrung zu einem quadratischen Summanden in den Formeln der Zentralprojektion führen. Diese können durch die DLT nicht mit betrachtet werden, sodass es eine Reihe weiterer Kalibrierungstechniken gibt, deren wichtigste Vertreter kurz dargestellt werden sollen.

Verfahren von Tsai Dieses 1987 von Roger Tsai in [24] vorgestellte Verfahren kommt bereits mit der Aufnahme einer Ebene aus, was die Einschränkung des DLT-Verfahrens aufhebt, dass die Passpunkte ein Volumen aufspannen müssen. Das Verfahren findet einen Teil der gesuchten Parameter zunächst durch Lösung zweier linearer Gleichungssysteme, um die Berechnung schließlich durch eine nichtlineare Optimierung abzuschließen. Die Kalibrierung bezieht radiale Linsenverzerrungen mit ein und benötigt mindestens 5 Passpunkte. Eine große Einschränkung stellen einige interne Kameraparameter dar, die bekannt sein müssen, damit das Verfahren angewendet werden kann. Da dies bei üblichen Consumer-Kameras nicht der Fall ist, scheidet dieses Verfahren häufig aus.

Verfahren von Zhang 1998 schlug Z. Zhang in [28] diese Methode zur Kamerakalibrierung vor, welche sich im Gegensatz zum Tsai-Verfahren durch erhöhte Flexibilität und Stabilität auszeichnet. Als Kalibriermuster wird ebenfalls eine Ebene verwendet, die allerdings in mehreren frei wählbaren Orientierungen aufgenommen werden muss. Pro aufgenommener Ebene benötigt man mindestens 4 Passpunkte. Das Verfahren an sich beschreibt nun die Beziehung zwischen Kalibrierebene und dem aufgenommenen Bild durch eine ebene Homographie mit 8 Freiheitsgraden, welche mit einer reduzierten Version des DLT-Algorithmus bestimmt werden kann. Wie bei dem Verfahren nach Tsai wird schlussendlich eine nichtlineare Optimierung durchgeführt, um die Genauigkeit der berechneten Parameter zu erhöhen. Auch mit dem Zhang-Verfahren können Linsenverzerrungen modelliert werden.

Bündelausgleich Dies stellt eine nichtlineare Optimierung dar, wobei die nichtlinearen Teile der Gleichungen 2.11 (implizit durch die Sinus- und Kosinus-Terme in den Elementen der Rotationsmatrix gegeben) durch ein lineares Taylor-Polynom approximiert werden können. Der Bündelausgleich ist ein iteratives Verfahren, für dessen Startwert es sinnvoll ist die DLT-Lösung zu verwenden, da diese im Konvergenzbereich des gesuchten Minimums liegt. Wird mit dieser das linearisierte Gleichungssystem gelöst, erhält man Verbesserungswerte der Startlösung. Durch mehrfache Anwendung dieses Vorgehens lässt sich der Fehler weiter

minimieren. Vorteil des Bündelausgleichs ist seine Exaktheit, wie bei den zuvor vorgestellten Verfahren lassen sich auch hier radiale Linsenverzerrungen mit einbeziehen. Nachteilig ist vorrangig die hohe Laufzeit zu nennen, die aus dem iterativen Vorgehen resultiert.

Neben dieser kleinen Auswahl an Verfahren existieren noch eine Vielzahl weiterer, deren Vorstellung den Rahmen dieser Arbeit sprengen würde. Dass hier die DLT als Verfahren der Wahl zum Einsatz kommt, hat verschiedene Gründe. Es ist das schnellste und intuitivste Verfahren, welches bei handelsüblichen Kameras nur geringfügige Abweichungen bezüglich der Parameterberechnungen aufweist. Außer 6 Passpunkten werden keine weiteren Parameter benötigt, was das Verfahren in der Praxis einfach anwendbar macht.

2.5 Messen in der Ebene

Wie bereits in Abschnitt 2.3.1 dargestellt wurde, ist die Umrechnung von Welt- in Bildkoordinaten unter Verwendung des entsprechenden Kameramodells problemlos möglich, in umgekehrter Richtung ist das aber nicht der Fall, da sich hier für jeden Bildpunkt eine Gerade im Weltkoordinatensystem ergibt, auf welcher der zugehörige Objektpunkt liegen kann. Somit benötigt man für die Rückrechnung von Objekt- in Weltkoordinaten noch die Information, wo sich der Punkt auf der Geraden befindet. Häufig will man ein Objekt in einer bekannten Ebene messen, wofür man den Schnittpunkt der Geraden mit der Ebene berechnet. Daraus ergibt sich sofort der gesuchte Weltpunkt.

O.B.d.A. kann man für diese Schnittebene die Ebene $Z_w = 0$ verwenden, die sich durch Rotation in jede andere Ebene überführen lässt. In diesem Fall lässt sich Gleichung 2.11, welche die Gesamt-Projektion angibt, nach X_w und Y_w auflösen:

$$\begin{aligned} X_w &= X_{w0} - Z_{w0}\frac{r_{11}(x-x_0) + r_{12}(y-y_0) - r_{13}c}{r_{31}(x-x_0) + r_{32}(y-y_0) - r_{33}c} \\ Y_w &= Y_{w0} - Z_{w0}\frac{r_{21}(x-x_0) + r_{22}(y-y_0) - r_{23}c}{r_{31}(x-x_0) + r_{32}(y-y_0) - r_{33}c} \end{aligned} \tag{2.14}$$

Diese Gleichungen erfordern das explizite Wissen über die Parameter der inneren und äußeren Orientierung. Meist hat man bei der Kalibrierung mittels DLT aber nur den mathematischen Zusammenhang zwischen Bild- und Weltkoordinaten in Form der Parameter $L_1 \cdots L_{11}$ errechnet, jedoch keine Rücksubstitution vorgenommen. Für diesen Fall lässt sich Gleichung 2.12 über Gleichung 2.13 umformen zu:

$$\begin{aligned} (L_1 - xL_9)X_w + (L_2 - xL_{10})Y_w + L_4 - x = 0 \\ (L_5 - yL_9)X_w + (L_6 - yL_{10})Y_w + L_8 - y = 0 \end{aligned} \tag{2.15}$$

Aus diesem linearen Gleichungssystem lassen sich leicht X_w und Y_w berechnen, die den gesuchten Punkt in der X_wY_w-Ebene angeben.

2.6 Praktische Erwägungen

Um eine verwendbare Begrenzung des Bildbereichs im Sinne von Kapitel 3 zu erhalten und eine möglichst genaue Einzeichnung des Lichtraumprofils und Entfernungsmessung durchführen zu können (was auch entscheidend für den Erfolg des eingesetzten Algorithmus ist), sollte eine Kalibrierung mittels DLT den folgenden Maßgaben genügen:

1. Der Standpunkt der Kamera im Führerhaus des Zuges muss fest und unbeweglich sein, sodass die Kalibrierung initial erfolgen kann und dann für alle Videos verwendbar ist, die darauf aufbauen.
2. Die Kalibrierung der Kamera hat auf einem exakt geraden Schienenstück zu erfolgen, d.h. nicht in einer Kurve.
3. Die Kamera muss so aufgestellt werden, dass die x-Achse des Bildkoordinatensystems annähernd parallel zur X_w-Achse des Weltkoordinatensystems liegt. Wenn sich die Kamera beispielsweise in der Mitte zwischen den beiden Schienen befindet, so sollte die mittlere x-Koordinate des Bildes mit der Koordinate des Fluchtpunktes der beiden Schienen am Horizont bei Geradeausfahrt übereinstimmen.
4. Der Ursprung des Weltkoordinatensystems muss direkt an der Innenkante der linken Schiene liegen, die Entfernung $Y_w = 0$ ist auf Höhe der Kamera im Führerhaus zu legen.
5. Die linke Schiene hat bei Geradeausfahrt mit der Y_w-Achse des Weltkoordinatensystems übereinzustimmen. Die Passpunkte sind so zu messen, dass alle Koordinaten auf der linken Schiene den Wert $X_w = 0$ besitzen.
6. Für die DLT sind mindestens 6 Passpunkte notwendig. Die ersten beiden sollten sich nahe am unteren Bildrand an der Innenseite von linker und rechter Schiene befinden. Für die nächsten beiden ist es sinnvoll, sie auf den Schienen in der Entfernung des einzuzeichnenden Lichtraumprofils zu messen. Die letzten beiden Punkte sind notwendig, um ein Volumen aufzuspannen und sollten die gleichen Koordinaten wie die Punkte 3 und 4 besitzen, nur mit einer erhöhten Z_w-Koordinate, die ca. der Höhe des Lichtraumprofils entsprechen sollte.
7. Allgemein gilt: Je höher die Kamera angebracht werden kann, desto besser (dadurch lassen sich Schienen auch in größerer Entfernung noch gut erkennen).

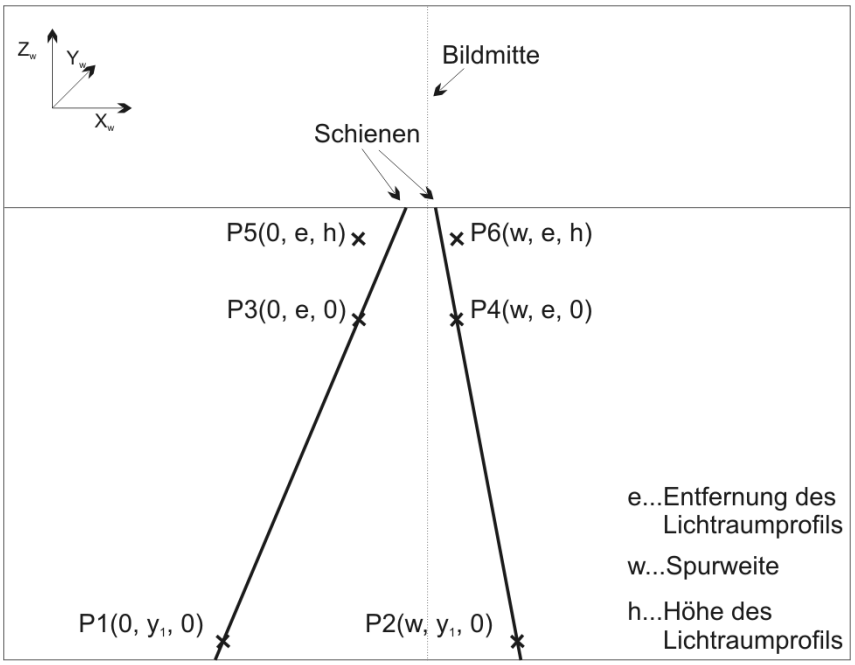

Abbildung 2.4: Praktische Maßgaben bei der Kalibrierung

Abbildung 2.4 verdeutlicht noch einmal einige der vorgestellten Anforderungen. Insgesamt lässt sich sagen, dass diese Maßgaben zwar einiges an Aufmerksamkeit erfordern, jedoch keine generelle Einschränkung darstellen. So ließe sich der Kalibrierungsprozess mit einem speziell entwickelten Tool sehr gut unterstützen und teilautomatisieren. Dieses könnte beispielsweise die Entfernung des einzuzeichnenden Lichtraumprofils und die Spurweite der Schiene vom Benutzer erfragen, anhand dessen die beiden hinteren unteren Weltpunkte generieren und den Benutzer dazu veranlassen, deren Entsprechungen im Bild zu messen. Ebenso könnten die Schienenschnittpunkte mit dem untersten Bildrand in Welt- und Bildkoordinaten abgefragt werden, wodurch sich ebenfalls einige ansonsten manuell einzugebende Parameter ersparen ließen.

3 Begrenzung des Suchbereichs

In diesem Kapitel soll untersucht werden, wie sich A-priori-Wissen zunutze machen lässt, um Beschränkungen bezüglich des Suchraums verschiedener Parameter zu definieren, was einen grundlegenden Bestandteil des Gesamtsystems darstellt. Zum einen hat es die *passive* Wirkung der Laufzeitverkürzung, da Algorithmen nur auf einem kleinen Teilbereich einer Problemgröße operieren müssen. Zum anderen ist der *aktive* Einsatz bedeutsam, durch den der Algorithmus zur Schienenerkennung unmittelbar unterstützt und verbessert werden kann.

3.1 Bildraumbegrenzung

Aufgrund der Tatsache, dass zum einen der Fahrweg durch das Schienensystem bei Zügen nicht verlassen werden kann und zum anderen Kurvenfahrten durch minimale Kurvenradien beschränkt sind, lässt sich der zu untersuchende Bildbereich initial (d.h. einmal zu Beginn der Auswertung) auf einen kleinen Teilbereich beschränken.

Zunächst ist die minimale y-Ordinate durch den Punkt gegeben, an dem das Lichtraumprofil bei Geradeausfahrt aufsetzen soll. Bei Kurvenfahrten ergeben sich natürlicherweise größere y-Werte. Für jede y-Ordinate des so beschränkten Bereichs lässt sich nun jeweils eine minimale und eine maximale x-Koordinate ermitteln. Betrachtet man den minimalen x-Wert, so ergibt er sich aus dem Wert, welcher durch das linke Gleis bei maximaler Linkskurvenfahrt gegeben ist. Diesen gilt es zu berechnen.

Zunächst muss bedacht werden, dass die aktuelle Fahrtrichtung des Zuges *nicht* tangential zur Schiene im Bereich des Führerhauses ist, in welchem sich auch die Kamera befindet. Vielmehr wird sie durch die Tangente zur Schiene an dem Punkt P_{Mitte} beschrieben, der die Mitte zwischen den beiden Radachsen des Lokwagens darstellt. Abbildung 3.1 verdeutlicht diesen Sachverhalt. Der Abstand $\overline{Y_{Kamera}Y_{Mitte}}$ stellt also einen Parameter des Systems dar und muss entsprechend berücksichtigt werden. Zur Vereinfachung kann dieser auch durch die Hälfte der Länge des Lokwagens approximiert werden, wodurch sich i. Allg. nur ein geringer Fehler ergibt.

Für eine gegebene Ordinate y' im Bild lässt sich die entsprechende x_{min}-Koordinate nun wie folgt berechnen: es wird der minimale Kurvenradius r_{min} zugrunde gelegt und im Weltkoordinatensystem ein Kreisbogen aufgespannt, wie in Abbildung 3.2 zu sehen ist. Ein typischer Wert für r_{min} ist 180m, der nach [3] für Neubauten von Nebengleisen gelten soll. Der Längsschnitt

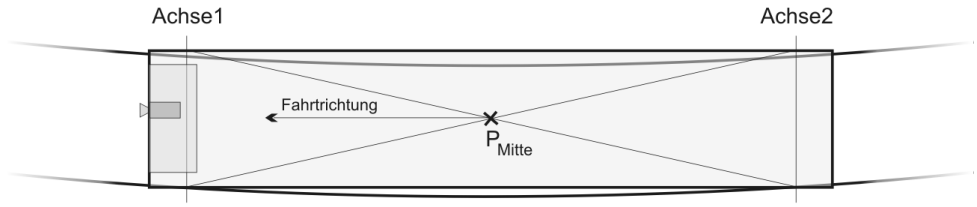

Abbildung 3.1: Fahrtrichtung tangential im Punkt P_{Mitte}

durch den Lokwagen ist tangential zu dem Kreisbogen im Punkt P_{Mitte}. Wie in Abschnitt 2.6 besprochen wurde, sei der Punkt auf der linken Schiene in Höhe der Kamera der Ursprung des Weltkoordinatensystems. Das gesuchte y' wird nun zunächst in Weltkoordinaten zu Y'_w umgerechnet und es wird die X-Koordinate des Schnittpunktes S' mit dem Kreisbogen ermittelt, die X_{min} entspricht. Dies geschieht durch

$$X_{min} = \sqrt{r_{min}^2 - Y_{Gegenkathete}^2} - r_{min} \qquad \text{wobei} \quad Y_{Gegenkathete} = Y'_w + \overline{Y_{Kamera}Y_{Mitte}}$$

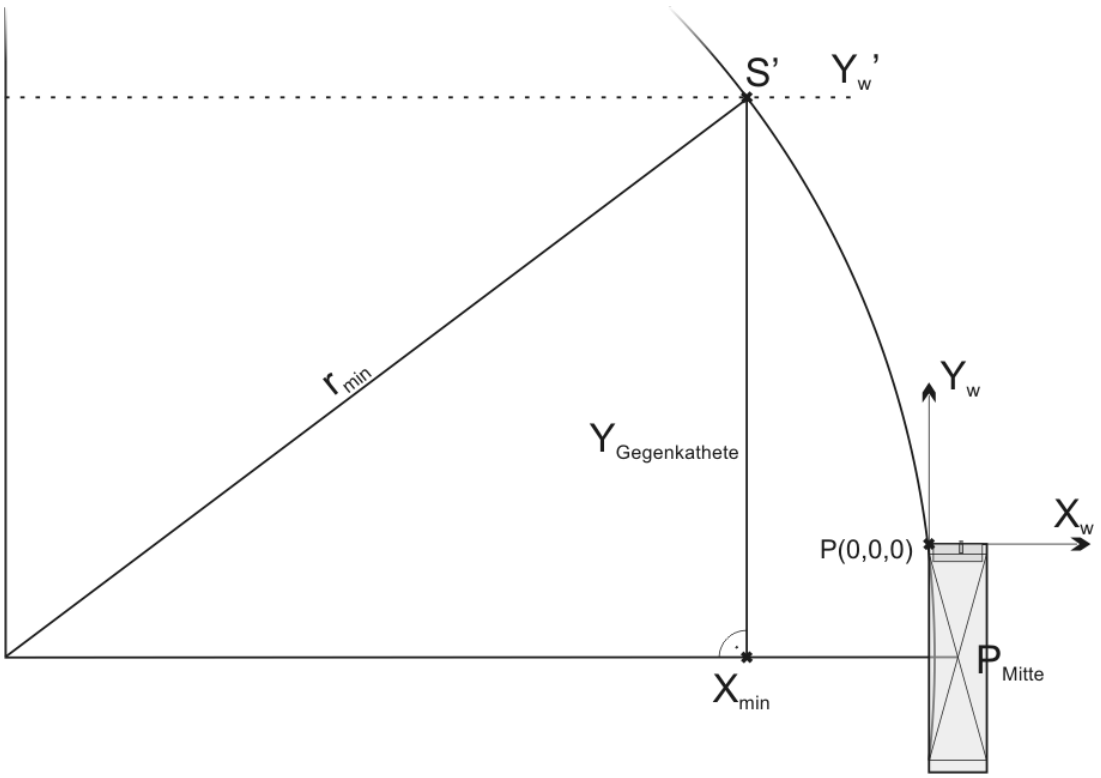

Abbildung 3.2: Berechnung der minimalen Koordinate X_{min} durch $S'(X_{min}, Y'_w)$

Durch erneute Transformation dieses Punktes in Bildkoordinaten erhält man schließlich den gesuchten x_{min}-Wert. Es sollte ersichtlich sein, dass $Y'_w \leq r_{min} - \overline{Y_{Kamera}Y_{Mitte}}$ gelten muss, um einen Schnittpunkt mit dem Kreisbogen zu erhalten. Ist dies nicht der Fall oder liegt x_{min} außerhalb des Bildbereichs, so kann für die entsprechende Ordinate y' keine Begrenzung durchgeführt werden.

Analog dazu verfährt man bei der Ermittlung des maximalen Wertes x_{max} durch Schnittpunktberechnung des rechten Gleises bei maximaler Rechtskurvenfahrt. Dies ergibt eine initiale Bildraumbeschränkung, wie sie in Abbildung 3.3 beispielhaft dargestellt ist. Schön ersichtlich ist dabei auch die Transformation des Kreises durch die perspektivische Abbildung in eine annähernd parabolische Form.

Abbildung 3.3: Beispiel einer Bildraumbeschränkung

3.2 Winkelbegrenzung

Neben der Bildraumbegrenzung stellt die Einschränkung zulässiger Winkel für Schienenstücke einen zweiten Parameter dar, der zur Effizienzsteigerung des Systems eingesetzt werden kann. Zum einen lässt sich damit der zu untersuchende Parameterraum der Hough-Transformation (siehe auch Abschnitt 4.4) effektiv begrenzen und somit die Anzahl aufwendiger Sinus- und Kosinus-Berechnungen reduzieren. Zum anderen lässt sich eine *Winkelfilterung* im Anschluss an eine Kantendetektion durchführen, wenn diese neben der Stärke des Gradienten auch dessen Richtung ausgibt (siehe Abschnitt 4.3.5.4). Dadurch können viele Pixel eliminiert werden, die zwar als Kantenpunkt erkannt wurden, jedoch zu uninteressanten Strukturen gehören.

Wie auch der Bildraum kann der zulässige Winkelbereich durch maximale Kurvenfahrten be-

schränkt werden. Für ein gegebenes y_i im Bild sind nur solche Winkel zulässig, die sich im Bereich zwischen maximaler Linkskurvenfahrt und maximaler Rechtskurvenfahrt befinden. Dabei muss durch die perspektivische Abbildung beachtet werden, dass die maximalen Winkel nicht immer durch die kurvenäußeren Schienen bestimmt werden, wie man zunächst annehmen könnte. Abbildung 3.4 verdeutlicht diese Überlegung: eine maximale Krümmung wird durch die linke und rechte Schiene begrenzt, allerdings ist es im unteren Bereich z.B. die linke Schiene bei Linkskurvenfahrt, die einen maximalen Winkel aufweist. Das kleine Parallelogramm in der linken unteren Ecke visualisiert diesen Sachverhalt.

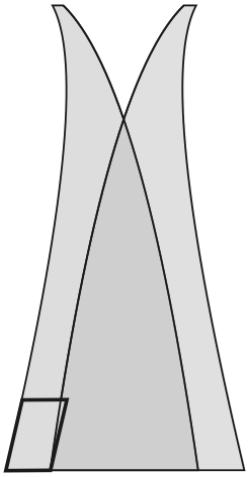

Abbildung 3.4: Maximale Krümmung durch maximale Kurvenfahrten

Der Bereich erlaubter Winkel lässt sich für jedes y_i nun dadurch beschränken, dass an die linken bzw. rechten Schienen bei maximalen Kurvenfahrten im Punkt y_i die Tangente gelegt wird. Abbildung 3.5 zeigt die beiden Tangenten exemplarisch an den kurvenäußeren Schienen für eine Ordinate y_i und den zugehörigen erlaubten Winkelbereich (grau hinterlegt).

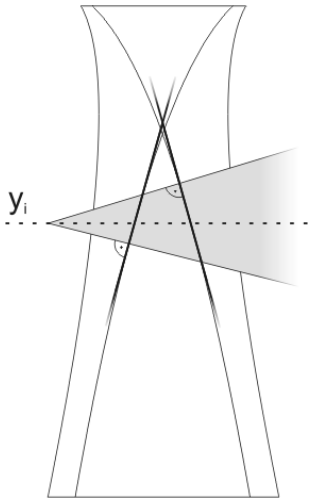

Abbildung 3.5: Tangenten an den Kurven und dadurch ermittelter Winkelbereich

Die Konstruktion der Tangenten beginnt einfacherweise im Weltkoordinatensystem durch Konstruieren der Tangente an dem jeweiligen Kreisbogen. Beispielhaft sei hier die rechte Schiene bei maximaler Linkskurvenfahrt betrachtet. Es wird der Schnittpunkt S_i mit der Ordinate Y_{wi} analog zu Abschnitt 3.1 gebildet durch:

$$S_i = (X_{S_i}, Y_{wi}) \quad \text{mit} \quad X_{S_i} = \sqrt{r_{min}^2 - \left(Y_{wi} + \overline{Y_{Kamera}Y_{Mitte}}\right)^2} - r_{min} + Spurweite$$

Anschließend erfolgt die Konstruktion der Tangente in Polarform durch Ermittlung des Winkels $\alpha = \arctan\left(\frac{Y_{wi}-Y_{R_i}}{X_{S_i}-X_{R_i}}\right)$ und des Tangentenradius mit $r_{tang} = X_{S_i} * \cos(\alpha) + Y_{wi} * \sin(\alpha)$. Der Punkt P_i lässt sich nun als zweiter Tangentenpunkt neben S_i berechnen. Diese beiden Punkte werden zurück in das Bildkoordinatensystem transformiert und es wird der Winkel der sich ergebenden Geraden berechnet. Abbildung 3.6 verdeutlicht die Zusammenhänge der beteiligten Komponenten im Weltkoordinatensystem. Analog dazu werden auch die Winkel der anderen Schienen und Kurvenfahrten berechnet, wodurch sich der begrenzende Winkelbereich ergibt.

In der Praxis ist die Transformation von P_i in das Bildkoordinatensystem problematisch. Da $Y_{P_i} = 0$ gilt, liegt er außerhalb des kalibrierten Bildbereichs, sodass der Fehler bei der Umwandlung zu groß wird. Besser ist es einen zweiten Geradenpunkt so zu berechnen, dass man z.B. $10m$ zur Ordinate von S_i addiert und den entsprechenden Tangentenpunkt ermittelt. Weiterhin ist es notwendig, aufgrund von Umrechnungs- und Kalibrierungsfehlern einen Toleranzbereich zu definieren, unter dem Winkel als gültig anerkannt werden sollen.

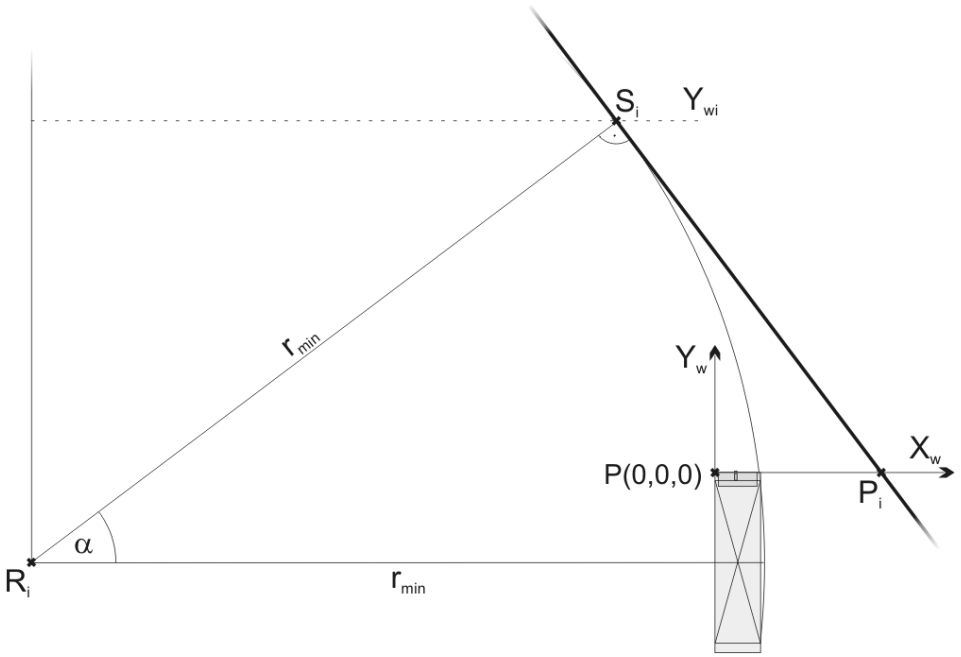

Abbildung 3.6: Berechnung des Winkels auf rechtem Gleis bei max. Linkskurvenfahrt

4 Algorithmische Realisierung

Dieses Kapitel stellt den zentralen Bestandteil der Arbeit dar: es beschreibt zunächst grundlegend den entwickelten Algorithmus zur Schienenextraktion, um dann auf Bildverarbeitungstechniken einzugehen, die dabei zum Einsatz gekommen sind. Darauf aufbauend kann schließlich detailliert auf die einzelnen Algorithmus-Schritte und auf das Einzeichnen des Lichtraumprofils in das Bild eingegangen werden. Am Ende wird anhand einiger Testfälle gezeigt, wo der Algorithmus gut funktioniert und wo es zu Problemen kommen kann. Anhand dessen wird versucht, Bedingungen an die auszuwertenden Bilder zu stellen.

Einschränkend sei hier direkt gesagt, dass der Algorithmus anhand eines einzigen Testvideos erstellt wurde, welches die Firma Eurailscout zur Verfügung gestellt hat. Zur Verbesserung und weiteren Entwicklung des Systems gilt es noch mehr Material auszuwerten und speziell die Arbeitsweise in Extremfällen wie Regen oder Nebel zu betrachten.

4.1 Vorhandene Algorithmen

Hier soll kurz auf einige interessante Algorithmen eingegangen werden, die aus dem Bereich der Spurerkennung und -verfolgung im Straßenverkehr stammen und teilweise erst in den letzten Jahren entwickelt wurden. Dies zeigt bereits, dass die Forschung auf dem Gebiet noch lange nicht abgeschlossen ist und auch zukünftig mit weiteren Entwicklungen gerechnet werden kann.

Ein interessanter Algorithmus stammt aus dem Jahr 2004 und wird in [6] vorgestellt. Jung und Kelber schlagen dort eine Unterteilung des Bildes in Nah- und Fernbereich vor und verwenden ein linear-parabolisches Modell zur Modellierung der Straße. Nach einer initialen Erkennung erfolgt die Extraktion der Fahrspur durch Einpassen des Modells mittels einer Ausgleichslösung, was voraussetzt, dass sich die meisten übergebenen Kantenpunkte auch tatsächlich auf der Fahrbahnbegrenzung befinden. Aufgrund dieser Annahme scheidet das Verfahren hier leider aus, allerdings wird das linear-parabolische Modell zur Beschreibung der Schiene übernommen (siehe 4.6.1.2).

Wang *et al.* stellen in [27] ein Verfahren vor, welches Catmull-Rom-Splines zur Modellierung der Straße benutzt und somit beliebige Kurvenformen ermöglicht. Zudem wird angenommen, dass die Fahrbahnbegrenzungen parallel sind und sich am Horizont schneiden. Mit dem Canny-Kantendetektor wird die Gradientenstärke und -richtung extrahiert und eine Wahrscheinlichkeitsberechnung durchgeführt, welche die Extraktion der beiden Fahrbahnbegrenzungen vorsieht.

Auch dieses Modell ist davon abhängig, dass der Kantendetektor bereits eine recht saubere Fahrspur extrahiert und kann daher nicht weiter berücksichtigt werden.

In [9] wird ein heuristischer Ansatz vorgestellt, der sich auf die Hypothese stützt, dass sich Fahrbahnmarkierungen hell vom Untergrund abheben. Es wurde allerdings schon in Abschnitt 1.3.1 erwähnt, dass ein solcher Ansatz hier nicht möglich ist.

Eine Abhandlung vorhandener Algorithmen, getrennt nach ihrer generellen Vorgehensweise, wird durch [14] gegeben. So nutzt das ALVINN-System z.B. ein Künstliches Neuronales Netz zur Extraktion der Straßenposition. Dieses ist mit einer großen Menge von Beispieldaten zu trainieren und soll dann in der Lage sein neue Bilder klassifizieren zu können. Andere Systeme wie SCARF oder PVR III kommen mit gänzlich unstrukturierten Straßen zurecht, stützen sich also nicht auf die Annahme, dass Fahrbahnmarkierungen vorhanden sind, nehmen die Fahrbahn allerdings als relativ homogenen Farbbereich an. Interessant sind sicher auch die Systeme GOLD und RALPH, welche die Straße als „flach“ annehmen und mit einer vorherigen Kamerakalibrierung das jeweils aktuelle Bild in die Vogelperspektive transformieren. Hier kann die Parallelitätsbedingung der Fahrbahnbegrenzungen viel leichter ausgenutzt werden.

Als Resumé lässt sich festhalten, dass keiner der obigen Algorithmen geeignet ist, um das hier vorhandene Problem zu lösen, vor allem aufgrund der Annahmen, die bei Schienensystemen nicht gültig sind (homogener Farbbereich, hellere Markierungen etc.). So wurde ein eigener Algorithmus entwickelt, der das Bild in horizontale Teilbereiche zerlegt und die Hough-Transformation benutzt um Geradenstücke aus den einzelnen Teilbereichen zu extrahieren. Bestärkt wurde diese Idee durch die kurze Abhandlung eines vergleichbaren Algorithmus ebenfalls in [9].

4.2 Grundlegende Algorithmus-Beschreibung

Die Essenz des Algorithmus zur Schienenerkennung beruht auf der Idee, nicht Kurven oder z.B. Ellipsen-Teilstücke extrahieren zu wollen. Vielmehr wird der zu untersuchende Bildbereich in horizontale Teilstücke zerlegt und die darin vorkommenden Schienen als linear angenommen. Die gesamte Schiene, welche als eine stetige Kurve darstellbar ist, wird somit durch lineare Teile approximiert. Dies erlaubt die Verwendung der *Hough-Transformation für Geraden*, um zunächst Linien aus den einzelnen Teilbereichen zu extrahieren und dann daraus die besten Schienen-Kandidaten zu ermitteln. Die Ergebnisse unterer Bereiche dienen dabei als Ausgangspunkt für das jeweils darüber liegende Gebiet, sodass dort nicht der gesamte Houghraum abgearbeitet werden muss (siehe Abschnitte 4.4 und 4.6.4.2).

Im Pseudocode gestaltet sich der grundlegende Algorithmus zur Verarbeitung eines Bildes wie folgt:

Algorithmus 4.1 Grundlegender Algorithmus zur Schienenerkennung

```
1:  function SCHIENENERKENNUNG(Bild)
2:      n ← Anzahl horizontaler Teilbereiche      ▷ 1 = unterster Bereich, n = oberster Bereich
3:      for i = 1, ..., n do                                  ▷ durchlaufe jeden Teilbereich
4:          BildBearb ← BILDVORVERARBEITUNG(Bild, Bereich[i])
5:          Kantenbild ← KANTENERKENNUNG(BildBearb, Bereich[i])
6:          if i = 1 then                                          ▷ unterster Teilbereich
7:              Linien ← HOUGH(Kantenbild, Bereich[i])
8:              Schiene[i] ← SCHIENENEXTRAKTION(Linien, Bereich[i])
9:          else                                                   ▷ andere Teilbereiche
10:             Linien ← HOUGH(Kantenbild, Bereich[i], Schiene[i-1])
11:             Schiene[i] ← SCHIENENEXTRAKTION(Linien, Bereich[i])
12:     return Schiene
```

Ausgehend von den einzelnen Arbeitsschritten des Algorithmus werden zunächst in den Abschnitten 4.3, 4.4 und 4.5 die grundlegenden verwendeten Bildverarbeitungsalgorithmen vorgestellt, deren Kenntnis erforderlich ist, um in Abschnitt 4.6 detailliert auf den Schienenerkennungs-Algorithmus eingehen zu können. Die eigentliche Einzeichnung des Lichtraumprofils wird dann in Abschnitt 4.7 beschrieben. Es ist zu bemerken, dass die Bildverarbeitung für jeden horizontalen Teilbereich einzeln durchgeführt wird. Dies erlaubt eine bessere lokale Verarbeitung z.B. der in 4.5 beschriebenen Algorithmen und ist bei der Auswahl geeigneter Methoden zu beachten.

4.3 Kantenerkennung

Die Kantendetektion stellt einen unverzichtbaren Bestandteil im gesamten Schienenerkennungs-Algorithmus dar, dienen die hier ermittelten Ergebnisse doch als Ausgangspunkt für die in Abschnitt 4.4 beschriebene Hough-Transformation. Diese kann nur so gut sein wie der zugrunde gelegte Kantendetektor, sodass an diesen hohe Ansprüche gestellt werden. Allgemein lässt sich die Kantenerkennung als kritischsten Teil des Schienenerkennungs-Algorithmus bezeichnen. Werden bereits hier schlechte Ergebnisse geliefert, so sind darauf aufbauende Schritte von vornherein zum Scheitern verurteilt.

Eine Kante kann als Grenze zwischen zwei Bild-Bereichen definiert werden, die für sich genommen relativ homogen sind, im Vergleich zueinander allerdings unterschiedliche Grauwert-Niveaus darstellen ([18]). Sie stellt damit eine Unstetigkeit bzw. Diskontinuität im Grauwertverlauf eines Bildes dar und grenzt i. Allg. mehrere Objekte voneinander ab. Über die Jahre hinweg entstand eine Vielzahl von Algorithmen zur Kantendetektion, welche sich jeweils im verwendeten Kantenmodell, in der Algorithmus-Idee und der Güte ihrer Ergebnisse unterscheiden. Das Gebiet der

Kantenerkennung ist wohl eines der umfangreichsten, die in der Literatur der Bildverarbeitung diskutiert werden (siehe auch [17, 23]).

Unter Kantendetektion versteht man die Extraktion bzw. Hervorhebung von Pixeln in einem Bild, die zu einer Kante gehören. Allen Kantenerkennungs-Operatoren ist gemein, dass sie von einem bestimmten Kantenmodell ausgehen. Die einfachen Detektionsmechanismen legen dabei ein ideales Kantenmodell zugrunde. In [22] werden solche idealen Modelle beschrieben, Abbildung 4.1 stellt diese dar. Durch Unterabtastung erscheint eine Kante im Bild i. Allg. über mehrere Pixel erstreckt, diesem Umstand wird am besten im Modell der Treppenkante Rechnung getragen. Die Schwierigkeit bei der Kantendetektion besteht darin, dass die reale Kantenform zwar einer der 4 idealen Kantenformen bzw. einer Kombination derer entspricht, allerdings durch Störungen wie Rauschen überlagert wird, die nur statistisch charakterisierbar sind. Dies stellt hohe Anforderungen an einen Kantendetektor, der tatsächlich auftretende Kanten von zufälligen Störungen unterscheiden können sollte.

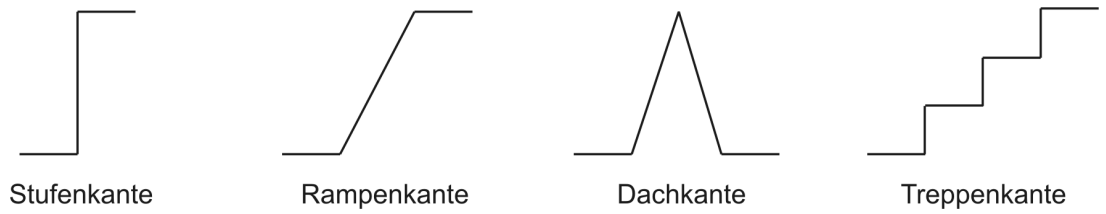

Abbildung 4.1: Ideale Kantenmodelle nach [22]

4.3.1 Anforderungen an einen Kantendetektor

Es gibt verschiedene Kriterien, die ein guter Kantendetektor erfüllen sollte und mit denen sich die Güte eines Verfahrens messen lässt. Die wichtigsten in verschiedenen Metriken verwendeten und auch mathematisch erfassbaren sind:

- **Fehlerfreiheit**: Es sollten genau die Kanten erkannt werden, die auch wirklich im Bild vorhanden sind (Vermeidung von *false-positives* und *false-negatives*), vor allem in Hinblick auf einen hohen Störungsanteil im Ursprungs-Bild.
- **Lokalisation**: Eine erkannte Kante sollte genau dort im Kantenbild erscheinen, wo sie auch im Originalbild vorhanden ist.
- **Eindeutigkeit**: Pro Kante sollte ein Kantendetektor nur eine Antwort liefern.
- **Rechenzeit**: In praktischen Anwendungen ist außerdem noch die Laufzeitkomplexität eines Kantendetektors wichtig - das beste Kantenbild nützt meist nichts, wenn man eine Stunde darauf warten muss.

Die häufig verwendeten Metriken zur Bewertung eines Kantenoperators bauen auf einigen dieser Kriterien auf (z.B. FOM - Figure-of-Merit (siehe [22]) und andere), eine allumfassende gibt es indes nicht.

4.3.2 Differenzoperatoren erster Ordnung

Ein Grauwertbild lässt sich nach [10] als eine von x und y abhängige diskrete Funktion $F(x, y)$ auffassen, welche durch eine weitere Funktion $f(x, y)$ stetig interpoliert werden kann. Eine Stufenkante lässt sich dann im Eindimensionalen wie in Abbildung 4.2a) gezeigt darstellen.

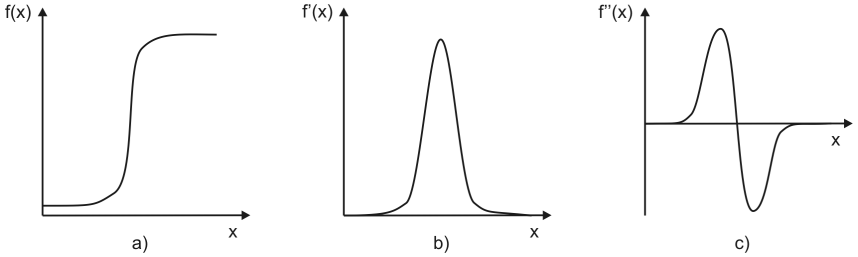

Abbildung 4.2: Stetige Approximation einer Stufenkante mit Ableitungen

Aus Abbildung 4.2b) ist leicht erkennbar, dass im Bereich solch einer Kante die erste Ableitung $f'(x)$ der stetigen Grauwertfunktion $f(x)$ ein Maximum aufweist. Auf dieser Eigenschaft basiert eine Klasse einfacher Kantenoperatoren, die *Differenzoperatoren erster Ordnung*. Analog zu $f'(x)$ wird im zweidimensionalen Bild $f(x, y)$ der Gradient gebildet:

$$\nabla f(x, y) = (G_x, G_y)^T \text{ mit } G_x = \frac{\partial f(x, y)}{\partial x},\ G_y = \frac{\partial f(x, y)}{\partial y}$$

Dieser hat als Vektor die Eigenschaft, dass er in Richtung der größten Grauwertänderung von f an der Stelle (x, y) zeigt und sein Betrag somit ein Maß für die Stärke der Kante ist. Gradientenbetrag und Richtungswinkel γ ergeben sich aus:

$$|\nabla f(x, y)| = \sqrt{G_x^2 + G_y^2} \qquad \text{und} \qquad \gamma = \arctan\left(\frac{G_x}{G_y}\right)$$

Häufig wird der Gradientenbetrag der Einfachheit halber auch berechnet mit:

$$|\nabla f(x, y)| \approx |G_x| + |G_y| \tag{4.1}$$

Diese Approximation erfordert weniger Rechenaufwand und ist so verwendbar, da meist nicht der genaue Wert des Gradientenbetrags interessiert, sondern nur seine relative Größe.

Für digitale Bilder und somit eine diskrete Grauwertfunktion $F(x, y)$ wurden Filtermasken entwickelt, welche bei Faltung mit F die partiellen Ableitungen G_x und G_y approximativ durch die Grauwertdifferenz in einer lokalen Region bestimmen. Im Anschluss daran ergibt sich das Gradientenbild mittels Gleichung 4.1, seine Intensität spiegelt die erkannten Bildkanten wider.

Die wichtigsten Filtermasken dieser Art sind der Roberts-, Prewitt-, Sobel- sowie der isotropische Filter, deren orthogonale Filtermasken H_1 und H_2 aus Tabelle 4.1 entnommen werden können.

	H_1	H_2
Roberts	$\begin{pmatrix} \mathbf{0} & 1 \\ -1 & 0 \end{pmatrix}$	$\begin{pmatrix} \mathbf{1} & 0 \\ 0 & -1 \end{pmatrix}$
Prewitt	$\begin{pmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1 & -1 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{pmatrix}$
Sobel	$\begin{pmatrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1 & -2 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 2 & 1 \end{pmatrix}$
Isotropisch	$\begin{pmatrix} -1 & 0 & 1 \\ -\sqrt{2} & \mathbf{0} & \sqrt{2} \\ -1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1 & -\sqrt{2} & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & \sqrt{2} & 1 \end{pmatrix}$

Tabelle 4.1: Filtermasken von Differenzoperatoren erster Ordnung

Obwohl größere Filtermasken problemlos angewendet werden können, beschränkt man sich meist auf 3x3-Matrizen. Eine Vergrößerung würde die Rauschempfindlichkeit verbessern, bringt aber einen höheren Rechenaufwand mit sich und führt dazu, dass schwächere Kanten verfehlt werden können.

Alle Operatoren haben den erwünschten Effekt, dass sie für homogene Regionen 0 ergeben (die Summe der Elemente einer Filtermatrix ergibt 0), sie unterscheiden sich hingegen hinsichtlich der Güte ihrer Ergebnisse. Eine eingehende Abhandlung kann hier nicht erfolgen, es sei nur festgehalten, dass der *Sobel*-Operator die höchste Güte der in Tabelle 4.1 aufgeführten Operatoren besitzt, da er die Pixel in der 4er-Nachbarschaft des Bezugspixels stärker mit einbezieht. Zudem erfolgt eine Glättung durch die (1 2 1)-Binomial-Filtermaske, die eine diskrete Approximation der Gauß-Verteilung darstellt. Bei größeren Filtermasken (und somit höherer Binom-Ordnung) steigt auch die Approximationsgüte. Da die Gauß-Funktion neben der Dirac-Funktion die einzige Funktion ist, welche sowohl im Orts- als auch im Ortsfrequenzraum dieselbe Form besitzt, wird durch diese spezielle Maske die Auswirkung im Ortsfrequenzraum begrenzt, d.h. es findet eine Glättung statt, Kanten bleiben allerdings erhalten. Nachteilig bei Sobel ist zu bemerken, dass Kanten verhältnismäßig dick dargestellt werden, wodurch feine Strukturen verloren gehen (siehe [7]). Abbildung 4.3 zeigt das Ergebnis einer Kantendetektion mit dem Sobel-Operator.

Differenzoperatoren erster Ordnung liefern bei nahezu rauschfreien Bildern und scharfen Übergängen gute Ergebnisse, versagen allerdings bei starkem Rauschen aufgrund der verwendeten Ableitung. Die Glättung vieler Operatoren (u.a. auch Sobel) kann diesen Effekt vermindern,

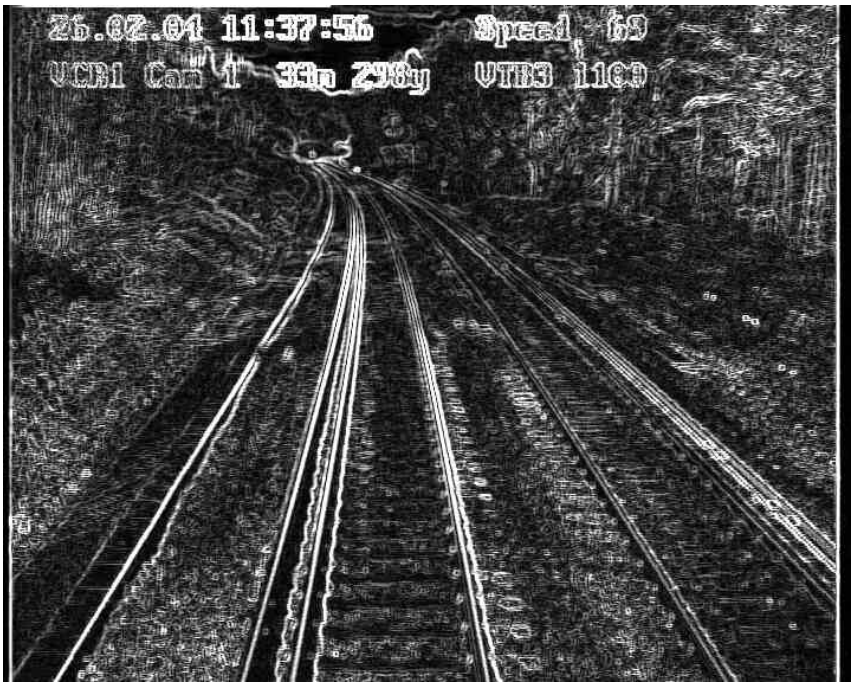

Abbildung 4.3: Ergebnis des Sobel-Operators

meist wird dies jedoch durch breitere und verschwommene Kanten erkauft, was die Kantendetektion hemmt. Ein weiterer Makel der Operatoren auf Basis des Gradienten ist, dass die Stärke des Gradientenbetrags für das Vorhandensein einer Kante entscheidend ist, wodurch schwache Kanten oft unterdrückt werden.

4.3.3 Differenzoperatoren zweiter Ordnung

Gradient-Operatoren arbeiten am besten, wenn Grauwertänderungen sehr abrupt sind. Für größere Übergangsregionen lassen sich die Differenzoperatoren zweiter Ordnung verwenden, deren Möglichkeit zur Erkennung von Kanten sich aus Abbildung 4.2c) ergibt: die zweite Ableitung der Bildfunktion besitzt am Ort einer Kante einen Nulldurchlauf, mit zwei Extremwerten links- und rechtsseitig. In der Regel wird für diese Zwecke der Laplace-Operator eingesetzt, welcher wie folgt definiert ist:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

Dieser ist punktsymmetrisch zum Ursprung und damit isotrop, d.h. er erkennt Kanten unabhängig von ihrer Richtung. Diskret approximiert wird der Operator wieder häufig mit einer 3x3-Filtermaske, welche unterschiedliche Formen annehmen kann, wie Tabelle 4.2 zeigt.

Allen Masken ist zu eigen, dass die Summe ihrer Elemente 0 ist, womit sich für homogene Grauwertbereiche ebenfalls 0 als Antwort ergibt.

	Maske				Maske			
Variante 1	$\begin{pmatrix} 0 & -1 & 0 \\ -1 & \mathbf{4} & -1 \\ 0 & -1 & 0 \end{pmatrix}$				Variante 2	$\begin{pmatrix} -1 & -1 & -1 \\ -1 & \mathbf{8} & -1 \\ -1 & -1 & -1 \end{pmatrix}$		
Variante 3	$\begin{pmatrix} 1 & -2 & 1 \\ -2 & \mathbf{4} & -2 \\ 1 & -2 & 1 \end{pmatrix}$				Variante 4	$\begin{pmatrix} -1 & -2 & -1 \\ -2 & \mathbf{12} & -2 \\ -1 & -2 & -1 \end{pmatrix}$		

Tabelle 4.2: Filtermasken des Laplace-Operators

Nachteile ergeben sich durch das Bilden der 2. Ableitung, womit diese Operatoren noch anfälliger gegenüber Störungen sind als die Gradientoperatoren. Weiterhin werden Doppelkanten ausgebildet. Da außerdem keine Kanten-Richtungen erkannt werden können ([8]), werden die Differenzoperatoren zweiter Ordnung kaum direkt zur Kantendetektion eingesetzt.

Eine bereits angeschnittene Eigenschaft, die trotz allem den Einsatz der 2. Ableitung rechtfertigt, ist der Nulldurchlauf dieser an einer Kante. Extrahiert man die Nulldurchläufe aus einem Bild (siehe auch 4.3.5.1), so ergeben diese neben zufälligen Störungen auch die Kantenpositionen.

Zusammenfassend lässt sich sagen, dass alle auf Ableitungen basierenden Operatoren bei stark verrauschten Bildern schlecht arbeiten. Ebenso wie bei Kanten handelt es sich auch bei Störungen um hochfrequente Bildanteile, welche durch eine Ableitung noch verstärkt werden. Operatoren wie Sobel glätten das Bild durch den speziellen Aufbau der Filtermaske, sodass der Effekt des Rauschens unterdrückt wird, bei Laplace ist es z.B. günstig zuvor eine Gauß-Glättung anzuwenden, was allerdings in verbreiterten Kanten resultiert und somit die Güte der Lokalisation verringert. Zudem werden dadurch auch feine Kanten leichter unterdrückt.

4.3.4 „Optimale“ Operatoren

Die Nachteile der einfachen Operatoren, welche auf Ableitungen basieren, führten dazu, dass man sich stärker mit den biologischen Eigenschaften des visuellen Systems auseinandersetzte und so zu Modellen kam, welche den realen Gegebenheiten besser entsprechen und teilweise Rauschen direkt mit in die Modelleigenschaften einbeziehen. Solche Operatoren stellen Kriterien an einen Kantendetektor. Die Maximierung bzw. Minimierung dieser Kriterien ergibt eine Optimalitätsaufgabe, weswegen man hier von „optimalen“ Operatoren spricht. Sie sollen die Optimalität bezüglich des verwendeten Modells und dessen Kriterien garantieren (*bedingte Optimalität*).

An dieser Stelle soll stellvertretend für die Vielzahl an optimalen Operatoren auf den Canny-Kantendetektor und den Algorithmus von Shen und Castan eingegangen werden, die sehr populär sind und sich in vielen Systemen bewährt haben.

4.3.4.1 Der Canny-Kantendetektor

1986 stellte John Canny eine Reihe von Zielen für einen Kantendetektor auf und beschrieb in [5] eine optimale Methode diese zu erreichen. So sollte nach Canny ein Filter folgenden Kriterien genügen: geringe Fehlerrate, gute Lokalisation und nur eine Antwort pro Kante. Dies führt zu einem mathematischen Optimierungsproblem, welches zu komplex ist um es analytisch lösen zu können. Canny fand jedoch eine effiziente Approximierung in der ersten Ableitung der Gauß-Funktion (hier im Eindimensionalen und nach [17]):

$$G_\sigma(x) = e^{-\frac{x^2}{2\sigma^2}} \qquad G'_\sigma(x) = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

Durch Faltung eines Bildes mit $G'_\sigma(x)$ erhält man ein Bild mit verstärkten Kanten, selbst bei verrauschten Bilddaten, sodass Störungen als Bestandteil der Modellierung angesehen werden können.

Der gesamte Algorithmus inklusive einer effizienten Implementierung ergibt sich wie folgt:

1. Erzeuge eine 1D-Gauß-Maske G in Abhängigkeit der Standardabweichung σ.
2. Erzeuge 1D-Masken für die 1. Ableitung der Gauß-Funktion, G_x und G_y.
3. Falte das Eingangsbild F zeilen- und spaltenweise mit G, um die x- bzw. y-Komponenten-Bilder F_x und F_y zu erhalten.
4. Führe die Faltungen $F'_x = F_x * G_x$ und $F'_y = F_y * G_y$ durch, um die x- und y-Komponenten des Eingangsbildes zu erhalten, auf welches die 1. Ableitung der Gauß-Funktion angewendet wurde.
5. Erzeuge das Gradientenbild M mittels Gleichung 4.1 aus F'_x und F'_y.
6. Wende *Non-Maximum-Suppression* auf das Gradientenbild M und die Richtungskomponenten an, um tatsächliche Kantenpixel zu ermitteln, ohne schwache Kanten zu unterdrücken (siehe 4.3.5.2).
7. Wende *Hysteresis-Tresholding* an, um das Bild zu binarisieren und Kantenzusammenhänge zu extrahieren (siehe 4.3.5.3).

Es ist erkennbar, dass der Canny-Filter auf der klassischen Vorstellung der Anwendung eines Ableitungsoperators auf ein geglättetes Bild beruht. Prinzipiell ist auch die gesamte grundlegende Vorgehensweise nicht neu: auf den eigentlichen Kantendetektor folgt eine Nachbearbeitung in Form der Non-Maximum-Suppression und des Hysteresis-Tresholding. Diese Kombination ist allerdings so geschickt gewählt, dass die Optimalität bezüglich der vorausgesetzten Kriterien angenähert werden kann.

Die vom Canny-Algorithmus gelieferten Ergebnisse sind denen der einfachen Ableitungsoperatoren überlegen, auch bei verrauschten Bildern werden Kanten gut erkannt. Das Erhöhen von σ zur

Berechnung der Filtermasken führt zu einer besseren Glättung, allerdings auch zu schlechterer Kantenlokalisation, ebenso können sich eng benachbarte Kanten gegenseitig beeinflussen. Canny schließt diesen Fall aus, indem er davon ausgeht, dass sich nur eine Kante im Einzugsbereich einer Filterfunktion befindet. Diese Annahme ist in der Praxis allerdings oft nicht haltbar, vor allem bei sich kreuzenden Kanten. Dies führt dazu, dass der Canny-Operator in solchen Bereichen nicht sehr gut funktioniert und es zu unerwünschten Unterbrechungen kommen kann. Diese lassen sich (wie in [22] beschrieben) z.B. durch *Constraint-Thinning* zumindest teilweise wieder schließen, was hier jedoch nicht betrachtet werden soll, da es in diesem Kontext weniger relevant ist.

Abbildung 4.4 zeigt das Ergebnis einer Kantendetektion mit dem Canny-Algorithmus für $\sigma = 1$ und automatisch ermittelten Schwellwerten.

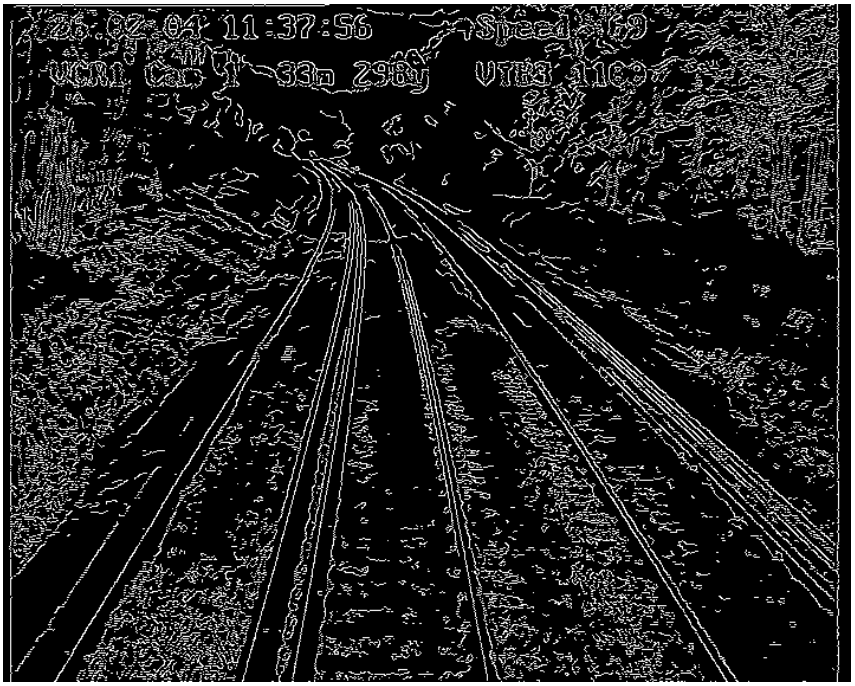

Abbildung 4.4: Anwendung des Canny-Kantendetektors

4.3.4.2 Der Shen-Castan-Kantendetektor (ISEF)

Dies ist ein weiterer optimaler Operator, der 6 Jahre nach dem Canny-Operator 1992 von Shen und Castan vorgeschlagen wurde und z.B. in [17] vorgestellt wird. Selbst wenn Canny's Operator optimal bezüglich der von ihm spezifizierten Kriterien ist, heißt das nicht, dass diese Kriterien nicht modifiziert werden können. Dies führt zu anderen Optimierungsaufgaben, die es zu lösen gilt. Der Filter, welcher sich daraus ergibt, ist wiederum optimal bezüglich der vorgegebenen Kriterien. Tatsächlich scheint der Vergleich von Kantendetektions-Techniken oftmals mehr auf einen Vergleich verschiedener Definitionen von Optimalität anstatt auf einen Vergleich der eigentlich

eingesetzten Schemata zur Kantenermittlung hinaus zu laufen.

Shen und Castan stimmen mit Canny in der allgemeinen Form eines Kantendetektors überein: das Bild ist zunächst zu glätten, um dann die eigentliche Kantensuche durchzuführen. Ihre Analyse führt allerdings zu einem anderen Optimierungsproblem, dessen Lösung einen optimalen Glättungsfilter für die Kantendetektion ergibt, welcher durch folgende Funktion gegeben ist:

$$f(x) = \frac{p}{2} e^{-p|x|} \qquad f(x,y) = a \cdot e^{-p(|x|+|y|)}$$

Dieser Filter wird als *unendlicher symmetrischer exponentieller Filter* (infinite symmetric exponential filter (ISEF)) bezeichnet. Er führt zu einem günstigeren Signal-zu-Rausch-Verhältnis und ergibt eine bessere Lokalisation von Kanten. Auf der anderen Seite beziehen Shen und Castan das Problem mehrerer Antworten auf eine Kante nicht in ihr Modell mit ein, sodass ihr Filter auf eine Kante unscharfe und verrauschte Ergebnisse liefern kann.

Eine effiziente Realisierung des ISEF ließe sich wie bei Canny durch 1D-Filter für die x- und y-Richtung entwickeln, allerdings wird hier ein anderer Weg verfolgt. Es kommen *rekursive Filter* zum Einsatz, wodurch die Faltung stark beschleunigt wird (diese Thematik soll hier nicht weiter betrachtet werden, [17] gibt z.B. eine kurze Einführung).

Der gesamte Shen-Castan-Algorithmus im kurzen Überblick:

1. Wende den ISEF-Filter auf das Eingangsbild $F(x,y)$ an, was Bild $I(x,y)$ ergibt.
2. Erstelle aus $I(x,y)$ das binäre Laplace-Bild $\nabla^2 I(x,y)$.
3. Ermittle tatsächliche Nulldurchläufe in $\nabla^2 I(x,y)$ durch *false zero-crossing suppression*, speichere diese im Bild $Z(x,y)$.
4. Wende für alle in $Z(x,y)$ gesetzten Pixel unter Einbeziehung des Eingangsbildes $F(x,y)$ den *adaptiven Gradienten* an, was zu Bild $A(x,y)$ führt.
5. Führe auf $A(x,y)$ das in 4.3.5.3 beschriebene *Hysteresis-Tresholding* aus.

Prinzipiell werden Kanten also dadurch gefunden, dass man die Nulldurchgänge des Laplace-gefalteten Bildes ermittelt. Auf die einzelnen Optimierungsschritte soll nachfolgend noch kurz eingegangen werden.

Das Laplace-Bild kann durch Subtraktion des Originalbildes $F(x,y)$ vom ISEF-geglätteten Bild $I(x,y)$ approximiert werden:

$$F(x,y) * \nabla^2 g(x,y) \approx I(x,y) - F(x,y)$$

Das resultierende Bild ist das *band-begrenzte Laplace-Bild*. Hiervon wird das *binäre Laplace-Bild* erzeugt, indem alle positiven Werte auf 1 (bzw. 255) und alle anderen Werte auf 0 gesetzt werden.

Kandidaten für Kantenpixel liegen dann auf den Grenzen der Regionen im binären Laplace-Bild vor und können als Nulldurchläufe erkannt werden.

Erkennt man Nulldurchläufe nach der einfachen in 4.3.5.1 beschriebenen Methode, so führt dies zu vielen Fehlerkennungen durch verrauschte Bilddaten. Shen und Castan nutzen anstelle dessen das *false zero-crossing suppression*-Verfahren. Dabei nutzt man aus, dass die erste Ableitung an einer Kante ein Maximum (aufsteigende Kante) bzw. Minimum (absteigende Kante) besitzt. Definiert sei der „positive Nulldurchlauf“ im band-begrenzten Laplace-Bild als ein Vorzeichenwechsel von positiv nach negativ und der „negative Nulldurchlauf“ als ein Vorzeichenwechsel von negativ nach positiv. Für Nulldurchläufe müssen dann folgende Bedingungen gelten: positive Nulldurchläufe müssen auch einen positiven Gradienten haben, negative Nulldurchläufe hingegen einen negativen Gradienten. Nulldurchläufe, welche diesen Bedingungen nicht genügen, sind auszuschließen und nicht weiter zu betrachten. Wie der Name des Verfahrens bereits besagt, werden somit „falsche“ Nulldurchgänge unterdrückt.

Für die Nulldurchläufe, welche aus dem vorangegangenen Schritt übrig bleiben, wird nun das Schwellwertverfahren des *adaptiven Gradienten* angewendet. Dabei wird ein Fenster der fixen Größe W auf jedem Kandidatenpixel (das einen echten Nulldurchlauf darstellt) zentriert. Wenn es sich tatsächlich um ein Kantenpixel handelt, so sind in W zwei Regionen mit unterschiedlichen Grauwerten vorhanden, die durch eine Kante (Linie von Nulldurchläufen) getrennt werden, zu der auch das Kantenpixel gehört. Es wird also für das Kandidatenpixel der Gradientenbetrag berechnet und in ein Bild $A(x, y)$ eingetragen. Der Gradient wird dabei durch die Grauwertdifferenz der beiden Regionen unter W approximiert. Eine Unterscheidung beider Regionen findet anhand ihres Wertes im binären Laplace-Bild statt.

Mit dem Bild $A(x, y)$ der Gradientenbeträge wird schlussendlich durch Anwendung des *Hysteresis-Tresholding* über die tatsächliche Zugehörigkeit eines Pixels zu einer Kante entschieden.

Im Vergleich zu Canny besitzt der Shen-Casten-Detektor vor allem bei stark verrauschten Bildern Vorteile, hier werden Kanten noch vergleichsweise gut erkannt. Nachteilig sind teilweise verrauschte Kanten nach der Detektion, da pro Kante mehrere Antworten geliefert werden können. Alles in allem hängt es schlussendlich vom Anwendungsgebiet ab, für welchen Operator man sich entscheidet. Bei dem hier vorkommenden Schienenerkennungs-Problem könnten beide Operatoren zum Einsatz kommen, da beide gute Ergebnisse liefern. Allerdings finden beim Shen-Castan-Filter weniger Störungen Einzug in das Kantenbild, wie ein direkter Vergleich der Abbildungen 4.4 und 4.5 zeigt. Verrauschte Kanten stellen kein Problem dar, weil sie die Ergebnisse der Hough-Transformation (HT) kaum beeinträchtigen (siehe 4.4). Da außerdem die Laufzeit vergleichbar ist, wird der ISEF-Kantendetektor hier bevorzugt verwendet. Das einzige Manko ist, dass der Gradientenbetrag durch den Shen-Castan-Algorithmus zwar geschätzt und somit für die HT verwendet werden kann, allerdings keine Ermittlung der Gradientenrichtung möglich ist. Wird diese benötigt, um z.B. die Kantenpixel anhand ihres Winkels nachträglich

zu filtern (wie in Abschnitt 4.3.5.4 beschrieben), so muss zusätzlich beispielsweise der Sobel-Operator angewendet werden, um die Gradientenrichtung der einzelnen Pixel zu erhalten.

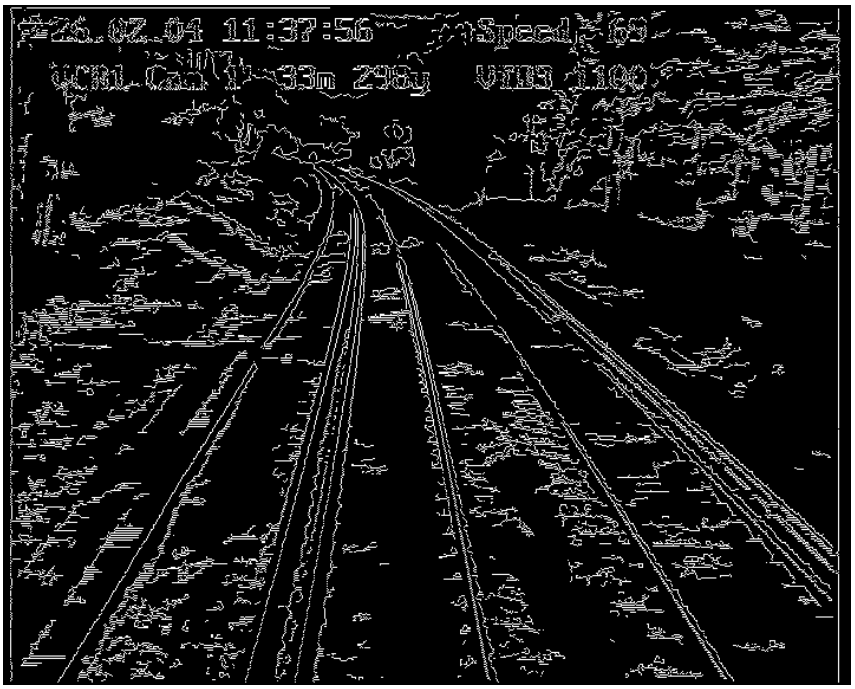

Abbildung 4.5: Anwendung des Shen-Castan-Kantendetektors

4.3.5 Kanten-Nachbearbeitungs-Verfahren

Im Idealfall würden Kantendetektoren dort und nur dort eine Antwort liefern, wo Kanten im Bild vorhanden sind. Da das Ursprungsbild aber nicht störungsfrei ist, muss nach der eigentlichen Kantendetektion im Normalfall noch eine Nachbearbeitung stattfinden, die über die tatsächliche Zugehörigkeit eines Pixels zu einer Kante entscheidet und erkannte Kanten auf 1 Pixel verdünnt. Die allgemein verwendeten und bei den optimalen Operatoren schon mit einbezogenen Verfahren sollen nachfolgend kurz beschrieben werden.

4.3.5.1 Nulldurchläufe erkennen

Diese Nachbearbeitung wird i. Allg. nach der Anwendung des Laplace-Operators verwendet, ansonsten hat sie kaum Bedeutung. Sinn und Zweck ist, dass die 2. Ableitung an einer Kante einen Nulldurchlauf aufweist und dieser extrahiert werden soll.

Das Bild wird mit dem Verfahren binarisiert. Ein Nulldurchlauf liegt genau dann an einem Pixel vor, wenn es in mindestens einer Richtung einen Nulldurchlauf gibt. Es sind also 4 Fälle zu betrachten: in senkrechter, horizontaler und den zwei diagonalen Richtungen. Unterscheiden sich

die Vorzeichen gegenüberliegender Nachbarpixel in mindestens einem der 4 Fälle, so liegt ein Nulldurchlauf vor und das Pixel wird mit 1 bzw. 255 markiert.

4.3.5.2 Non-Maximum-Suppression

Wurde ein Gradientenbild berechnet, so liefern Schwellwertverfahren als Nachbearbeitungsschritt oft zu schlechte Ergebnisse, da schwache Kanten eliminiert werden. In diesem Fall ist es sehr viel effizienter, anhand der Gradientenrichtung zu entscheiden, welche Kanten im Bild erhalten bleiben sollen. Die grundlegende Idee ist, dass Kanten eine Richtung besitzen und der Gradientenbetrag eines Kantenpixels größer sein sollte als die Gradientenbeträge der Pixel auf beiden Seiten der Kante.

Zur Bestimmung der Gradientenrichtung nutzt man die Werte aus, welche man durch Ableitung in x- und y-Richtung als G_x sowie G_y errechnet hat. Nun kann man jedoch nicht voraussetzen, dass die Richtung des Gradienten eines Pixels genau auf ein Nachbarpixel zeigt. Das nächstliegende Pixel zu nehmen wäre auch zu ungenau, da zwei unterschiedliche Nachbarpixel mindestens einen 45°-Winkel zum Bezugspixel aufspannen (im ungünstigsten Fall hätte man so einen unvertretbaren Fehler von 22.5°). Die Lösung liegt hier in der linearen Interpolation der Gradientenstärke durch Einbeziehung der zwei Nachbarpixel, auf welche der Gradient des Bezugspixels zeigt.

Aus der Interpolation erhält man zwei Werte D_1 und D_2 für die Gradientenstärke in Richtung des Gradienten und entgegen dieser. Nun werden nur die Bezugspixel mit ihrer Intensität behalten, deren Gradientenbetrag $D(x, y)$ größer ist als von den Nachbarpixeln. Es muss also gelten: $D(x, y) \geq D_1$ und $D(x, y) \geq D_2$. Ist dies nicht der Fall, so wird die Pixelintensität im Ergebnisbild auf 0 gesetzt.

4.3.5.3 Hysteresis-Tresholding

Dieses Schwellwertverfahren ist mit in den Canny- und Shen-Castan-Kantendetektoren verankert, kann aber auch herausgelöst aus diesen verwendet werden. Ziel ist es zu entscheiden, ob es sich bei einem Punkt aufgrund seines Wertes (bei Canny und Shen-Castan: seines Gradientenwertes) um einen Kantenpunkt handelt oder nicht.

Das Verfahren arbeitet mit einem oberen Schwellwert S_o und einem unteren Schwellwert S_u ($S_u < S_o$). Es beruht auf der Beobachtung, dass herkömmliche Schwellwertverfahren häufig Unterbrechungen in Kanten liefern, die auf zeitweises Unterschreiten des Pixelwertes zurückzuführen sind und dass schwache Kanten zu schnell eliminiert werden.

Der Algorithmus arbeitet nach folgendem Prinzip: jedes Pixels wird einzeln betrachtet. Liegt sein Wert unter dem von S_o, so wird es übergangen. Übersteigt sein Wert jedoch diesen oberen Schwellwert, dann handelt es sich um ein Kantenpixel. Weiterhin wird jedes Pixel in der Nachbarschaft eines solchen Pixels betrachtet. Übersteigt dessen Wert den Schwellwert S_u, so handelt es sich dabei ebenfalls um ein Kantenpixel. Auch dessen Nachbarpunkte werden dann verarbeitet. Dies führt zu einer rekursiven Abarbeitung, bis kein Nachbar eines Kantenpixels mehr den unteren Schwellwert erreicht. Mit dieser Vorgehensweise ist das Verfahren als einfacher Konturverfolgungs-Algorithmus interpretierbar. Die Schwellwerte sollten an die jeweilige Situation von Hand angepasst werden, was allerdings anhand von Histogramm-Untersuchungen auch adaptiv geschehen kann.

Das Verfahren kann als sehr robust angesehen werden. Es liefert verlässliche Ergebnisse in der Kanten-Nachbearbeitung und trennt zufällige Störungen wie Rauschen von tatsächlichen Kanten, selbst wenn diese nur schwach im Bild vorhanden sind. Die Unterbrechungsfreiheit von Kanten bleibt auf längere Strecken hin erhalten, was ebenfalls eine wichtige Eigenschaft des Verfahrens ist.

4.3.5.4 Winkelfilterung

Die Gradientenrichtung eines jeden Pixels, welche z.B. bei Sobel und Canny direkt ausgegeben werden kann, ist ein Maß für die jeweilige Kantenrichtung. Wurde wie in 3.2 beschrieben eine Begrenzung zulässiger Winkel ermittelt, so kann diese Information nun für eine weitere Kantennachbearbeitungs-Technik genutzt werden. Dabei betrachtet man jedes Pixel im ermittelten Kantenbild und vergleicht dessen Gradientenrichtung mit dem zugehörigen Bereich zulässiger Winkel. Liegt sie außerhalb, so wird der jeweilige Punkt aus dem Kantenbild ausgeschlossen.

Abbildung 4.6 zeigt die Winkelfilterung als Nachbearbeitungsschritt des Canny-Kantendetektors. Gut zu erkennen ist die hohe Anzahl von Bildpunkten, die dadurch eliminiert werden können und nicht mehr störend in die HT einfließen. Wird der Shen-Castan-Kantendetektor eingesetzt, kann eine Winkelfilterung nur dadurch erreicht werden, dass man die Gradientenorientierung z.B. mittels Sobel-Operator nachträglich ermittelt. Da das Shen-Castan-Verfahren jedoch recht „saubere“ Kantenbilder liefert, kann häufig auf eine nachträgliche Winkelfilterung verzichtet werden.

4.4 Hough-Transformation

Der in Abschnitt 4.2 beschriebene grundlegende Algorithmus zur Schienenerkennung stützt sich wesentlich auf die *Hough-Transformation* (HT), ein mathematisches Verfahren zur Erkennung von beliebigen geometrischen Figuren in einem Bild, welches nachfolgend vorgestellt werden soll.

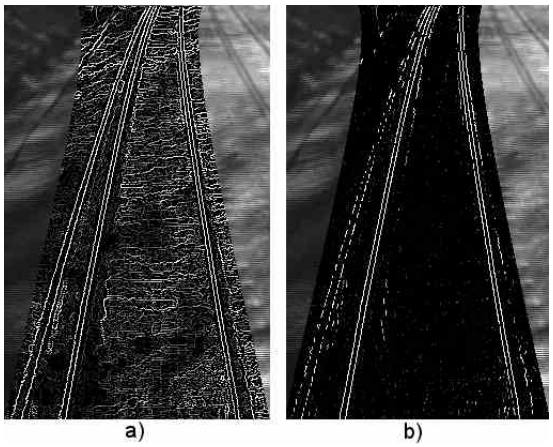

Abbildung 4.6: a) Kantenbild nach Canny, b) winkelgefiltertes Kantenbild

Als Ausgangspunkt diene dabei das mit Hilfe eines Kantendetektors ermittelte Kantenbild. Die hier fest gehaltenen Kanten sind sowohl für das menschliche Auge als auch für die Bildverarbeitung wichtige Informationsträger. Durch die Extraktion dieser mit einem Kantendetektor findet allerdings noch keine Interpretation statt: es wird keine Aussage darüber getroffen, was die Daten im Kantenbild eigenlich bedeuten. Ein weiteres Problem ist die eingeschränkte lokale Sicht des Kantendetektors. So enthalten Kantenbilder in der Regel zahlreiche irrelevante Strukturen, bedeutsame sind hingegen häufig unvollständig. Die Hough-Transformation stellt hier ein wichtiges Verfahren zur Nachbearbeitung eines Kantenbildes dar. Sie wurde von Paul Hough ursprünglich als US-Patent veröffentlicht und ist eine robuste globale Methode zur Bildanalyse, mit der beliebige parametrisierbare geometrische Figuren wie Geraden, Kreise oder Ellipsen gefunden werden können. Diese Arbeit beschränkt sich auf die hier benötigte Erkennung von Geraden.

4.4.1 Parameterraum

Allgemein lässt sich eine Gerade in der *Hesse'schen Normalform* (HNF) darstellen:

$$r = x \cdot \cos(\phi) + y \cdot \sin(\phi) \qquad \text{mit } \phi \in [0 \ldots \pi],\ r \in R \tag{4.2}$$

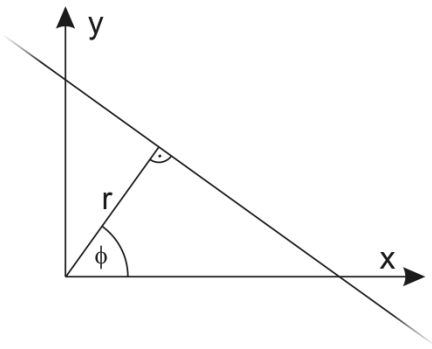

Abbildung 4.7: Geradengleichung in HNF

Dient das (zunächst binäre) Kantenbild als Ausgangspunkt, so ist nun das Ziel, solche Parameter r und ϕ zu finden, auf deren Gerade möglichst viele Kantenpunkte liegen. Ein trivialer Ansatz wäre, alle möglichen Geraden im Bild zu erzeugen, was aufgrund der Diskretheit des Bildraums zwar möglich ist, allerdings eine zu rechenaufwändige Brute-Force-Methode darstellt.

Bei der Hough-Transformation wird ein anderer Weg eingeschlagen. Durch die beiden Geradenparameter r und ϕ wird ein zweidimensionaler so genannter *Parameterraum* (auch *Houghraum* oder *$r\phi$-Raum*) aufgespannt. Während im Bildraum über Gleichung 4.2 eine Gerade mit den Parametern r und ϕ und den Variablen x und y definiert wird, erhält man im Parameterraum mit den Parametern x und y und den Variablen r und ϕ eine Sinoide. Somit korrespondiert jeder Punkt P_i im Bildraum über die HNF mit einer sinusförmigen Kurve im Parameterraum und jeder Punkt g_i im Parameterraum mit einer Geraden im Bildraum. Abbildung 4.8 verdeutlicht diese Wechsel-Beziehung zwischen Bildraum und Parameterraum.

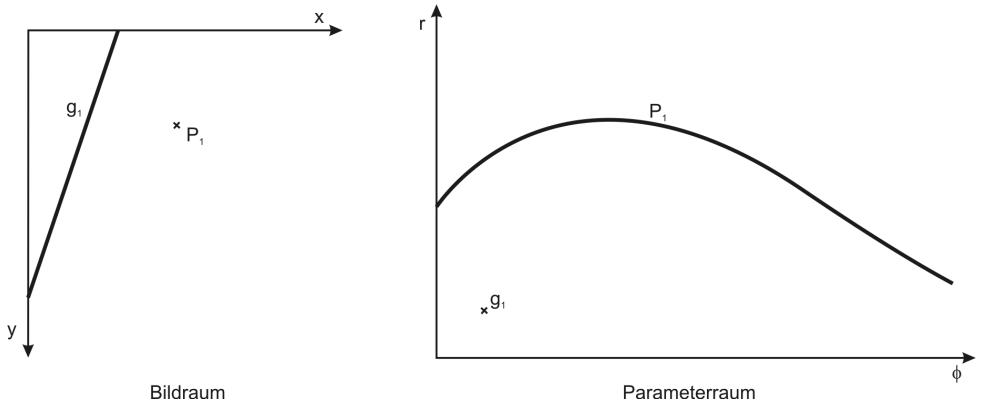

Abbildung 4.8: Beziehung zwischen Bild- und Parameterraum

Da eine Gerade im Bildraum durch einen Punkt g_i im Parameterraum repräsentiert wird, muss dieser Punkt g_i in Zusammenhang mit allen Punkten P_i im Bildraum stehen, welche auf dieser Geraden liegen. Diese Beziehung wird dadurch widergespiegelt, dass die Kurven aller Punkte P_i im Parameterraum den gemeinsamen Schnittpunkt g_i haben. Abbildung 4.9 verdeutlicht diesen Sachverhalt.

4.4.2 Akkumulator-Array

Aus dem vorigen Abschnitt geht hervor, wie sich Geraden im Bildraum finden lassen: man sucht einfach nach Punkten im Parameterraum, an denen sich möglichst viele Kurven schneiden. Diese Schnittpunkte stellen dann die stärksten Geraden im Bild dar.

Ein Algorithmus zur Geradenermittlung kann natürlich nicht auf einem stetigen Parameterraum basieren, das verbietet bereits der diskrete Bildraum, wodurch mehrere Punkte auf derselben Geraden nie genau in einem Schnittpunkt resultieren, sondern vielmehr in einem „Schnittpunkt-

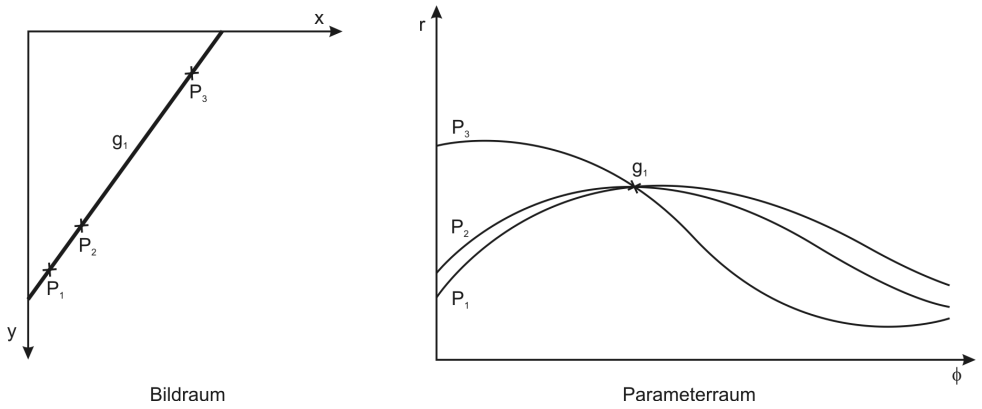

Abbildung 4.9: Schnittpunkt-Beziehung der Geradenpunkte

Gebiet“. Diesem Gebietsbegriff wird durch die Diskretisierung des kontinuierlichen Parameterraumes Rechnung getragen, die sich im zweidimensionalen *Akkumulator-Array* ausdrückt.

Wichtig ist die Parametrisierung des Akkumulator-Arrays. Wenn das Zentrum des Bildes als Referenzpunkt für die Bildkoordinaten benutzt wird, dann beschränkt sich der mögliche Bereich für den Radius auf die Hälfte der Bilddiagonale (vgl. [26]), d.h. für ein Bild der Größe $M \times N$:

$$r_{max} = -r_{min} = \frac{1}{2}\sqrt{M^2 + N^2}$$

Der Winkelbereich kann entsprechend Gleichung 4.2 auf $0 \leq \phi \leq \pi$ beschränkt werden.

Wie groß das Akkumulator-Array gewählt wird stellt einen entscheidenden Faktor für den Erfolg der HT dar. Nimmt man beispielsweise 90 Elemente in der ϕ-Dimension, so entspricht das einer Auflösung von 2 Grad pro Akku-Zeile. Wählt man den Akkumulator zu klein, so umfasst jede Zelle einen zu großen Winkel- bzw. Radius-Bereich. D.h. dass Punkte z.B. auf eine Gerade gelegt werden, obwohl sie zu benachbarten parallelen oder annähernd parallelen Geraden gehören. Andererseits führt ein großes Akkumulator-Array zu einer höheren Laufzeit und einer vielleicht zu hohen Auflösung des Parameterraumes, bei der Punkte, die auf derselben Gerade liegen sollen, zu verschiedenen Linien gezählt werden. Die sorgfältige Wahl der Akku-Dimensionen ist somit unverzichtbar. Im Kontext dieser Arbeit unter Verwendung eines stückweise linearen Modells sollte die Auflösung nicht zu hoch sein, sodass leichte Krümmungen in einzelnen Segmenten noch als Gerade erkannt werden können.

4.4.3 Algorithmus

Die Vorgehensweise zur Geradenermittlung gestaltet sich nun wie folgt. Für jeden im Kantenbild eingetragenen Punkt wird die zugehörige Kurve im Parameterraum durchlaufen und die entsprechenden Akku-Elemente inkrementiert. Im Anschluss daran werden die Maxima im Akkumulator gesucht - diese entsprechen schließlich den stärksten Bildgeraden. Algorithmus 4.2 fasst diese Ideen zusammen.

Algorithmus 4.2 Algorithmus zur Hough-Transformation

```
1:  function HoughTransformation(Kantenbild)
2:      for all Kantenbildpunkte (u,v) do                ▷ 1.) Aufsummierung des Akkus
3:          if Kantenbild[u, v] ≠ 0 then
4:              (x, y) ← (u − u_c, v − v_c)              ▷ Koordinaten relativ zum Bildzentrum
5:              for φ = 0 ... π do
6:                  r = x · cos(φ) + y · sin(φ)
7:                  (i, j) ← Diskretisiere(r, φ)         ▷ Akku-Indizes ermitteln
8:                  Akku[i, j]++
9:      Linien ← AkkuAuswertung(Akku)                    ▷ 2.) Auswerten des Akkus
10:     return Linien
```

4.4.4 Aufsummierung des Akkumulator-Arrays

Die grundlegende Aufsummierung des Akkumulator-Arrays wurde bereits in Algorithmus 4.2 dargestellt, allerdings gibt es einige Verbesserungen, die sich hier mit einbringen lassen:

1. Berücksichtigung der Kantenstärke: Statt den Akku nur zu inkrementieren, wenn das entsprechende Pixel im Kantenbild $\neq 0$ ist, lässt sich eine gewichtete Aufsummierung abhängig von der Stärke einer Kante durchführen. Dies stellt eine sinnvolle Verbesserung dar, da schwache Kanten einen geringeren Einfluss bekommen als starke. Ausgangspunkt muss dabei ein Kantenbild sein, das Grauwertverteilungen entsprechend den Gradientenbeträgen der einzelnen Pixel enthält. Statt einer Inkrementation findet nun eine Summierung über die Gradientenbeträge statt, wodurch direkt die Kantenstärke mit einbezogen wird.

2. Begrenzung des Winkelbereichs: Die Winkelbegrenzung, wie sie in Abschnitt 3.2 besprochen wurde, führt zu einer weiteren Effizienzsteigerung der HT. Da für jede Ordinate y_i ein eingeschränkter Winkelbereich existiert, muss das Akkumulator-Array nur für solche ϕ aktualisiert werden, die innerhalb dieses Bereiches liegen. Damit lassen sich viele Sinus- und Kosinus-Berechnungen einsparen. Abbildung 4.10 zeigt die Unterschiede deutlich auf.

4.4.5 Auswerten des Akkumulator-Arrays

Der zweite und wichtigste Schritt der HT besteht im Finden der Maxima im Akkumulator-Array. Erschwert wird er dadurch, dass aufgrund diskreter Koordinatengitter Maxima nicht exakt in einer Akku-Zelle liegen, sondern über mehrere Elemente verteilt sind. Es gibt für dieses Problem keine Patentlösung, dies ist wiederum auch abhängig von der gegebenen Aufgabenstellung.

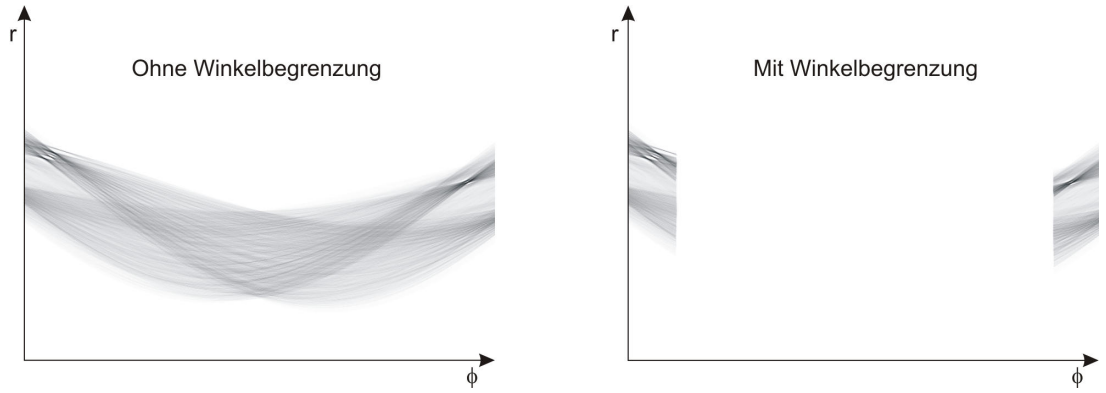

Abbildung 4.10: Akku-Summation ohne und mit Winkelfilterung

Oftmals findet zunächst die Unterdrückung nicht-maximaler Werte mittels *Non-Maximum-Suppression* statt, worauf dann eine Schwellwertoperation folgt, wie es Abbildung 4.11 zeigt. Non-Maximum-Suppression wird z.B. auch in Abschnitt 4.3.5.2 als Nachbearbeitungsschritt der Kantendetektion beschrieben. Die Eignung des Verfahrens hängt wesentlich von der Größe des Akkumulator-Arrays ab. Erst aber einer bestimmten Auflösung dessen ist eine Unterdrückung nicht-maximaler Werte vorteilhaft anzusehen, weil dadurch eine große Menge schlechter Kandidaten ausgeschlossen wird. Ist der Akku hingegen recht klein, so werden durch Anwendung dieser Methode schwächere Kandidaten benachbarter Zellen eliminiert, was sich als ungeeignet herausgestellt hat, da beim Problem der Schienenerkennung viele Schienen-Kandidaten in sehr geringem Abstand zueinander liegen und somit auch benachbarte Zellen im Akkumulator-Array einnehmen. In diesem Fall sollte lediglich eine einfache Schwellwertoperation zur Anwendung kommen. So erhält man zwar *Regionen* von Maximalwerten, jedoch werden keine Kandidaten ausgeschlossen, die möglicherweise Beachtung finden müssen.

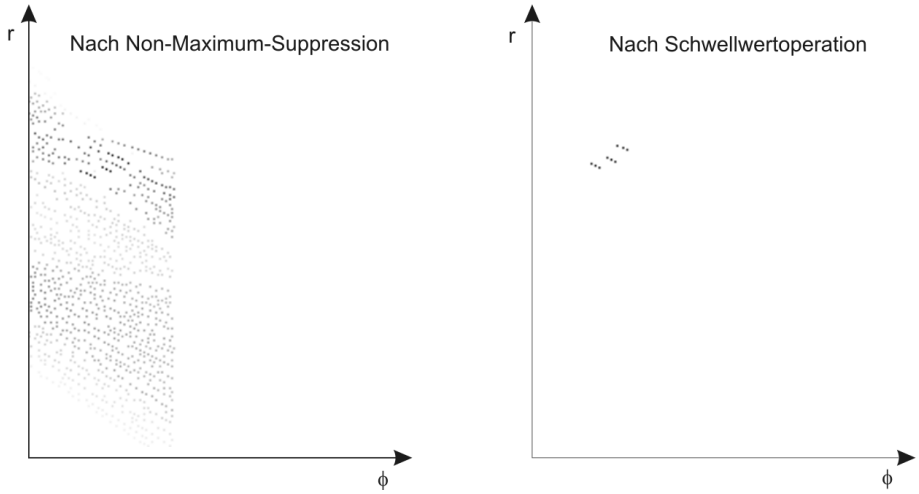

Abbildung 4.11: Akku nach Non-Maximum-Suppression und Schwellwertoperation

Eine einfache Schwellwertoperation könnte die stärksten Kandidaten extrahieren, sodass z.B. alle Akku-Elemente erhalten bleiben, deren Betrag mindestens 80% des Maximalbetrags im Akku

entspricht. Abbildung 4.12 zeigt das Ergebnis nach der Operation mit einem Schwellwert von 0.8 (entspricht 80% des Maximalwertes). Zur besseren Visualisierung wurde dabei nur der linke ϕ-Bereich von Abbildung 4.10 verwendet. Die übrig bleibenden Punkte entsprechen den HNF-Parametern der stärksten Geraden im Bild.

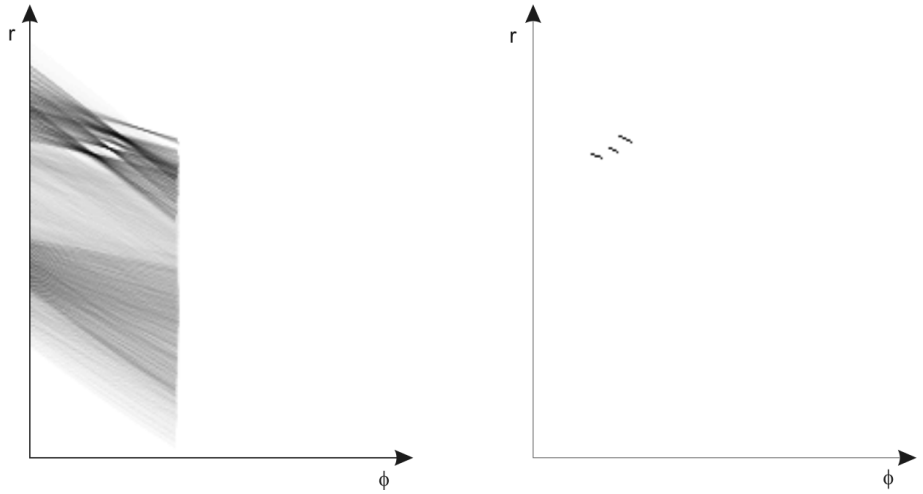

Abbildung 4.12: Akku vor und nach einfacher Schwellwertoperation

Abbildung 4.13 zeigt den untersten Teilbereich eines auszuwertenden Bildes in jeweils 3 Verarbeitungsschritten. Teilbild c) stellt dabei die mittels HT ermittelten Bildgeraden dar.

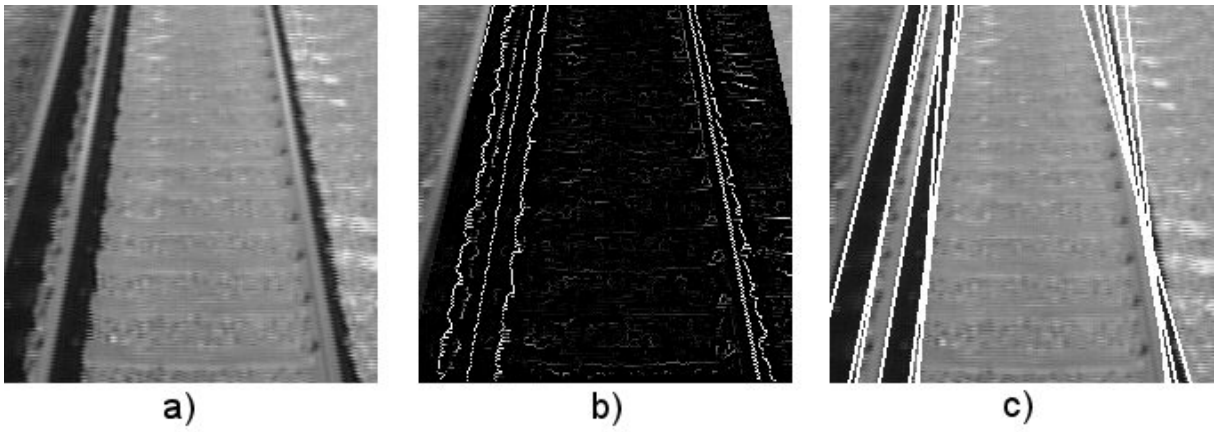

Abbildung 4.13: a) Grauwertbild, b) Kantenbild, c) mittels HT erkannte Geraden

Bei der Auswertung gibt es ein weiteres wichtiges Merkmal zu beachten: sollen z.B. Geraden in einem sehr schmalen Teilbereich ermittelt werden, so hat man das Problem, dass lange Geraden bevorzugt werden, auch wenn vielleicht nur wenige starke Pixel auf diesen liegen. Doch bereits wenige solcher Punkte lassen den Wert des jeweiligen Akku-Elements so groß werden, dass kurze aber durchgängig erhaltene Geraden nicht mehr erkannt werden können. Die Lösung dieses Problems liegt in einer Normalisierung derart, dass bei der Aufsummierung des Hough-Akkus zusätzlich die Anzahl der Pixel gespeichert wird, die in jedes Akku-Element mit eingeflossen

sind sowie die Länge der Geraden, welche mit dem Element korrespondiert. Dann ergibt sich der (normalisierte) Wert eines Akku-Elementes aus:

$$\text{Akku-Wert}_{norm} = \frac{\text{Akku-Wert} \cdot \text{Pixelanzahl}}{\text{Länge der Geraden}}$$

4.5 Bildvorverarbeitung

Die Algorithmen der Bildvorverarbeitung kommen zwar im ersten Schritt bei der Verarbeitung eines Bildes zum Einsatz, werden hier jedoch als letzte Bildverarbeitungs-Techniken vorgestellt. Zum einen weil sie dazu dienen, die zuvor beschriebenen Algorithmen der Kantendetektion sowie die Hough-Transformation zu unterstützen und deren Ergebnisse zu verbessern, zum anderen weil ihre Relevanz nicht sehr hoch ist, wie nachfolgend noch zu sehen sein wird. Vor allem da es sich bei der HT um ein robustes rausch-unempfindliches Verfahren handelt, das selbst bei relativ schlechten Kantenbildern gute Ergebnisse liefern kann, wird eine Bildvorverarbeitung meist hinfällig.

In diesem Abschnitt soll keine umfassende Beschreibung der Techniken erfolgen, die für eine Vorverarbeitung in Frage kommen. Vielmehr soll anhand von Beispielbildern gezeigt werden, bei welchen Algorithmen sich eine tatsächliche Verbesserung der Qualität eines Kantenbildes feststellen lässt, wodurch man auch auf bessere Ergebnisse der HT schließen könnte.

4.5.1 Gauß-Tiefpassfilter

Der Gauß-Tiefpassfilter ist ein linearer Filter, der das Bild glätten soll. Allgemein werden Kanten durch solch eine Operation verwischt, dieser Effekt wird jedoch durch die Verwendung einer (1 2 1)-Binomial-Filtermaske abgeschwächt, die eine diskrete Approximation der Gauß-Funktion darstellt. Solch ein Filter schwächt Rauschanteile ab und unterdrückt auch *Interlacing*, wie es bei Videobildern auftritt.

Abbildung 4.14 zeigt die Wirkung des Gauß-Filters auf den vorderen Teil eines auszuwertenden Bereichs unter Verwendung des Shen-Castan-Kantendetektors. Schön erkennbar ist die Entfernung der Auswirkungen des Interlacing, welches vor allem im unteren Bildbereich auftritt, da hier die Änderungen zwischen zwei Bildern am größten sind. Allerdings sei erwähnt, dass die Verbesserung für den Betrachter zwar offensichtlich ist, die HT dadurch allerdings nicht überwältigend stark beeinflusst wird, da es hier weitestgehend egal ist, ob eine Kante durchgehend detektiert wurde oder nur bruchstückhaft. Daher kann die Gauß-Filterung zwar in einzelnen Situationen die Erkennung verbessern, verrichtet aber keine Wunder. Je nach zur Verfügung stehender Rechenzeit muss daher abgewogen werden, ob sie zum Einsatz kommt oder nicht.

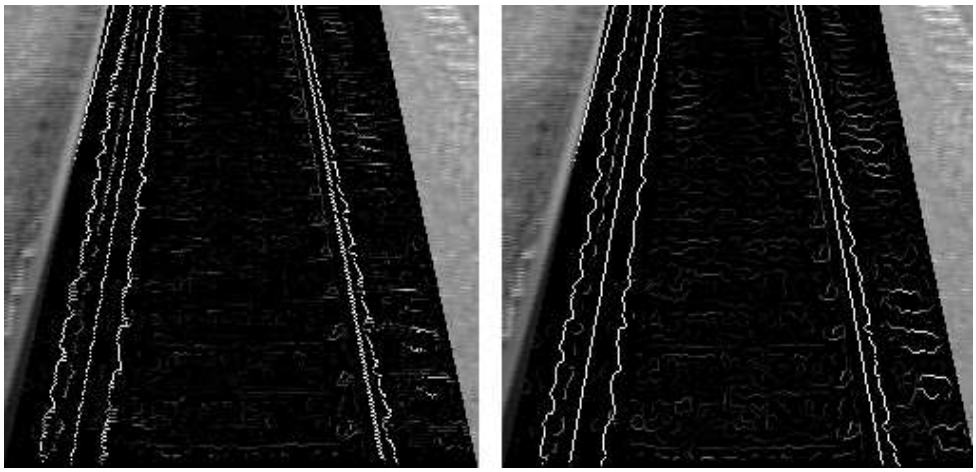

Abbildung 4.14: Kantenbild... links: ohne Gauß-Filterung, rechts: mit Gauß-Filterung

4.5.2 Highboost-Filter

Beim Highboost-Filter wird das Original-Bild gewichtet und davon ein tiefpassgefiltertes Bild subtrahiert. Dadurch werden niederfrequente Bildanteile unterdrückt und hochfrequente verstärkt, woraus eine Bildschärfung resultiert. Zu den hochfrequenten Anteilen gehören Kanten, allerdings auch Rauschen, was meist unakzeptabel ist. Von Interesse wäre ein Highboost-Filter im vorliegenden Fall nur für die Verstärkung von Kanten in oberen Teilbereichen, die wenig Rauschanteile enthalten. Abbildung 4.15 zeigt das Ergebnis solch einer Filterung.

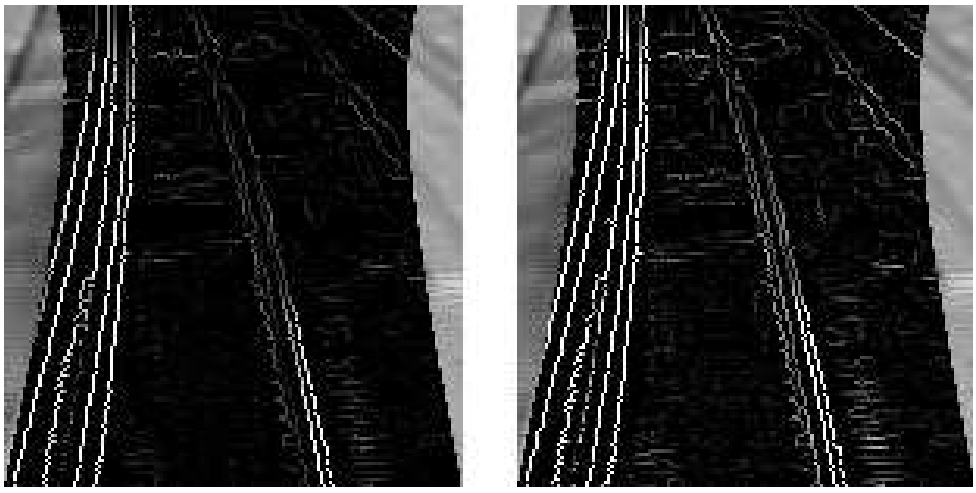

Abbildung 4.15: Kantenbild... links: ohne Highboost-Filterung, rechts: mit Highboost-Filterung

Allgemein betrachtet werden relevante Kanten zwar verstärkt, jedoch natürlich auch solche, die nicht von Interesse sind. Dies stellt einen Grund dar, der gegen den Highboost-Filter bzw. Hochpassfilter im allgemeinen spricht. Der andere Grund ist die verhältnismäßig hohe Laufzeit, da erst ein Tiefpassfilter angewendet werden muss und dann noch eine Bild-Subtraktion durchzuführen ist. Entscheidend ist jedoch, dass sich auch hier keine Verbesserung der HT-Ergebnisse ergibt. Zu begründen ist dies damit, dass die Stärke der Punkte im Kantenbild für die HT irrelevant

ist, solange ihr Verhältnis zueinander gleich bleibt. Sieht man einmal von Kanten mit maximaler Stärke ab, verändert der Highboost-Filter dieses Verhältnis kaum, wodurch dessen Einsatz hinfällig wird.

4.5.3 Histogrammausgleich

Der *(lineare) Histogrammausgleich* ist eine Punktoperation, die ein gleichverteiltes Histogramm und daraus folgend ein linear ansteigendes kumulatives Histogramm zum Ziel hat, was bei diskreten Verteilungen nur angenähert werden kann. Stellt $h(g)$ das Histogramm eines Bildes dar, dann wird der neue Grauwert g_{neu} eines jeden Pixels berechnet aus dessen altem Grauwert g_{alt} mit:

$$g_{neu} := \frac{g_{max} \cdot \sum_{g=0}^{g_{alt}} h(g)}{M \cdot N}$$

wobei g_{max} den maximalen Grauwert (i. Allg. 255) darstellt und M sowie N die Bilddimensionierung angeben. Allgemein ergibt sich für ein so bearbeitetes Bild eine Kontrastspreizung, allerdings können bei einer globalen Bearbeitung viele Details verloren gehen, wenn Grauwerte eng beeinander liegen. Im vorliegenden Fall könnte für jeden einzelnen Teilbereich ein Histogrammausgleich stattfinden. Abbildung 4.16 zeigt das Ergebnis-Kantenbild ohne und mit Verwendung des Histogrammausgleichs einzelner Teilbereiche.

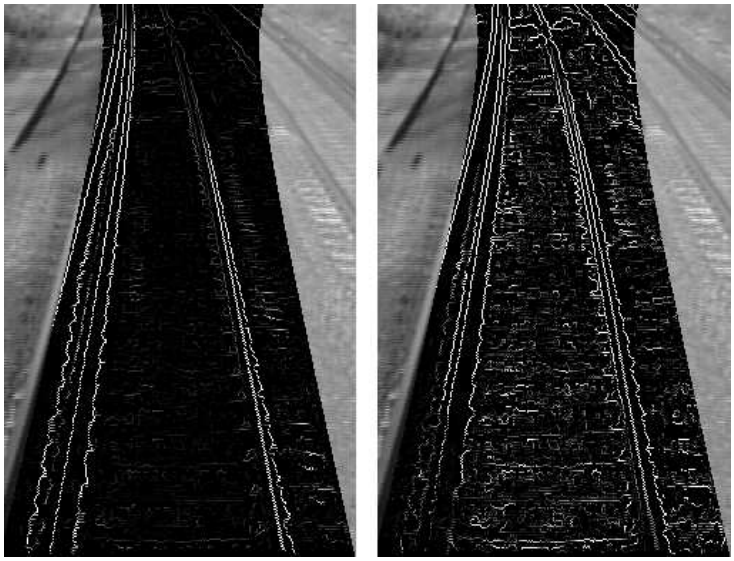

Abbildung 4.16: Kantenbild... links: ohne Histogrammausgleich, rechts: mit Histogrammausgleich

Es lässt sich feststellen, dass Schienen im Kantenbild zwar generell verstärkt werden, es in einzelnen Fällen (z.B. bei der linken Schiene im unteren Bildbereich) aber trotzdem zu einer Eliminierung wichtiger Informationen kommt. Viel kritischer ist allerdings die Eigenschaft zu sehen, dass selbst kleinste Details hervor gehoben werden und hemmend auf die Ergebnisse der HT

wirken können. Daher stellt auch ein Histogrammausgleich kein adäquates Hilfsmittel dar, durch welches die Linienerkennung untertützt werden könnte.

4.5.4 Kontrastanpassung

Als letztes Hilfsmittel sei die (automatische) Kontrastanpassung erwähnt. Hierbei wird zunächst auch das Histogramm $h(g)$ berechnet. Anschließend werden 2 Grauwerte s_u und s_o so berechnet, dass der Bereich kleiner s_u bzw. größer s_o im Histogramm jeweils eine gewisse Prozentzahl p an Bildpunkten übersteigt (z.B. $p = 0.5$). Nun werden die Grauwerte aktualisiert durch:

$$g_{neu} = \begin{cases} 0 & \text{für } g_{alt} \leq s_u \\ \frac{255 \cdot (g_{alt} - s_u)}{s_o - s_u} & \text{für } s_u < g_{alt} < s_o \\ 255 & \text{für } g_{alt} \geq s_o \end{cases}$$

Somit findet eine lineare Abbildung des Bereichs $(s_u \ldots s_o)$ auf den gesamten Grauwertbereich statt, wodurch dieser komplett ausgenutzt wird. Auch hier gilt, dass die Kontrastverbesserung für den Betrachter zwar offensichtlich ist, die HT aus den schon genannten Gründen aber nur gering beeinflusst. Die Kontrastanpassung ist jedoch allemal besser anzusehen als die Ergebnisse des Highboost-Filters oder des Histogrammausgleichs, sodass je nach gewünschter Laufzeit ihr Einsatz in Erwägung zu ziehen ist. Abbildung 4.17 zeigt die Auswirkung der Kontrastanpassung auf ein Kantenbild mit $p = 0.5$. Vor allem im oberen Bereich der rechten Schiene ist eine Verbesserung ersichtlich.

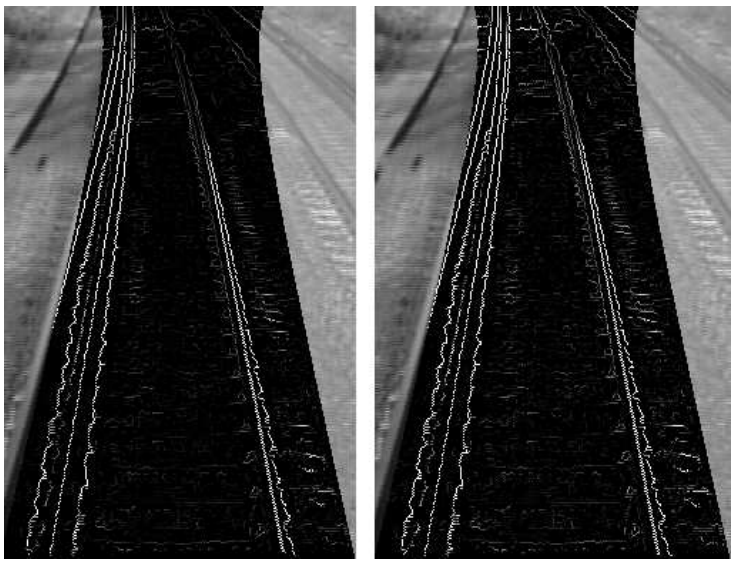

Abbildung 4.17: Kantenbild... links: ohne Kontrastanpassung, rechts: mit Kontrastanpassung

4.5.5 Fazit

Zusammenfassend lässt sich sagen, dass die meisten Operationen zur Bildvorverarbeitung zwar einen sichtbaren Verbesserungseffekt für den Betrachter haben, ihr Einfluss auf die nachfolgende Hough-Transformation allerdings gering ist. So lautet das Fazit, dass keine Vorverarbeitung den Schienenerkennungsalgorithmus umfassend verbessern kann. Wenn etwas Rechenzeit zur Verfügung steht, so empfiehlt sich am ehesten eine Gauß-Tiefpassfilterung mit anschließender Kontrastanpassung, was die Erkennung in einzelnen Fällen unterstützen kann.

4.6 Der Algorithmus im Detail

Nachdem in den vorigen Abschnitten auf die grundlegenden Bildverarbeitungs-Techniken eingegangen wurde, kann nun unter Kenntnis dieser der verwendete Algorithmus zur Schienenextraktion detaillierter betrachtet werden. Zunächst wird die mathematische Modellierung der Schiene (das verwendete Schienenmodell) sowie die Unterteilung des eingegrenzten Bildraums in horizontale Teilbereiche beschrieben. Dann wird in 4.6.3 und 4.6.4 ausführlich auf die einzelnen Arbeitsschritte des Algorithmus eingegangen, was die Grundlage dafür darstellt, in 4.6.5 schlussendlich das eigentliche Schienenpaar extrahieren zu können.

4.6.1 Mathematische Modellierung der Schiene

Der grundlegende Schienenerkennungs-Algorithmus basiert wie in 4.2 beschrieben auf der Modellierung der Schiene aus linearen Teilstücken. Allerdings wird keine reale Schiene diesem Modell gerecht. Daher ist eine erweiterte mathematische Modellierung nötig, die in diesem Abschnitt beschrieben werden soll. Mit dieser kann eine Ausgleichslösung für die tatsächliche Schiene aus den einzelnen linearen Teilstücken hergestellt werden, auch um einzelne „Ausreißer“ in den Segmenten auszugleichen. Als grundlegende Voraussetzung sei dabei der sichtbare Bereich als *flach* angenommen, d.h. es können keine Berg- und Talfahrten berücksichtigt werden, die Schiene befindet sich in der X_wY_w-Ebene. Diese Annahme ist i. Allg. nicht gültig und stellt eine Vereinfachung der Realität dar, wird jedoch für den Einsatz der Kamerakalibrierung benötigt. Der dadurch entstehende Fehler ist bei Schienensystemen als gering anzusehen, sodass diese Annahme keine Einschränkung darstellen sollte.

4.6.1.1 Parabolisches Modell

Zunächst ist es möglich, eine Schiene als allgemeine Parabel in folgender Form zu beschreiben:

$$x = a + by + cy^2$$

Dabei wird x in Abhängigkeit von y berechnet, da eine Schiene nur einen x-Wert pro y-Ordinate aufweist, hingegen jedoch mehrere y-Werte pro x-Koordinate besitzen kann.

Die grundlegende Einschränkung des Modells ist, dass sich damit nur einfache Kurven beschreiben lassen, jedoch keine komplexeren wie z.B. s-förmige Kurven. Dies wird erst durch höherwertige Funktionen und Splines ermöglicht. Hier ist die Modellierung über eine Parabel allerdings ausreichend, da Festlegungen zu maximalen Kurvenradien im Schienenverkehr und die recht kurze Entfernung des einzuzeichnenden Lichtraumprofils (siehe Abschnitt 4.7) komplexe Kurven ausschließen.

Abbildung 4.18 zeigt eine Schienenerkennung mit eingezeichneten Parabeln. Diese wurde als Ausgleichslösung des linearen Gleichungssystems ermittelt, das durch die 3 Parameter a, b und c der obigen quadratischen Funktion entsteht. Es kann mit mindestens 3 ermittelten Schienenpunkten gelöst werden.

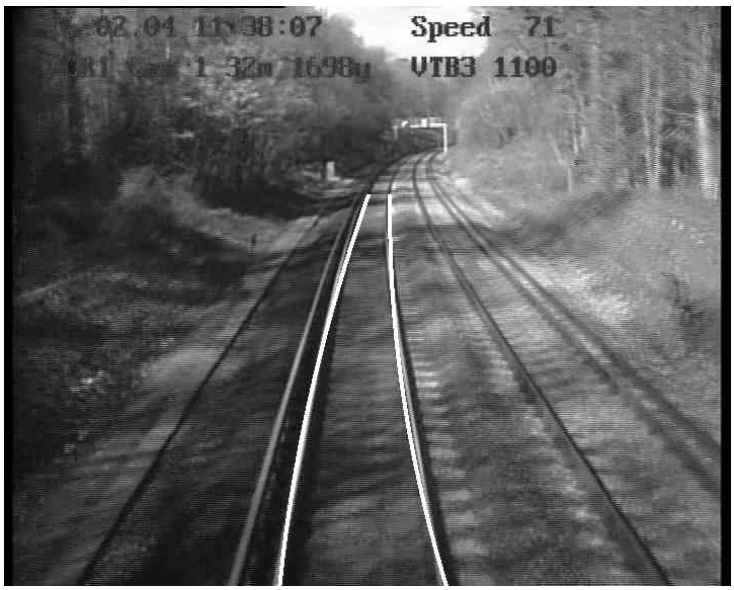

Abbildung 4.18: Parabolisches Modell

4.6.1.2 Linear-parabolisches Modell

Abbildung 4.18 zeigt ein grundlegendes Problem des parabolischen Modells: im Nahbereich am unteren Bildrand wird die vorhandene Linearität der Schiene durch die perspektivische Abbildung nicht in das Modell mit einbezogen, sodass die Ausgleichslösung zu einer unerwünschten Krümmung führen kann. Aufgrund dessen soll hier ein linear-parabolisches Modell Anwendung finden, wie es in [6] vorgeschlagen wird. Der untere Bildbereich (*Nahbereich*) wird bis zu einer Ordinate y_m als linear angenommen, darüber (*Fernbereich*) findet eine Parabel Anwendung (daher auch der Name des Modells).

Mathematisch formuliert:

$$f(y) = x = \begin{cases} a + by & \text{für } y > y_m \\ c + dy + ey^2 & \text{für } y \leq y_m \end{cases} \tag{4.3}$$

Da die Funktion stetig und differenzierbar sein soll, ergibt sich aus den daraus folgenden Bedingungen $f(y_m^-) = f(y_m^+)$ und $f'(y_m^-) = f'(y_m^+)$:

$$\begin{aligned} a + by_m &= c + dy_m + ey_m^2 \\ b &= d + 2ey_m \end{aligned}$$

Löst man diese Gleichungen nach den Variablen c und e auf, so erhält man als Modell aus 4.3:

$$f(y) = x = \begin{cases} a + by & \text{für } y > y_m \\ \frac{2a+y_m(b-d)}{2} + dy + \frac{b-d}{2y_m}y^2 & \text{für } y \leq y_m \end{cases} \tag{4.4}$$

Somit benötigt man aufgrund der erweiterten Voraussetzungen anstatt 5 nur noch 3 Parameter zur Modellbeschreibung.

Zur Berechnung der 3 Parameter bietet sich folgende Vorgehensweise an: man ermittelt eine Menge von m Schienenpunkten P_i ($m \geq 1$) im linearen Nahbereich und n Schienenpunkte P_j ($n \geq 2$) im parabolischen Fernbereich. Durch Umstellung von Gleichung 4.4 ergibt sich das folgende lineare Gleichungssystem mit 3 Unbekannten und m+n Gleichungen, welches sich mit den ermittelten Punkten lösen lässt:

$$\begin{cases} a + y_i b & \text{für } i = 1 \ldots m \\ a + \left(\frac{y_m}{2} + \frac{1}{2y_m}y^2\right) \cdot b + \left(\frac{-y_m}{2} + y - \frac{1}{2y_m}y^2\right) \cdot d & \text{für } j = 1 \ldots n \end{cases} \tag{4.5}$$

Abbildung 4.19 zeigt das linear-parabolische Modell in Aktion. Es bezieht die Linearität des vorderen Schienenbereichs mit ein und ist für die Zwecke dieser Arbeit gut geeignet. Es sollte allerdings bedacht werden, dass dieses Modell umso unpassender wird, je größer die Entfernung des einzuzeichnenden Lichtraumprofils ist. Dann reicht eine Beschreibung mittels einer Parabel nicht mehr aus und es müssen komplexere Modelle zum Einsatz kommen.

Abbildung 4.19: Linear-parabolisches Modell

4.6.2 Aufteilung in Teilbereiche

Da der in 4.2 grundlegend beschriebene Algorithmus von linearen Teilstücken ausgeht, muss zunächst initial eine horizontale Zerlegung des eingegrenzten Bildraums erfolgen. Dabei sollte darauf geachtet werden, dass Schienenstücke des jeweiligen Teilbereiches selbst im Extremfall einer maximalen Kurvenfahrt nahezu linear sind. Die hier verwendete Realisierung basiert auf einer Heuristik, welche die Teilbereiche nach oben hin kleiner werden lässt. Dadurch wird der Tatsache Rechnung getragen, dass die Schiene aufgrund der perspektivischen Abbildung im Nahbereich über eine große Spanne annähernd linear ist und im Fernbereich stärkere Krümmungen aufweisen kann.

Abbildung 4.20 zeigt die verwendete Heuristik am Beispiel der Zerlegung in $n = 4$ Teilbereiche. Zunächst wird die x-Achse im Intervall $[0 \dots x_{max}]$ in 4 gleichgroße Segmente unterteilt. Zu jedem so entstandenen x_i wird dann der Wert w_i folgender Funktion berechnet:

$$f(x_i) = w_i = \frac{1}{e^{\frac{x_{max}}{n} \cdot x_i}}$$

Der Bereich w_0 bis w_n entspricht dabei dem zu unterteilenden y-Bereich y_{unten} bis y_{oben}, sodass die Spanne y_i eines Teilbereichs berechnet werden kann mit:

$$y_i = \frac{(w_{i-1} - w_i) \cdot (y_{unten} - y_{oben})}{1 - w_n} \qquad \text{für } i \in [1 \dots n]$$

Abbildung 4.21 zeigt die so durchgeführte Zerlegung des eingegrenzten Bildraums in 4 Teilbereiche. Es sei angemerkt, dass die Größe der einzelnen Teilbereiche mit x_{max} variiert. Niedrige

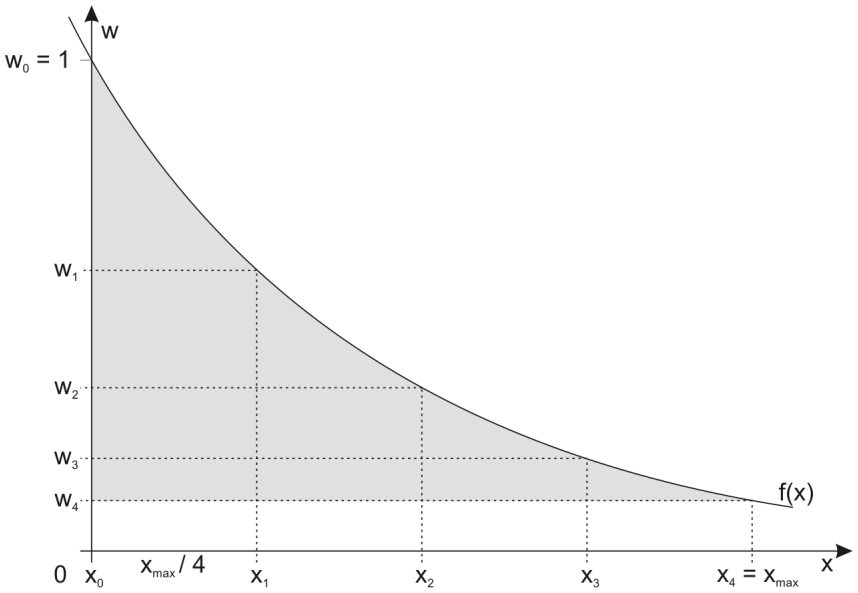

Abbildung 4.20: Heuristik zur Teilbereichs-Einteilung

x_{max}-Werte führen zu einer relativ gleichmäßigen Verteilung der einzelnen Segmente, während große Werte den vorderen Gebieten einen höheren Betrag zukommen lassen. Durch Tests konnte ein Wert von $x_{max} = 3$ als in vielen Fällen geeignet ermittelt werden. Der zweite zu beachtende Parameter ist die Anzahl der Teilbereiche n. Ist n zu groß, so sind stärkere Fehlerkennungen aufgrund sehr kleiner Segmente möglich. Wird n wiederum zu klein gewählt, so können Schienenstücke der einzelnen Teilgebiete stark von der Linearitätsannahme abweichen.

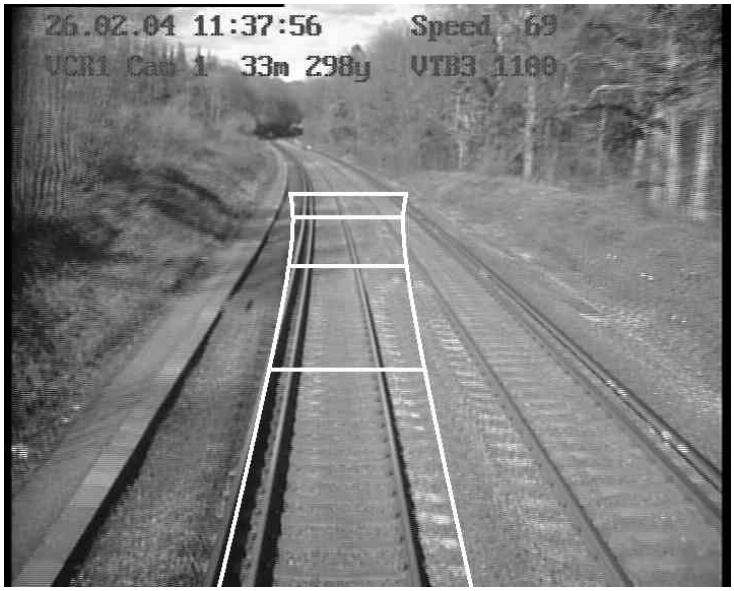

Abbildung 4.21: Beispiel einer Unterteilung in 4 Teilbereiche

4.6.3 Ermitteln möglicher Schienenstücke

Nun sollen die beiden Hauptschritte des in 4.2 beschriebenen Schienenerkennungs-Algorithmus vorgestellt werden. Der erste Schritt besteht in der Extraktion möglicher Schienenstücke mit Hilfe der Hough-Transformation (HT), die in 4.4 beschrieben wurde. Dabei ist zunächst für jeden Teilbereich eine Bildvorverarbeitung und als wichtigster Bestandteil eine Kantendetektion durchzuführen, auf der die Hough-Transformation aufbaut (siehe hierzu die Abschnitte 4.3 und 4.5).

Bei der Durchführung der HT gilt es zwei Fälle zu unterscheiden:

1. Es sollen Linien als Kandidaten für Schienenstücke aus dem *untersten Bildsegment* extrahiert werden. Unter der Annahme, dass Schienenstücke die auffälligsten Geraden im Kantenbild sind (deren Gradientenbetrag am größten ist), wird die HT wie in 4.4 dargestellt mit einem festen Schwellwert ausgeführt und die sich daraus ergebenden Linien extrahiert. Tests haben ergeben, dass ein Schwellwert von 0.6 des maximalen Wertes im Hough-Akkumulator-Array gute Ergebnisse liefert.

2. Die Linienerkennung in den *anderen Bildsegmenten* kann optimiert durchgeführt werden. Dabei dienen die n Schienenkandidaten $cand_i$ des jeweils unteren Bereiches als *Saat* für die HT. Das Akkumulator-Array wird wie gehabt aufsummiert, anschließend allerdings keine globale Extraktion mittels Non-Maximum-Suppression bzw. Schwellwert-Operation durchgeführt, sondern folgende Strategie angewandt: nimm den obersten Punkt P_i eines jeden $cand_i$ und nutze ihn als Ausgangspunkt für eine Linie. D.h. dass alle Linien durch P_i führen müssen. P_i korrespondiert im Parameterraum der HT mit einer Sinoide und so ist es nur notwendig, diese Sinoide zu durchlaufen und eine Anzahl x von Maxima des Akkumulator-Arrays zu extrahieren. Diese bestimmen die von P_i ausgehenden stärksten Linien des Teilbereiches.

   Abbildung 4.22 verdeutlicht das Prinzip anhand eines Kandidaten $cand_i$ mit Startpunkt P_i. Die zugehörige Sinoide wird im (diskreten) Houghraum punktweise durchlaufen, hierbei erhält man die beiden Maxima g_1 und g_2, die im Bildraum 2 Geraden definieren und die Startpunkte P_1 und P_2 besitzen.

Durch Filterung der Gradientenrichtung anhand des eingegrenzten Winkelbereichs (siehe 3.2) kann das Ergebnis der Linienerkennung weiter verbessert werden. Zudem können in einer Nachbearbeitungsphase Linien, deren Endpunkte außerhalb des jeweiligen Teilbereiches liegen, als unzulässig ausgeschlossen werden. Dies beschränkt die Anzahl tatsächlicher Kandidaten und ist unverzichtbar für den nachfolgenden Arbeitsschritt.

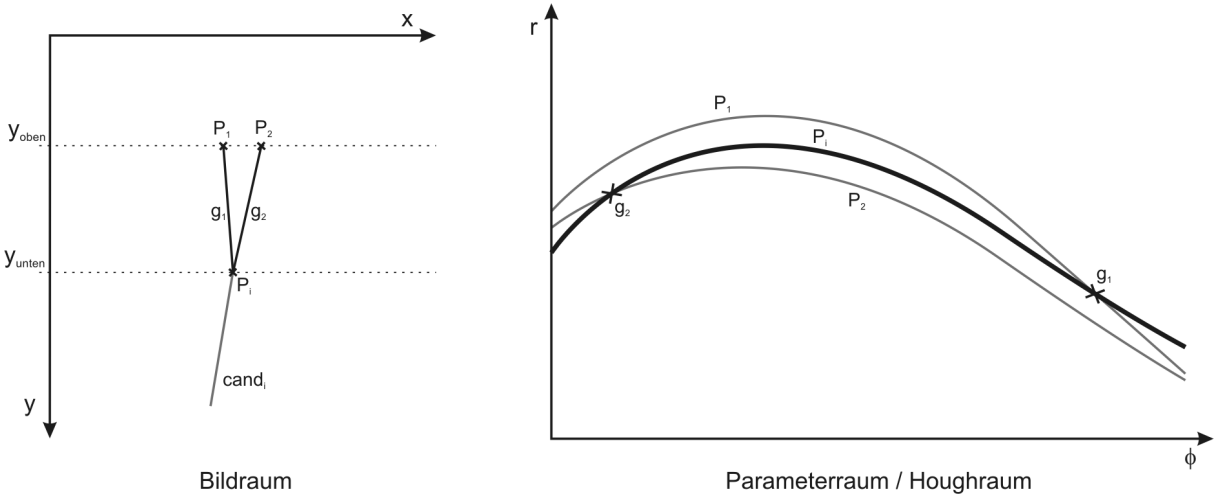

Abbildung 4.22: HT für obere Teilbereiche

4.6.4 Ermitteln tatsächlicher Schienenkandidaten

Durch die Linienerkennung erhält man mehrere Kandidaten für tatsächliche Schienen. Im nächsten Schritt gilt es, daraus die besten nach verschiedenen Kriterien auszuwählen und dadurch Kandidatenpaare für linke und rechte Schienenstücke zu erhalten. Wie auch zuvor ist hier wieder eine Unterscheidung in der Bearbeitung des untersten Teilbereichs und der anderen Segmente notwendig.

4.6.4.1 Unterster Teilbereich

Für eine Kandidatenermittlung in diesem Segment sind zwei grundsätzliche Herangehensweisen denkbar:

1. Zunächst wird zu jeder erkannten Linie l_1 über die Kamerakalibrierung die Linie s_{Ideal} erzeugt, welche sich im Abstand der Spurweite von l_1 befindet. Dann wird eine andere erkannte Linie l_2 ermittelt, die sich im minimalen Abstand von s_{Ideal} befindet, wobei dieser Abstand einen maximalen Wert d_{max} nicht überschreiten darf. l_1 und l_2 stellen dann einen Kandidaten für ein Schienenpaar dar. Algorithmus 4.3 verdeutlicht dieses Prinzip.

Algorithmus 4.3 Schienenpaare in unterstem Teilbereich ermitteln, Variante 1

```
1: function ErhalteSchienenpaareUnten(ErkannteLinien)
2:     for all Linien l_1 in ErkannteLinien do
3:         s_Ideal ← IdealerPartner(l_1)          ▷ Linie zu l_1 im Spurweitenabstand
4:         l_2 ← Linie aus ErkannteLinien im minimalen Abstand zu s_Ideal
5:         d ← Abstand(l_2, s_Ideal)
6:         if d ≤ d_max then
7:             Schienenpaare.Add(l_1, l_2)        ▷ zu Liste der Kandidatenpaare hinzufügen
8:     BestePaare ← ErhalteBestePaareUnten(Schienenpaare)
9:     return BestePaare
```

Wurde für jede Linie ein Partner gefunden und somit die möglichen Kandidatenpaare ermittelt, so werden im Anschluss daran die besten Paare nach verschiedenen Kriterien extrahiert. In Algorithmus 4.3 wird dies durch die Funktion ErhalteBestePaareUnten erledigt. Dabei wird zu jedem Kandidatenpaar ein Wert berechnet, der umso größer ist, je stärker angenommen werden kann, dass es sich bei diesem Kandidaten um das gesuchte Schienenpaar handelt. Die Kriterien 1, 2 und 3 der Auflistung unter 4.6.4.3 dienen dabei zur Berechnung dieses Wertes, wodurch der Betrag der Linien im Hough-Akku, der Abstand vom Ideal und der Abstand vom Voraussagewert mit in die Entscheidungsfindung einbezogen werden.

2. Wie in Variante 1 wird auch hier zu jeder erkannten Linie l_1 die Linie s_{Ideal} im Abstand der Spurweite erzeugt, allerdings wird nicht nach einer Geraden l_2 im minimalen Abstand zu s_{Ideal} gesucht, sondern s_{Ideal} selbst wird als der Schienenpartner von l_1 verwendet. Somit wird Kriterium 1 von Abschnitt 4.6.4.3 hinfällig, zur Ermittlung der besten Kandidatenpaare werden nur noch die Kriterien 2 und 3 (Wert des Hough-Akkus und Entfernung vom Voraussagewert) verwendet.

Die Notwendigkeit nach einer Unterteilung in 2 Varianten ist zunächst nicht offensichtlich. Sie ergibt sich erst aus der näheren Betrachtung beider Varianten. Variante 1 fußt auf der Annahme, dass es sich bei Schienen um die bedeutendsten Merkmale in einem Bild handelt, die auch primär im entsprechenden Kantenbild hervortreten. Dies kann allerdings nicht verallgemeinert behauptet werden. Abbildung 4.23 zeigt ein Beispiel, wo die linke Schiene aufgrund schlechter Grauwertübergänge im Kantenbild kaum in Erscheinung tritt und der Algorithmus nach Variante 1 nicht funktioniert.

Hier greift allerdings Variante 2, welche den idealen Partner zu den Kandidaten der rechten Schiene erzeugt und somit selbst in dieser prekären Situation ein verlässliches Ergebnis liefert. Warum also nicht immer Variante 2 verwenden? Auch hier ergeben sich Nachteile. Wenn durch die HT

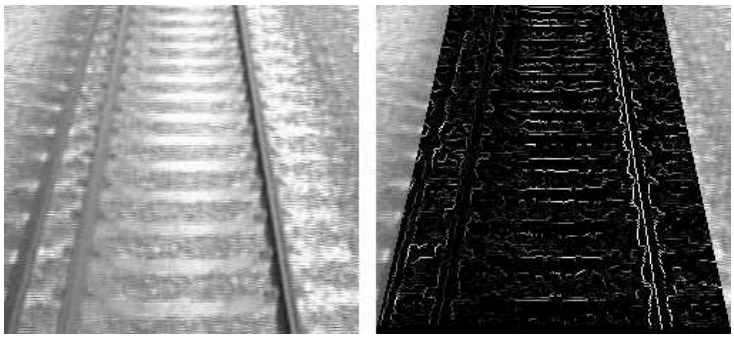

Abbildung 4.23: Problematische Situation für 4.6.4.1, Variante 1

tatsächlich linke und rechte Schienenkandidaten gefunden wurden, so kann es mit Variante 2 dazu kommen, dass ein Kandidatenpaar gewinnt, bei dem die im idealen Abstand erzeugte Schiene weit entfernt ist von einer mittels HT erkannten Linie. Abbildung 4.24 verdeutlicht diesen Sachverhalt.

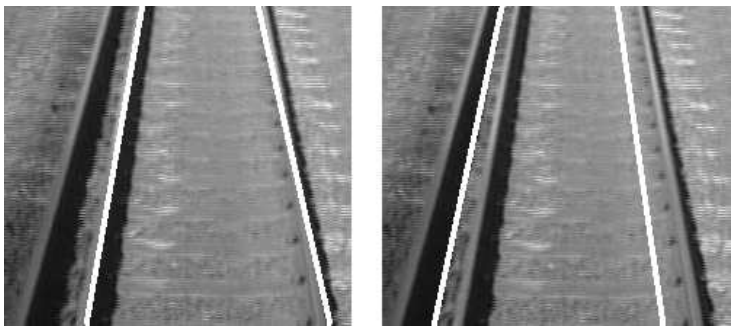

Abbildung 4.24: 4.6.4.1, Variante 1 (links) funktioniert gut, Variante 2 (rechts) nicht

Allgemein lässt sich festhalten, dass Variante 1 in klaren Kantensituationen zu überzeugen vermag, während Variante 2 dann gute Ergebnisse liefert, wenn nur eine Schienenseite erkannt wurde. Dieser Umstand wird in dem verwendeten Algorithmus berücksichtigt: wenn Variante 1 aufgrund schlechter Linien zu keinen Ergebnissen kommt, wird Variante 2 verwendet. Dies führt zu einer Erkennung, die auch in schwierigen Situationen gut funktionieren kann (siehe Abschnitt 4.8).

4.6.4.2 Andere Teilbereiche

Ausgangspunkt für eine Schienen-Extraktion in einem der oberen Teilbereiche bilden die Gewinnerpaare $gewinner_u$ des direkt darunter befindlichen Bereichs. Das bedeutet auch, dass man durch die Linienermittlung in 4.6.3 nun für jedes Element aus $gewinner_u$ eine Menge von Geraden erhält. Dies stellt den größten Unterschied zu 4.6.4.1 dar, da man so eine Unterteilung in linke und rechte Schienenkandidaten besitzt und z.B. für linke Schienen nicht erst nach möglichen Partnerschienen gesucht werden muss.

Auch hier sind zwei Vorgehensweisen bei der Extraktion der besten Kandidaten zu unterscheiden, die den beiden Varianten in Abschnitt 4.6.4.1 sehr ähnlich sind:

1. Da zu jedem Paar aus $gewinner_u$ Kandidaten für linke und rechte Schienen ermittelt wurden, wird jedes Element w_i aus $gewinner_u$ entnommen und zu jedem linken Schienenkandidaten l_{ij} alle rechten Schienenkandidaten r_{ik} betrachtet. Für jedes dieser Kandidatenpaare $cand_{ijk} = (l_{ij}, r_{ik})$ wird nach allen 4 Kriterien von 4.6.4.3 die gewichtete Summe gebildet und n Kandidaten mit dem höchsten Wert extrahiert. Dabei darf der Abstand d zwischen jeweiliger rechter Schiene r_{ik} und der Schiene s_{Ideal} im idealen Abstand zu l_{ij} einen Grenzwert d_{max} nicht überschreiten. Die n Kandidaten mit höchstem Wert sind dann die Paare, welche nach den verwendeten Kriterien einem Schienenpaar am ähnlichsten sind. Algorithmus 4.4 verdeutlicht dieses Konzept.

Algorithmus 4.4 Schienenpaare in oberen Teilbereichen ermitteln, Variante 1

```
1:  function ErhalteSchienenpaareOben(gewinner_u)
2:                                  ▷ gewinner_u = beste Paare aus nächstunterem Bereich
3:      for all Schienenpaare w_i in gewinner_u do
4:          for all Schienen l_ij in w_i do
5:              s_Ideal ← IdealerPartner(l_ij)        ▷ Linien zu l_ij im Spurweitenabstand
6:              for all Schienen r_ik in w_i do
7:                  d ← Abstand(r_ik, s_Ideal)
8:                  if d ≤ d_max then
9:                      Kandidatenpaare.Add(l_ij, r_ik)        ▷ zu Paarliste hinzufügen
10:     BestePaare ← ErhalteBestePaareOben(Kandidatenpaare)
11:     return BestePaare
```

2. Die zweite Variante durchläuft ebenfalls alle Elemente w_i von $gewinner_u$, betrachtet nun aber die linken und rechten Schienen l_{ij} und r_{ik} sequentiell. Zu jeder dieser Schienen wird nun wie gehabt die Schiene s_{Ideal} im Spurweitenabstand erzeugt und direkt als Partner der jeweiligen Schiene verwendet. Dann werden über die Kriterien 2, 3 und 4 (1 ist hinfällig) aus 4.6.4.3 die gewichteten Summen dieser Schienenpaare gebildet und wieder die n stärksten Kandidaten extrahiert.

Auch hier stellt sich die Frage nach den Vor- und Nachteilen beider Varianten und dem Sinn ihres Einsatzes. Variante 1 bezieht nur die tatsächlich durch die HT erkannten Linien mit ein und sucht die besten Paare solcher Linien. Dadurch ist gewährleistet, dass Schienenpaare auch bei kleineren Kalibrierungsfehlern gefunden werden. Allerdings kann es passieren, dass erkannte Schienenpaare in Weltkoordinaten nicht idealerweise parallel sind. Variante 2 generiert zu jeder Schiene die Partnerschiene im Abstand der Spurweite. Dadurch ist die Parallelität der Schienen

im Weltkoordinatensystem gewährleistet. Jedoch werden die durch die HT erkannten Paare nicht mit einbezogen, was die Möglichkeit verschenkt, kleine Fehler auszugleichen.

Allgemein gesehen ist Variante 2 zu statisch und bezieht die tatsächliche Erkennung nur halbherzig mit ein, was in einer schlechteren Verfolgung des tatsächlichen Fahrweges resultieren kann. Abbildung 4.25 zeigt ein Beispiel für solch eine Fehlerkennung. Variante 1 hat hier Vorteile, allerdings auf Kosten der idealen Parallelität im Weltkoordinatensystem. So können leichter Ungenauigkeiten in der Erkennung auftreten, wie Abbildung 4.26 verdeutlicht.

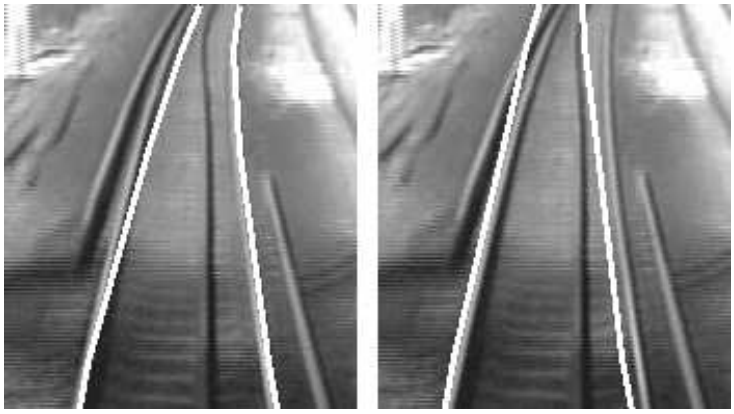

Abbildung 4.25: 4.6.4.2, Variante 1 (links) funktioniert gut, Variante 2 (rechts) nicht

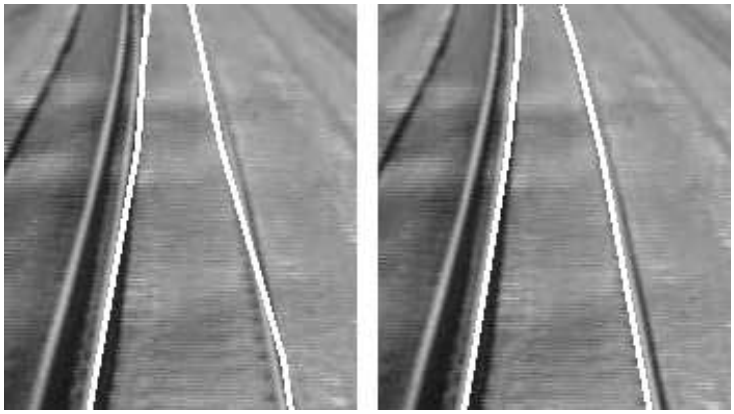

Abbildung 4.26: 4.6.4.2, Variante 2 (rechts) funktioniert gut, Variante 1 (links) nicht

Insgesamt ist die tatsächliche Verfolgung der Schiene bei Variante 1 besser, teilweise aber auf Kosten der Genauigkeit. Durch die Ermittlung der Spurparabeln können solche Ungenauigkeiten allerdings häufig wieder ausgeglichen werden. So kommt primär Variante 1 zum Einsatz. Sekundär wird Variante 2 verwendet, wenn durch Variante 1 keine Schienenpaare erkannt werden konnten.

4.6.4.3 Kriterien für die Ermittlung bester Kandidatenpaare

In dieser Arbeit wurden 4 Kriterien verwendet, die dazu benutzt werden, um aus einer Menge von Schienenkandidaten die besten zu ermitteln (4.6.4.1 und 4.6.4.2 beschreiben ihren Einsatz). Dabei

entscheidet eine *gewichtete Summe* dieser Kriterien über den Wert, welchen ein Schienenkandidat erreichen kann. Die Gewichtung der einzelnen Kriterien ist nicht-trivial und muss sorgfältig gewählt werden. Ebenso ist auf eine Normierung zu achten, sowohl jedes Einzelkriterium als auch der Gesamtwert eines Kandidaten sollte normiert sein. Im Zuge dieser Arbeit findet dabei eine *1-Normierung* statt. Die Normierung der einzelnen Kriterien ist unten aufgelistet, zur 1-Normierung des Gesamtwertes erfolgt eine Division durch die Summe der Gewichtsfaktoren der verwendeten Kriterien.

1. **Abstand zur idealen Partner-Schiene** s_{Ideal}:
   - *Hintergrund*: Wurde zu einer Schiene l_1 eine zweite Schiene l_2 mit Abstand d zu s_{Ideal} gefunden, so kann d als Parameter der Berechnung mit einfließen.
   - *Idealfall*: $d = 0$ ist der Idealfall, auf den sich allerdings nicht verlassen werden darf. Durch Kalibrierungsfehler bzw. Ungenauigkeiten in der Linienextraktion kann die tatsächliche Schiene leicht vom Ideal abweichen.
   - *Maximalwert*: Kann im Extremfall den vorher festgelegten Wert d_{max} erreichen.
   - *Berechnung*: Über die Gauß-Funktion mit einem Wert für σ von $\frac{d_{max}}{2}$.
   - *Normierung*: Durch die Gauß-Funktion 1-normiert.
   - *Gewichtungsfaktor*: Sollte relativ hoch sein, z.B. 3-4.
2. **Wert der beiden Linien des Kandidatenpaares im Hough-Akkumulator**:
   - *Hintergrund*: Die Linienkandidaten wurden zwar schon durch die Extraktion maximaler Werte im Hough-Akku ermittelt, allerdings kann immernoch angenommen werden, dass höhere Akku-Werte mit den besten Linienkandidaten korrelieren.
   - *Idealfall*: Der Wert kann nicht größer als der maximale Wert im Hough-Akku werden. Nur aufgrund der höchsten Akku-Werte auf die besten Kandidaten zu schließen ist jedoch auch nicht korrekt, schließlich ist damit z.B. noch nicht gesagt, in welchem Abstand sich die beiden Linien zueinander befinden.
   - *Maximalwert*: Im Extremfall $2 *$ maximaler Akku-Wert.
   - *Berechnung*: Durch Summierung der Werte der zwei Akkumulator-Zellen, welche die beiden Linien repräsentieren.
   - *Normierung*: Eine Division durch den zweifachen *Maximalwert* (siehe oben) führt zu einer 1-Normierung.
   - *Gewichtungsfaktor*: Hoher Einfluss, z.B. 4-5.

3. **Abweichung der obersten Punkte von vorausgesagten Werten**:

   - *Hintergrund*: Für jeden zu untersuchenden Teilbereich lässt sich über die Kamerakalibrierung ein Geradenpaar (l_{Gerade}, r_{Gerade}) ermitteln, welches die Schienen bei Geradeausfahrt darstellt. Sei *cand* ein Schienenkandidat mit einem Schienenpaar (l, r). Betrachtet sei zunächst nur die Voraussage für die linke Schiene l. Für den Schnittpunkt $P(x_P, y_P)$ von l mit der unteren Teilbereichsgrenze ermittelt man den Abstand d_P von P zum linken Teilbereichsrand und den Abstand d_l von l_{Gerade} zum linken Teilbereichsrand. Das Verhältnis d_l/d_P wird berechnet und auf den Abstand des linken Randes der oberen Teilbereichsgrenze zu l_{Gerade} übertragen. Dies ergibt als lineare Interpolation eine Voraussage, wo der oberste Punkt von l anzutreffen sein sollte. Der Abstand d_{Voraus} von tatsächlichem zu vorhergesagtem oberen Punkt ist ein Maß für das Zutreffen dieser Voraussage auf l. Äquivalent verfährt man mit der rechten Schiene r von *cand*.
   - *Idealfall*: $d_{Voraus} = 0$ ist möglich, allerdings nicht immer als ideal anzusehen. In unteren Teilbereichen wird die Vorhersage stärker zutreffen aufgrund der Linearität hier vorhandener Schienen. In oberen Teilbereichen kann es leichter zu Abweichungen kommen, da nichts über den tatsächlichen Kurvenverlauf bekannt ist und eine lineare Interpolation zur maximalen Kurvenfahrt hier nur eine grobe Schätzung darstellen kann.
   - *Maximalwert*: Für die linke Schiene l und ein gegebenes y: wenn x_{min} die zugehörige Koordinate bei maximaler Linkskurvenfahrt und x_{min1} die Koordinate bei maximaler Rechtskurvenfahrt auf der linken Schiene definiert, dann ist der Maximalbetrag $d_{VorausMax}$ der Abweichung zwischen vorausgesagtem und tatsächlichem Wert $d_{VorausMax} \approx (x_{min1} - x_{min})/2$.
   - *Berechnung*: Unter Verwendung der Gauß-Funktion auf den Wert von $d_{Voraus}/d_{VorausMax}$, der maximal 1 sein kann. Dabei sollte σ für untere Teilbereiche einen kleinen Wert besitzen (z.B. $\sigma = 0.1$), für obere Teilbereiche allerdings relativ groß sein (z.B. $\sigma = 0.8$), um die Wahrscheinlichkeit einer ungenauen Vorhersage in diesen Bereichen mit einzubeziehen.
   - *Normierung*: 1-Normierung durch die Gauß-Funktion.
   - *Gewichtungsfaktor*: In unteren Teilbereichen hoher Faktor, z.B. 4-5. In oberen Teilbereichen geringerer Wert, z.B. 2-3.

4. **Wert des Schienenkandidaten aus dem vorhergehenden Teilbereich** (für obere Teilbereiche):

   - *Hintergrund*: Außer für den untersten Teilbereich gibt es durch die sequentielle Abarbeitung von unten nach oben für jedes Schienenkandidaten-Paar $cand_i$ eines Bereichs ein Vorgängerpaar $cand_{iVorg}$ im nächstunteren Gebiet, welches den Ausgangspunkt

für $cand_i$ darstellt. Auch $cand_{iVorg}$ wurde nach den hier aufgeführten Kriterien ein Wert im Bereich $[0 \dots 1]$ zugewiesen und dieser kann und sollte Einfluss nehmen auf den Wert von $cand_i$.

- *Normierung*: Bereits normiert durch Gesamtnormierung des Wertes von $cand_{iVorg}$.
- *Gewichtungsfaktor*: Der Einfluss sollte spürbar sein, allerdings ist er schon deswegen vorhanden, weil es sich bei $cand_i$ um einen Nachfolger von $cand_{iVorg}$ handelt. Eine geringe bis mittlere Gewichtung genügt, z.B. 2-3.

4.6.5 Ermitteln des Gewinner-Kandidatenpaars

Durch Algorithmus 4.1 erhält man für jeden Teilbereich eine Menge von Schienenpaaren, die Kandidaten für die tatsächliche Schiene sein können. Zur Einzeichnung des Lichtraumprofils muss nun aus jedem Teilbereich ein Kandidat ausgewählt werden, sodass eine kontinuierliche Kette von Kandidaten entsteht, welche das jeweilige Schienenpaar darstellt.

Der Ansatz zur Ermittlung dieser Kandidatenkette lautet wie folgt: es wird das Kandidatenpaar mit dem höchsten Wert im obersten Teilbereich bestimmt. Dieses stellt das oberste Stück der Kandidatenkette dar. Nun wird dessen Vorgänger-Paar im nächstunteren Teilbereich ermittelt und dieser Vorgang bis zum untersten Teilbereich wiederholt. So wird die Kandidatenkette vom obersten Teilbereich nach unten hin aufgebaut, indem stets das jeweilige Vorgänger-Paar hinzugefügt wird. Dadurch entsteht ein Paar von Linienzügen, welche die Erkennung für die linke und rechte Schiene darstellen. Anwendbar ist diese Technik, da der Wert eines Kandidatenpaars in den Wert der darauf aufbauenden Kandidaten des darüber liegenden Teilbereichs mit einfließt.

4.7 Einzeichnen des Lichtraumprofils

Wurde eine stückweise lineare Schiene und deren mathematische Modellierung nach 4.6.1 ermittelt, so kann das Einzeichnen des Lichtraumprofils stattfinden. Dieser Schritt ist nicht sonderlich schwer im Vergleich zur voraus gelagerten Schienenerkennung, allerdings gibt es ein paar Dinge zu beachten. So muss zunächst geklärt werden, in Bezug auf welche Schiene die Maskendistanz gelten soll. Es ist offensichtlich, dass z.B. bei starker Linkskurvenfahrt die linke Schiene kürzer ist als die rechte. Als „unabhängige Dritte“ soll die Mitte zwischen den beiden Schienenparabeln als Bezugskurve für das Abtragen der Distanz gelten. Weiterhin ist das Lichtraumprofil maßstäblich und korrekt gedreht einzuzeichnen, was die Umrechnung von Welt- in Bildkoordinaten und die Ermittlung des Drehwinkels erfordert. Ein im Bahnverkehr übliches Lichtraumprofil zeigt Abbildung 4.27, welches in Anlehnung an [1] erstellt wurde.

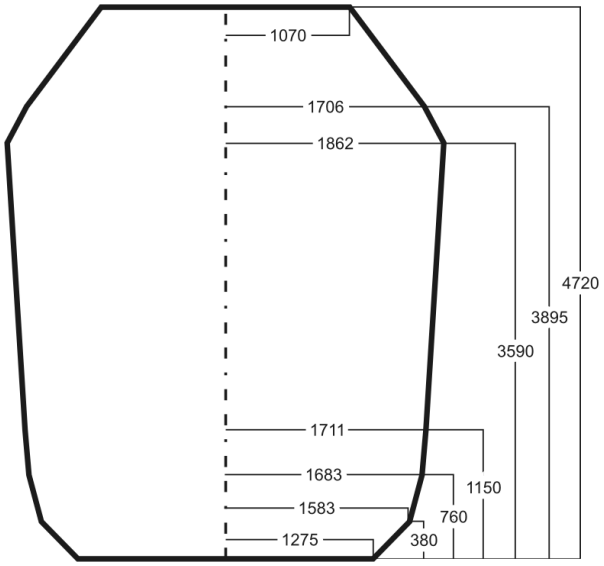

Abbildung 4.27: Beispiel für ein Lichtraumprofil nach [1], Bemaßung in mm

Zunächst ist die Distanz zur Einzeichnung des Lichtraumprofils auf der Mittelkurve zwischen den beiden Spurparabeln abzutragen. Dazu werden Ordinaten Y_{wi} des Weltkoordinatensystems im Abstand Y_{step} erzeugt und die zugehörigen Bild-Ordinaten y_i errechnet. Im Bild berechnet man dann die Schnittpunkte S_i der y_i mit der Mittelkurve, transformiert diese in Weltkoordinaten und trägt die Maskendistanz aus dem Abstand zwischen jeweils 2 benachbarten Punkten ab. Da dies i. Allg. nicht genau auf einem Weltpunkt S_{wi} endet, ist für das letzte Teilstück eine lineare Interpolation erforderlich, die zum endgültigen Maskenpunkt M_w in der Welt und dessen Entsprechung M im Bild führt. Abbildung 4.28 verdeutlicht diese Überlegungen, Algorithmus 4.5 fasst die Vorgehensweise zusammen.

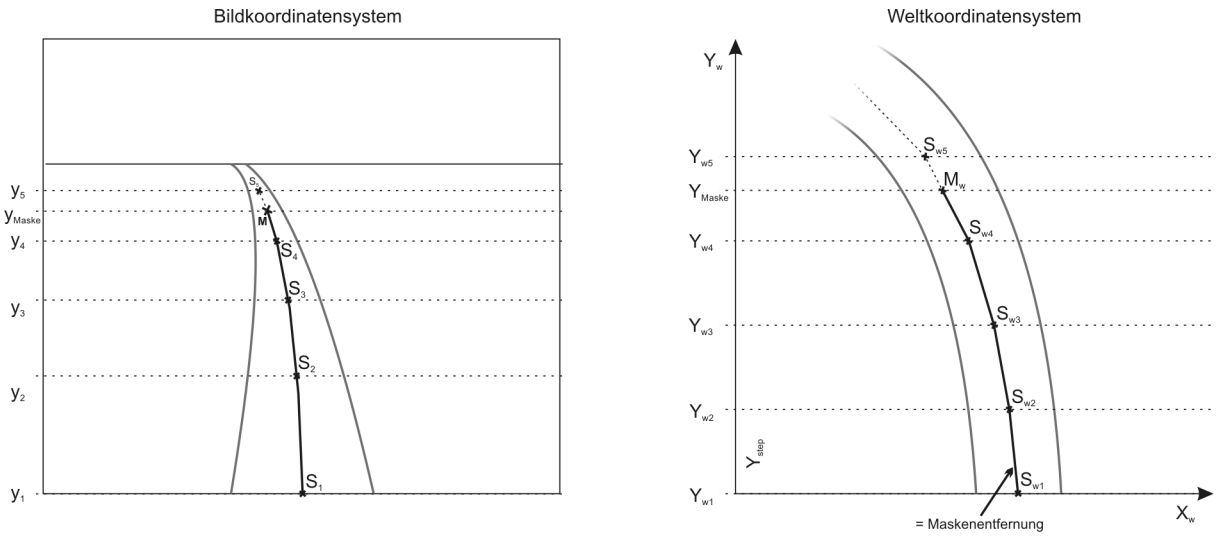

Abbildung 4.28: Abtragen der Entfernung des Lichtraumprofils

Algorithmus 4.5 Algorithmus zum Abtragen der Entfernung

function ENTFERUNGSABTRAGUNG
  $Y_{step} \leftarrow$ Schrittweite über Y_w
  $S_1 \leftarrow$ PARABELPUNKT(unterster Bildrand)
  $S_{w0} \leftarrow (\frac{Spurweite}{2}, 0, 0)$ ▷ Fußpunkt der Mittelschiene auf X_w-Achse
  $S_{w1} \leftarrow$ WELTKOORDINATEN(S_1) ▷ 1. Punkt = unterster Bildpunkt
  Distanz $\leftarrow$ ENTFERNUNG(S_{w0}, S_{w1}) ▷ Entfernung zum Fußpunkt
  i $\leftarrow 1$
  while Distanz $<$ Maskendistanz **do**
    i++
    $Y_{wi} \leftarrow (S_{wi-1}).Y + Y_{step}$ ▷ Weltordinate um Y_{step} inkrementieren
    $y_i \leftarrow$ BILDKOORDINATEN($\frac{Spurweite}{2}, Y_{wi}, 0$).Y
    $x_i \leftarrow$ PARABELKOORDINATE(y_i) ▷ Punkt auf Mittel-Parabel bestimmen
    $S_{wi} \leftarrow$ WELTKOORDINATEN(x_i, y_i) ▷ Schnittpunkt in Weltkoordinaten
    Distanz $\leftarrow$ Distanz + ENTFERNUNG(S_{wi-1}, S_{wi}) ▷ Distanz aktualisieren
  $M_w \leftarrow$ LINEAREINTERPOLATION(Distanz, Maskendistanz, S_{wi-1}, S_{wi})
  return M_w

Die Messung der Entfernung im Weltkoordinatensystem entspricht also wieder einer Rechnung über lineare Teilstücke, die Genauigkeit variiert dabei mit der Größe eines Teilstücks Y_{step}. Allerdings ist der dadurch verursachte Fehler als gering anzusehen, allein schon aufgrund der Tatsache, dass die Messung auf der Kurve im Bild in Subpixel-Genauigkeit durchgeführt wird, das Lichtraumprofil aber nur mit einfacher Pixel-Genauigkeit eingezeichnet werden kann.

Die Ermittlung des Drehwinkels α in der Welt erfolgt über die Tangentenermittlung am Maskenfußpunkt M im Bild. Diese lässt sich durch die erste Ableitung der Parabel an M bestimmen. Nach einer Transformation der Tangente in Weltkoordinaten erhält man augenblicklich den Rotationswinkel α, wie Abbildung 4.29 illustriert.

Das eigentliche Einzeichnen des Lichtraumprofils gestaltet sich nun als triviale Aufgabe. Algorithmus 4.6 verdeutlicht die Vorgehensweise unter der Voraussetzung, dass das Lichtraumprofil in der Einheit des Weltkoordinatensystems als Liste von aufeinander folgenden Punkten vorliegt. Die Rotation erfolgt dabei im Weltkoordinatensystem um die Z_w-Achse entgegen dem Uhrzeigersinn. Zwei Beispiele für die Einzeichnung des Lichtraumprofils zeigt schließlich Abbildung 4.30.

Es bleibt noch auszuwerten, welche Entfernungen sich für die Einzeichnung des Lichtraumprofils als günstig erweisen. Allgemein gilt, dass die Schienenerkennung umso schwieriger wird, je größer die Entfernung der Lichtraumprofils ist. Aufgrund der Unterabtastung und damit verschwommener Bildteile in großen Entfernungen können keine zuverlässigen Ergebnisse gefunden

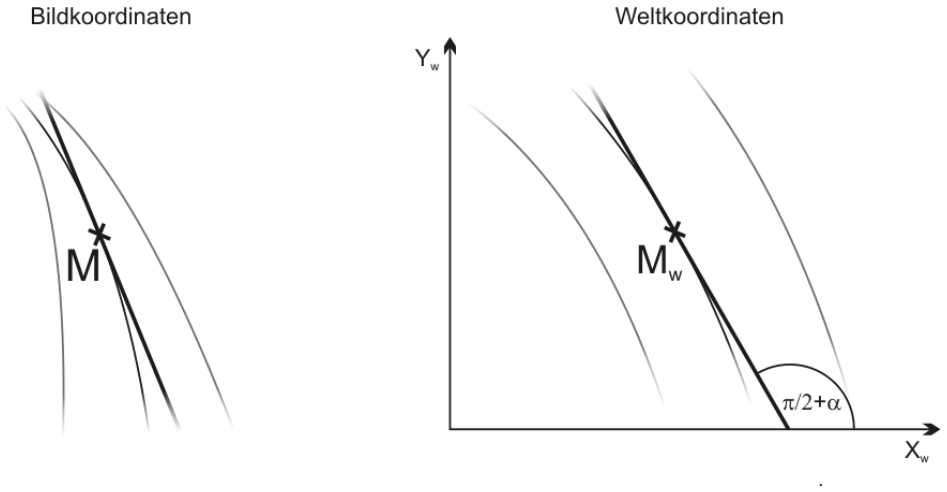

Abbildung 4.29: Ermittlung des Rotationswinkels α im Weltkoordinatensystem

Algorithmus 4.6 Einzeichnen des Lichtraumprofils ins Bild

```
function MaskeEinzeichnen(M_w, α, Maskenpunkte, Bild)
    for all Punkte P_w in Maskenpunkte do
        P_w1 ← DreheUmWinkel(P_w, α)              ▷ Rotation um α
        P_w2 ← (P_w1.X + M_w.X, P_w1.Y + M_w.Y)   ▷ Translation um M_w→
        p ← Bildkoordinaten(P_w2)                 ▷ In Bildkoordinaten transformieren...
        Linienzug.Add(p)                          ▷ ... und dem Linienzug hinzufügen
    ZeichneLinienzug(Linienzug, Bild)
    return Bild
```

werden. Auch im Sinne der besseren Auswertbarkeit durch einen Inspekteur ist daher eine relativ nahe Einzeichnung von Vorteil. Im Testvideo haben sich Entfernungen im Bereich von 15-25m als günstig erwiesen, dieser Wert variiert allerdings abhängig von der verwendeten Kamera, der Kalibrierung und dem Kamera-Standpunkt.

4.8 Testfälle und Bedingungen an die Videobilder

In Anhang B sind einige Testbilder aufgeführt, welche die Arbeit des Algorithmus in verschiedenen Situationen zeigen. Auf einige sei nachfolgend kurz eingegangen.

Die Abbildungen in Abschnitt B.1 zeigen gute Erkennungen in vielen Situationen, darunter auch in schattierten Bereichen, bei Tunnelfahrten und annähernd maximalen Kurvenradien. Abbildung B.6 erlaubt durch Verwendung der 2. Algorithmus-Varianten aus Abschnitt 4.6.4 sogar eine gute Erkennung, wenn die linke Schiene durch das Kantenbild nahezu gar nicht extrahiert werden konnte. Dies zeigt die Vorteile des verwendeten Algorithmus deutlich auf.

Abbildung 4.30: Eingezeichnetes Lichtraumprofil in 2 Entfernungen

Abschnitt B.2 zeigt Situationen, in denen die erkannten Schienen teilweise von den tatsächlichen abweichen. Abbildung B.7 erkennt die Schiene gut, allerdings wird im unteren Bildbereich der Schatten als Schiene erkannt, da er u.a. einen höheren Gradientenbetrag besitzt. Dies ist jedoch nicht weiter schlimm, wenn man die korrekte Einzeichnung des Lichtraumprofils im oberen Bildbereich betrachtet. Die anderen Bilder zeigen teilweises Fehlverhalten entweder im mittleren Bereich der Erkennung, was recht unproblematisch ist oder aber im oberen Bereich, wo auch das Lichtraumprofil eingezeichnet werden soll. Diese Fehlerkennungen werden durch Bildstörungen verursacht oder da eine parallele Schiene zeitweise einen höheren Gradientenbetrag besitzt. Sie treten aber nur temporär auf, i. Allg. lassen sich die Erkennungen im verwendeten Testvideo als gut einstufen.

In Abschnitt B.3 sind tatsächliche Fehlerkennungen verbucht. Abbildung B.15 zeigt die Einfahrt in einen Tunnel. Hier sind im dunklen Bereich keine Grauwerte mehr vorhanden, die zu einer Schiene gehören könnten, daraus resultiert die zeitweise Fehlerkennung. Abbildung B.16 zeigt schließlich noch eine Weichen-Situation. Hier springt die erkannte Schiene von der tatsächlichen Fahrspur zu der alternativen Spur über, allerdings nur für ein paar Frames, in denen die andere Spur als Kandidat in Frage kommt.

Allgemein lässt sich sagen, dass der Algorithmus gut arbeitet, es zwischen einzelnen Frames aber schnell zu einem Hin- und Herspringen unter einzelnen Schienenkandidaten kommen kann. Dies ließe sich durch einen Glättungsalgorithmus eliminieren, wodurch die Erkennung allgemein verbessert werden würde und sich auch das Weichen-Problem behandeln ließe.

Es sollte nochmals erwähnt sein, dass der Algorithmus nicht unter *worst case* Bedingungen getestet werden konnte, da diese im Video nicht vorkamen. Dazu zählen schlechtere Lichtverhältnisse, Nebel, Regen und Schnee. Vor allem bei den letzten beiden wird es zwangsläufig zu Problemen kommen, wenn sich Wassertropfen auf der Frontscheibe des Führerhauses befinden. Hier dürften

die Störungen im Bild durch Verzerrungen der Linien zu groß sein, um ordentliche Erkennungen durchführen zu können.

Verallgemeinernde Bedingungen an Videobilder zu stellen fällt schwer. Auf jeden Fall ist die Erkennung unabhängig von der generellen Grauwertintensität im Bild. In Abbildung B.3 wurden die Schienen dank Einsatz der robusten Hough-Transformation selbst bei einer dunklen Tunnelfahrt erkannt. Die Qualität der Erkennung ist lediglich abhängig von den Grauwertunterschieden zwischen der Schiene und ihrer Umgebung, da diese das wichtigste Hilfsmittel darstellen und nötig sind, damit der Kantendetektor zu guten Ergebnissen kommen kann. Durch Einsatz der Algorithmus-Varianten 2 aus Abschnitt 4.6.4 funktioniert die Erkennung zwar selbst dann gut, wenn durch die Kantenerkennung nur eine Schiene extrahiert wird, trotzdem muss diese sich dann sichtbar von der Umgebung abheben. Allgemein lassen sich daher folgende Bedingungen postulieren:

1. Schienen sollten die auffälligsten Kanten im zu untersuchenden Bildbereich darstellen.
2. Der Grauwertunterschied zwischen Schiene und Umgebung muss im Abstand von 3 Pixeln mindestens 15 Einheiten betragen (grob).
3. Mindestens eine zu erkennende Schiene muss der Bedingung 2 genügen.

Der in der 2. Bedingung angegebene Wert kann natürlich nur geschätzt sein und ist u.a. stark davon abhängig, wie die Umgebung um die Schiene herum beschaffen ist.

5 Implementierung

In diesem Kapitel soll die Implementierung der Schienenerkennung und der zugrunde liegenden Algorithmen beschrieben werden. Dazu wird die Klassenstruktur dargestellt und kurz auf die einzelnen Komponenten eingegangen. Intensiv auf die Umsetzung einzelner Algorithmen kann dabei allerdings nicht eingegangen werden, das würde den Rahmen dieser Arbeit sprengen.

5.1 Systemvoraussetzungen und eingesetzte Werkzeuge

Die Implementierung der Algorithmen und der Testapplikation wurde unter Einsatz folgender Werkzeuge realisiert:

- Betriebssystem: Microsoft Windows XP
- Programmiersprache: C# unter Verwendung des .NET Framework 2.0
- Programmierumgebung: Visual Studio 2005
- Bibliotheken: DirectShow .NET v1.2 (siehe [2])

Die Entwicklung fand auf einem Rechner mit AMD XP 2400+ Prozessor und 768 MB RAM statt, als Mindestvoraussetzungen für den Einsatz der Applikation sollten gelten:

- Prozessor-Geschwindigkeit: 1000 MHz
- Arbeitsspeicher: 512 MB RAM
- Betriebssystem: Windows 2000 / XP
- Laufzeitbibliothek: .NET Framework 2.0
- Grafikbibliothek: DirectX v9.0

5.2 Komponentenstruktur

Hier sollen die entwickelten Komponenten und ihr Zusammenhang untereinander kurz dargestellt werden. Die Verteilung der Klassen auf einzelne Dateien ist im Anhang C zu finden.

5.2.1 Mathematische Operationen

Die grundlegenden mathematischen Funktionen, Datenstrukturen und die mathematische Modellierungsvarianten der Schiene sind in den folgenden Klassen enthalten:

Klasse `MathCalc:` Beinhaltet ausschließlich statische Methoden, die von den anderen Komponenten genutzt werden können. U.a. sind dies: Gauß-Funktion, Lösung linearer Gleichungssysteme mittels QR-Zerlegung, Diskrete Fourier-Transformation, Schnelle Fourier-Transformation.

Klasse `Complex:` Datenstruktur zum Rechnen mit komplexen Zahlen, wird von der Fourier-Transformation benötigt.

Klasse `ParabolaModel (abstract):` Stellt gemeinsame Klassenmethoden-Köpfe zur Verfügung, die von den parabolischen Schienenmodellen implementiert werden müssen.

Klasse `Parabola : ParabolaModel:` Implementiert das parabolische Schienenmodell (siehe 4.6.1.1).

Klasse `LinearParabola : ParabolaModel:` Implementiert das linear-parabolische Schienenmodell (siehe 4.6.1.2).

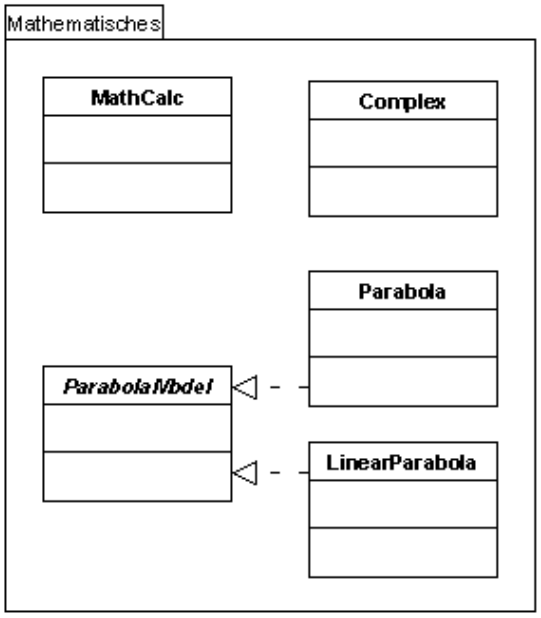

Abbildung 5.1: Klassendiagramm der mathematischen Operationen

5.2.2 Photogrammetrie-Algorithmen

So wichtig diese Implementierung ist, so gering ist auch ihr Umfang. Die gesamte Funktionalität befindet sich in folgender Klasse:

Klasse `Calibration`: Stellt Funktionen für die Kamerakalibrierung mittels DLT und die Umrechnung von Welt- in Bildkoordinaten und umgekehrt zur Verfügung. Die Kalibrierung wird durch Übergabe einer XML-Datei (welche die Kalibrierungspunkte enthält) oder von mindestens 6 Passpunkten erstellt, indem das entstehende Gleichungssystem gelöst wird (siehe Kapitel 2).

Klasse `Point2D`: Stellt einen Punkt (x, y) im Bildkoordinatensystem dar.

Klasse `Point3D`: Stellt einen Punkt (X_w, Y_w, Z_w) im Weltkoordinatensystem dar.

Klasse `PassPoint`: Speichert je einen Bild- und einen Weltpunkt, um ihn als Passpunkt für die Kalibrierung zu verwenden.

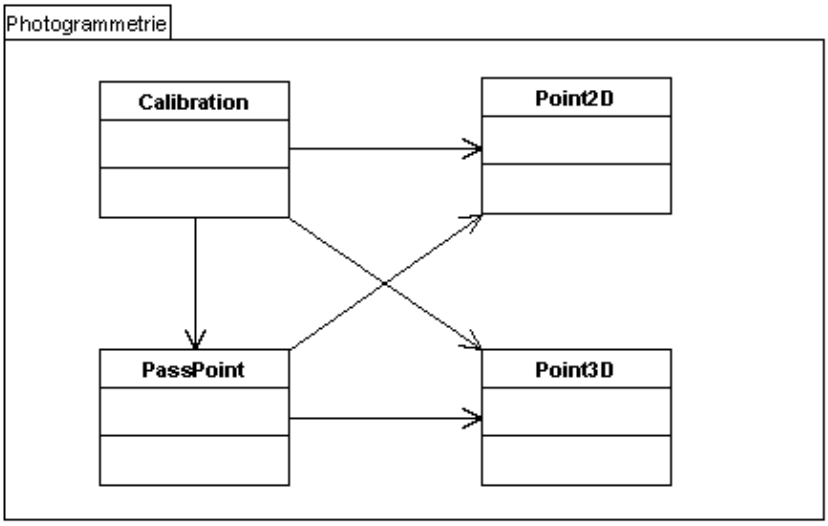

Abbildung 5.2: Klassendiagramm der Photogrammetrie-Algorithmen

5.2.3 Bildverarbeitungs-Algorithmen

Der größte Entwicklungsaufwand floss in diese Algorithmen. Zentral ist die Klasse `ImageGray` anzusehen, die ein Grauwertbild bereitstellt, auf dem alle Bildverarbeitungs-Routinen arbeiten. Diese können problemlos in einer eigenständigen kleinen Bildverarbeitungsbibliothek zusammengefasst und später wieder verwendet werden.

Klasse `ImageGray`: Datenstruktur für ein Grauwertbild. Dem Konstruktor kann ein normales `Bitmap` übergeben werden, welches in Graustufen umgewandelt wird. Ebenso kann ein `ImageGray`-Objekt in ein `Bitmap` zurück gewandelt werden. Zur Beschleunigung der Zugriffe auf ein `Bitmap` wurde dabei unsicherer Code verwendet. Direkt hier enthaltene Algorithmen

sind u.a.: Addition / Subtraktion von Grauwertbildern, Erkennung von Nulldurchläufen, Binarisierung, Unscharfmaskierung, Highboost-Filter, Kontrast-Dehnung, Kontrast-Anpassung, Anwendung einer LUT, Gamma-Korrektur, Winkelfilterung nach 4.3.5.4.

Klasse `ImageComplex`: Datenstruktur für ein Grauwertbild, dessen Pixelwerte als komplexe Zahlen gespeichert werden, um eine Fourier-Transformation auf diesem Bild durchführen zu können. Beinhaltet im Besonderen die folgenden Operationen im Fourier-Raum: Laplace-Operator, ideale Hoch- und Tiefpassfilterung, Bandpass-Filter, LoG-Operator.

Klasse `HistOps`: Beinhaltet Methoden für die Berechnung von einfachem und kumulativem Histogramm und Operationen darauf, z.B.: linearer Histogramm-Ausgleich, Histogramm-Anpassung, Histogramm-Anpassung an die Gauss-Verteilung.

Klasse `HoughTransform`: Bietet Funktionen für die Hough-Transformation von Geraden, u.a. die globale HT, aber auch eine Version, die Linien erkennt, welche von bestimmten Punkten ausgehen.

Klasse `Filter (abstract)`: Stellt gemeinsame Methoden-Köpfe für die Anwendung von Filtern zur Verfügung.

Klasse `LinearFilter : Filter`: Allgemeine Implementation linearer Filter. Dem Konstruktor kann eine Filtermatrix übergeben werden, mit der ein Bild verarbeitet werden soll. Die Implementation der abstrakten Methode zur Filteranwendung der Klasse `Filter` nutzt diese Matrix dann für die lineare Filterung eines Bildes.

Klasse `LinearFilters`: Enthält lediglich statische get-Properties, die bestimmte lineare Filter zurückgeben. Implementiert sind derzeit u.a.: Gauss-, Tiefpass-, Hochpass- und Laplace-Filter.

Klasse `AdaptiveGaussFilter : LinearFilter`: Stellt einen linearen Filter dar, an den zusätzlich ein Wert σ zur adaptiven Berechnung der Filtermaske über die Gauss-Funktion übergeben werden kann.

Klasse `GradientFilter : LinearFilter`: Allgemeine Implementation von Gradientenfiltern, überschreibt die Methode zur Anwendung eines Filters, wobei das Filterergebnis aus dem Absolutbetrag der horizontalen und vertikalen Gradientenbeträge resultiert. Ebenso können die Gradientenorientierungen berechnet und zurückgegeben werden. Dem Konstruktor wird wieder eine Matrix übergeben, welche den jeweiligen Operator in horizontaler oder vertikaler Richtung darstellt.

Klasse `GradientFilters`: Definiert ausschließlich statische get-Properties, die jeweils spezifische `GradientFilter`-Objekte zurückgeben. Implementiert sind zur Zeit die Roberts-, Prewitt- und Sobel-Operatoren.

Klasse `MarrHildrethFilter : Filter`: Implementation einer Kantenerkennung nach dem Marr-Hildreth-Kantendetektor.

Klasse `CannyFilter : Filter`: Implementation des Canny-Kantendetektors.

Klasse `ShenCastanFilter : Filter`: Implementation des Shen-Castan-Kantendetektors.

Klasse `OptimalEdgeFilters`: Beinhaltet statische get-Properties, die jeweils Objekte der optimalen Kantendetektoren nach Marr-Hildreth, Canny und Shen-Castan zurückgeben.

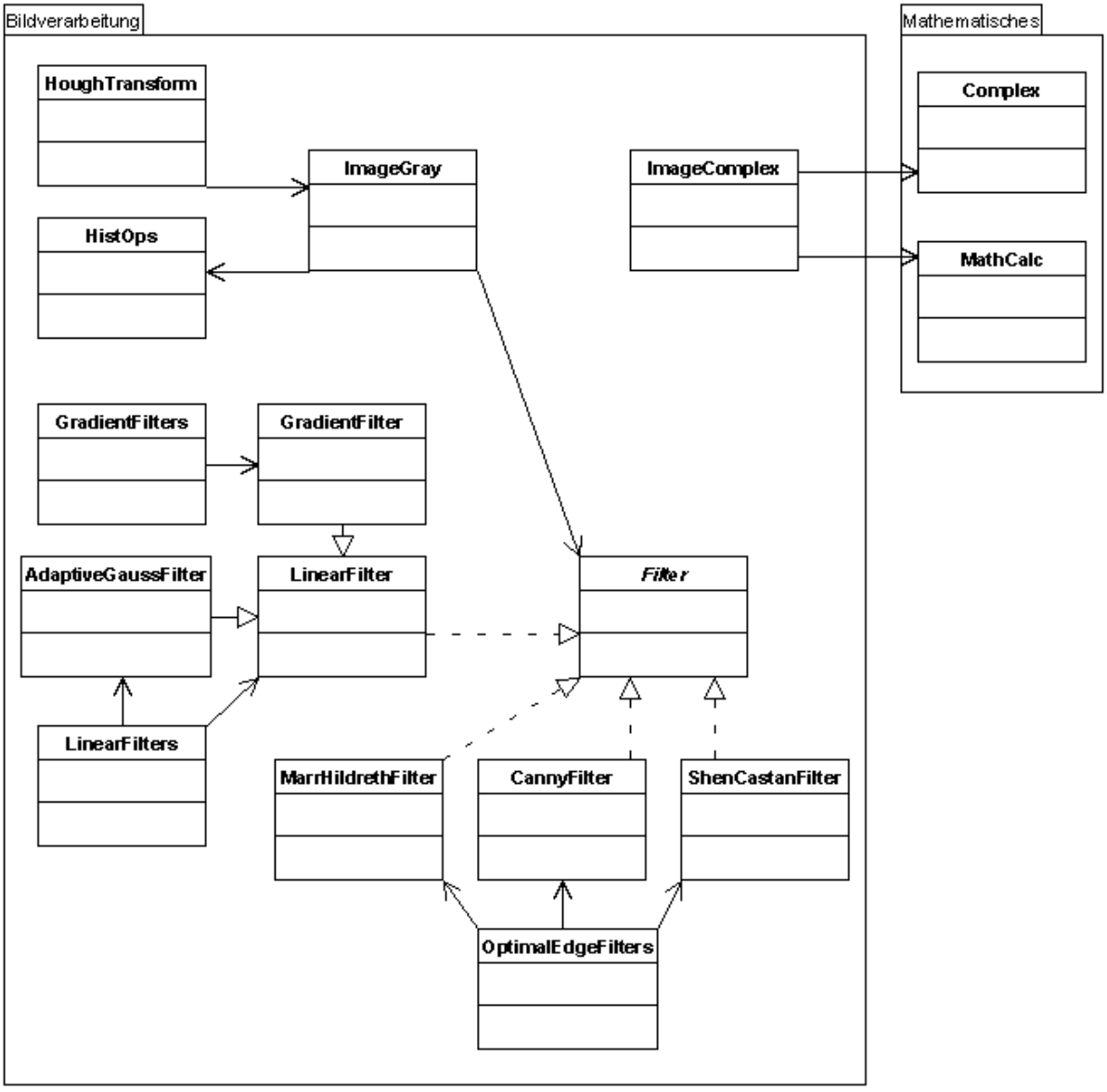

Abbildung 5.3: Klassendiagramm der Bildverarbeitungs-Algorithmen

5.2.4 Schienenerkennungs-Algorithmus

Die hier entworfenen Klassen stellen den wichtigsten Teil der Arbeit dar und führen die Photogrammetrie sowie die Bildverarbeitung zusammen, um den in 4.6 beschriebenen Algorithmus zu implementieren:

Klasse `RailProcessing`: Stellt die eigentliche Implementierung des Schienenerkennungs-Algorithmus dar. Dem Konstruktor werden Kalibrierungsdaten übergeben, woraufhin die Kalibrierung erzeugt und der Bildraum initial begrenzt wird (siehe Kapitel 3). Ebenso kann ein Lichtraumprofil gesetzt werden, welches nach der Erkennung einzuzeichnen ist. Über eine Methode kann die Schienenerkennung in einem Bild gestartet werden, eine andere ist für die Einzeichnung wichtiger Informationen in ein Bild zuständig. Dazu gehören: Bildraumbegrenzung, Kantenbild, Schienenkandidaten, lineare Schienenstücke, Schienenparabeln, Lichtraumprofil.

Klasse `ImageLimiter`: Datenstruktur, die zur Begrenzung einer Bildzeile mit der Ordinate y dient. Dazu werden u.a. y, x_{min} sowie x_{max} aufgenommen und zusätzlich der Winkelbereich, welcher für diese Zeile gültig ist. Ein zweidimensionales Array aus `ImageLimiter`-Elementen kann für die initiale Bildraumbegrenzung und die Unterteilung in Teilbereiche nach 4.6.2 dienen, wobei die 1. Dimension den Teilbereich und die 2. Dimension die begrenzten Zeilen des jeweiligen Teilbereiches angibt.

Klasse `Line`: Speicherstruktur für eine Gerade, wie sie z.B. durch die HT ermittelt wurde. Die Speicherung erfolgt in HNF, mit Winkel α und Radius r, zusätzlich sind Definitionen für den Start- und Endpunkt einer Linie enthalten.

Klasse `HoughLine`: Um den Hough-Wert einer Linie nicht doppelt berechnen zu müssen, kann hier sowohl das jeweilige `Line`-Objekt als auch sein Wert im Hough-Akku gespeichert werden.

Klasse `CandidateLines`: Beinhaltet je ein Array zur Aufnahme linker und rechter Schienenkandidaten in Form von `HoughLine`-Objekten. Ein Array von `CandidateLines` wird beispielsweise von der Hough-Transformation für die oberen Teilbereiche zurück gegeben, um daraus dann die besten Kandidaten auswählen zu können. Dabei findet eine Verwendung so statt, dass ein `CandidateLines`-Objekt genau die Kandidaten aufnimmt, die von einem Gewinner-Schienenpaar des nächstunteren Teilbereiches ausgehen.

Klasse `WinningCandidate`: Nimmt ein Gewinner-Schienenpaar eines Teilbereiches auf. Dazu zählt die jeweils linke und rechte Schiene als `Line`-Objekt sowie deren Vorgänger-Schienen aus dem nächstunteren Teilbereich. Hinzu kommt der Wert, den das Gewinner-Paar nach den Kriterien von 4.6.4.3 erhalten hat. Ein Array von `WinningCandidate`-Objekten bildet den Ausgangspunkt (die Saat) für die Verarbeitung des nächsten Teilbereichs.

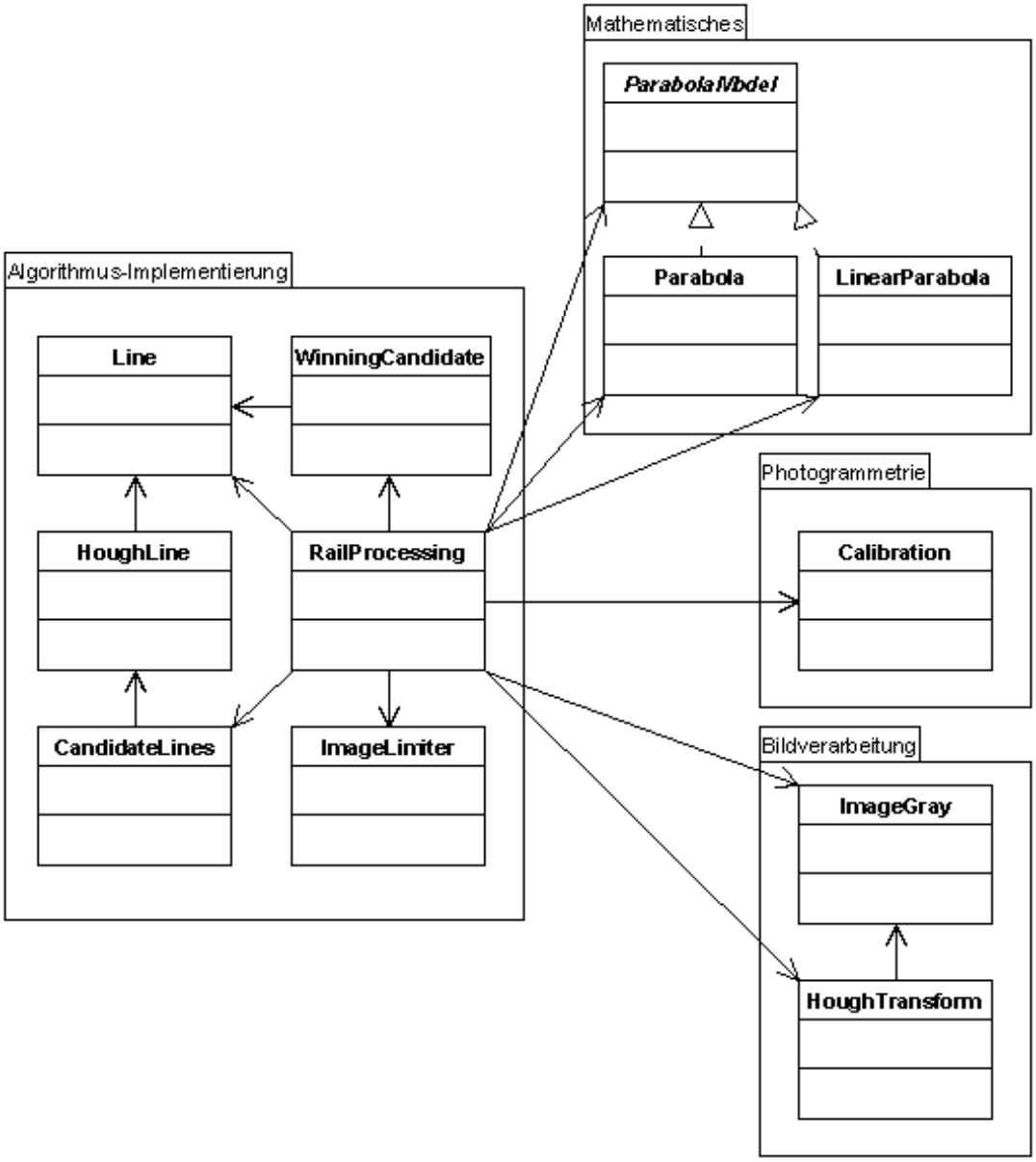

Abbildung 5.4: Klassendiagramm des Schienenerkennungs-Algorithmus

5.2.5 Rahmenprogramm

Das Rahmenprogramm bildet die Grundlage für das Testen des Algorithmus und beinhaltet viele Funktionen, mit denen sich einzelne Verarbeitungsoptionen ein- bzw. ausschalten lassen. Es erlaubt dabei das Laden einer Kalibrierung und eines einzuzeichnenden Lichtraumprofils, um dann ein gewähltes Video damit auswerten zu können. Die Parameter für die Kalibrierung und das Lichtraumprofil werden dabei in XML-Dateien gespeichert, deren Struktur in Anhang D erläutert wird. Die Verarbeitung der Videobilder kann auf 2 unterschiedliche Arten erfolgen: 1.) es wird jedes Videobild einzeln ausgewertet, 2.) es wird nicht jedes Bild ausgewertet, die

Abspielgeschwindigkeit des Videos ist dabei stufenlos einstellbar. Eine ausführliche Beschreibung der einzelnen Funktionalitäten finden Sie auf der CD im Ordner DOKUMENTATION.

Folgende Klassen bilden das Rahmenprogramm:

Klasse `DxPlay`: Baut auf der *DirectShow .NET* Bibliothek (siehe [2]) auf und nutzt deren Funktionalitäten, um ein Video abzuspielen und zu steuern sowie einzelne Frames als Bitmaps zu erhalten. Löst ein Ereignis beim Eintreffen neuer Frames aus.

Klasse `FormDemoApp`: Diese stellt die eigentliche Programmfunktionalität zur Verfügung und beinhaltet dabei das Hauptfenster sowie die Konfigurationsmenüs. Von hier aus wird die Auswertung eines Videos gesteuert.

Klasse `FormStartProcessing`: Über diese Windows Form können das auszuwertende Video, die Kalibrierung sowie das einzuzeichnende Lichtraumprofil gewählt und die Verarbeitung gestartet werden. Ebenso lassen sich die wichtigsten Parameter definieren (Entfernung des Lichtraumprofils, Spurweite etc.), welche bei der Auswertung verwendet werden sollen.

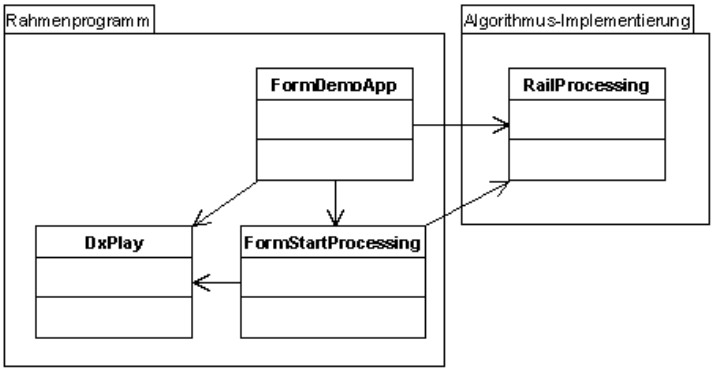

Abbildung 5.5: Klassendiagramm des Rahmenprogramms

6 Schlussbemerkungen

6.1 Ausblick

Auch wenn der Algorithmus in seiner derzeitigen Form und angewendet auf das vorhandene Testvideo gut funktioniert, sind weitere Schritte denkbar, die zu einer Verbesserung der Erkennung und einer Effizienzsteigerung führen können:

1. Weiteres Testmaterial ist auszuwerten und der Algorithmus daran zu messen.
2. Die Bildverarbeitungsalgorithmen sind auf eine Laufzeitverkürzung hin zu überprüfen, vor allem bei der HT gibt es viele Variationen, die eine Senkung der Berechnungskosten versprechen.
3. Es ist zu prüfen, ob sich weitere A-priori-Informationen zunutze machen lassen.
4. Die parabolischen Modelle könnten durch komplexere ersetzt werden, z.B. Splines.
5. Um „Ausreißer“ in aufeinander folgenden Erkennungen zu vermeiden, könnte eine Glättung stattfinden, wobei Kandidaten bevorzugt als Gewinner akzeptiert werden, wenn sie nahe am vorherigen erkannten Fahrweges liegen.
6. Es wäre denkbar, dass der vorgestellte Algorithmus nur für eine initiale Erkennung heran gezogen wird und darauf folgende Bilder anhand dieser initialen Erkennung ausgewertet werden. So könnte mitunter auch der zu untersuchende Bereich weiter eingegrenzt werden.

6.2 Fazit

Diese Arbeit hatte zum Ziel, die Instandhaltungsaufgabe von Bahnanlagen durch maßstäbliches Einzeichnen von Lichtraumprofilen auf den aktuellen Fahrweg zu unterstützen. Dabei stellte sich heraus, dass die Hauptaufgabe in der Entwicklung eines Algorithmus zur Schienenerkennung bestand. Dieser beruht auf der Erkennung von Geraden in Teilbereichen eines initial eingegrenzten Bildraums und nutzt vorhandene A-priori-Wissen aus, um die tatsächlich vorkommenden Schienen möglichst exakt zu extrahieren.

Der entstandene Algorithmus liefert in einer Vielzahl von Fällen eine gute Erkennung der Fahrspur, auch in Situationen wo die Kantendetektion nur für eine Schiene ordentliche Ergebnisse liefert. Eine hundertprozentige Erkennung wird man indes nicht erhalten, da das Bild von zu

vielen Störfaktoren beeinflusst werden kann und der Kantendetektor zum einen selten die Schiene exakt extrahieren kann und zum anderen auch auf andere Bildelemente reagiert. Dies stellt aber allgemein ein Problem aller Algorithmen dar, die auf einer Kantendetektion beruhen. Im vorliegenden Fall ist sie ein unverzichtbarer Bestandteil, der durch nichts ersetzt werden kann.

Im Zuge der Erprobung verschiedener Techniken ist ganz nebenbei noch eine kleine Bildverarbeitungsbibliothek entstanden, die neben vielen linearen Filtern auch Grauwerttransformationen, FT-Operationen, die Hough-Transformation für Geraden sowie einige Kantendetektoren beinhaltet und die Grundlage für weitere Entwicklungen bilden kann.

Meiner Meinung nach war die mir gestellte Aufgabe fordernd, aber lösbar. Das entstandene System ist zur manuellen Störungssuche im Fahrweg-Umfeld durch Einzeichnen des Lichtraumprofils gut geeignet, wobei zusätzliche Erweiterungen denkbar sind. So wäre es z.B. möglich, durch Verwendung von 2 Kameras und einer Stereokalibrierung dieser sowie Algorithmen zur Objektverfolgung vollautomatisch zu ermitteln, ob ein Objekt das Lichtraumprofil schneidet.

6.3 Danksagung

Ich bedanke mich zunächst bei meinem Betreuer Prof. K.-U. Jahn, der mir mit seinem Fachwissen und der stetigen Diskussionsbereitschaft sehr zu helfen wusste.

Ebenso möchte ich mich bei Herrn Markus Maspfuhl, Jens Heinrich und der gesamten Firma CCC bedanken, die mir das Thema der Bachelorarbeit bereitgestellt und mich jederzeit tatkräftig unterstützt haben.

Mein Dank richtet sich auch an meine Kommilitonen, die immer gute Ratschläge und Korrekturen eingebracht haben und mich bei guter Laune hielten.

Last but *not* least danke ich meiner Verlobten Katrin, die mich jederzeit unterstützt und mir die Kraft und Motivation gegeben hat, die ich während des Anfertigens der Bachelorarbeit benötigt habe. Ohne sie wäre mein Tatendrang schnell verloren gegangen und die hier vorliegende Arbeit würde nicht existieren.

A Glossar

1-Normierung Dabei wird ein Vektor oder ein Wert so normiert, dass seine maximale Länge bzw. Größe 1 ergeben kann.

Äußere Orientierung Gibt die Lage der Kamera bezüglich des Weltkoordinatensystems an. Besteht aus dem Ursprung des Weltkoordinatensystems 0_w sowie den 3 Winkeln ω, ϕ und κ, welche die Rotation beschreiben, um das Kamerakoordinatensystem in das Weltkoordinatensystem zu transformieren.

Akkumulator-Array Diskrete Speicherungsstruktur des → Parameterraums der HT.

Bandpassfilter Filter, der nur Frequenzen in einem bestimmten Bereich erlaubt.

Bedingte Optimalität Optimalität bezüglich wohldefinierter Kriterien bzw. Bedingungen.

Bildanalyse Nach D. Marr: „Aus Bildern erkennen, was in der Welt ist und wo es ist."

Bildverarbeitung Nutzt Mittel der Signalverarbeitung zur Aufbereitung und Speicherung visueller Informationen. Dient als Zwischenstufe zu einer weiteren maschinellen Verarbeitung der Bilddaten.

Binomial-Filtermaske Maske, deren Elemente der Binomial-Verteilung genügen.

Brute-Force Problemlösung durch Ausprobieren aller möglichen Varianten.

BV Bildverarbeitung.

DFT → Diskrete Fourier-Transformation.

Differenzfilter Linearer Filter, bei dem einzelne Filterkoeffizienten negativ sind. Dadurch werden örtliche Unterschiede verstärkt. Differenzfilter werden oft zur Kantenextraktion sowie Bildschärfung verwendet.

Direkte Lineare Transformation Ein einfaches Verfahren zur Kamerakalibrierung.

Diskrete Fourier-Transformation Diskrete Version der FT, die im Falle eines diskreten, periodischen Signals angewendet werden kann.

DLT → Direkte Lineare Transformation.

Fourier-Raum → Frequenzraum.

Fourier-Transformation In der BV Zerlegung von Bildsignalen in Sinus- und Kosinusfunktionen, womit die Bilder in den Frequenzraum transformiert werden.

Frequenzraum Ist das Ergebnis der vorwärtsgerichteten Fourier-Transformation. Stellt die in einem Bild vorkommenden Frequenzen dar.

FFT → Schnelle Fourier-Transformation.

FT → Fourier-Transformation.

Gauß-Funktion Funktion, die der Gauß-Verteilung genügt. Definiert durch $g_\sigma(x) = e^{(\frac{-x^2}{2\sigma^2})}$.

get-Property Lesender Zugriff auf die spezifizierte Eigenschaft in C#. Dabei könnten Klassenvariablen zurück gegeben werden, es ist aber z.B. auch möglich neue Objekte zu erzeugen bzw. eigenen Code auszuführen und dessen Ergebnisse zurückzugeben.

Gradientenfilter Differenzfilter, der eine Schätzung der 1. Ableitung realisiert und somit eine Kantenextraktion in eine bestimmte Richtung ermöglicht.

Hesse'sche Normalform Gleichung in der analytischen Geometrie, die eine Ebene im R^3 bzw. eine Gerade im R^2 beschreibt.

Highboost-Filter Filter, bei dem vom gewichteten Originalbild ein tiefpassgefiltertes Bild subtrahiert wird. Führt zu einer Verstärkung hoher Frequenzen und damit zu einer Bildschärfung.

Histogramm Eindimensionales Feld $h(i)$, in dem die Anzahl der Elemente der Anzahl der maximal vorkommenden Grauwerte (i. Allg. 255) entspricht. Jedes Feldelement enthält dabei die Summe der Vorkommen des jeweiligen Grauwertes.

Histogrammanpassung Methode, bei der das Histogramm bzw. das kumulative Histogramm eines Bildes an eine bestimmte Verteilungsform bzw. ein anderes Histogramm angepasst wird.

Histogrammausgleich Verteilt Grauwerte eines Bildes so, dass ein annähernd gleichverteiltes Histogramm entsteht.

HNF → Hesse'sche Normalform.

Hochpassfilter Linearer Filter, der hohe Frequenzen erlaubt und das Bild somit schärft.

Homogene Koordinaten Rotation, Skalierung und Scherung dreidimensionaler Objekte lassen sich durch Matrizenmultiplikation realisieren, Translation jedoch nur durch Addition. Homogene Koordinaten erweitern den Raum um eine Dimension und erlauben somit, alle Transformationen mittels Multiplikation von Matrizen zu beschreiben.

Houghraum → Parameterraum.

Hough-Transformation Robustes rauschunempfindliches Verfahren zur Erkennung von Geraden, Kreisen oder beliebigen anderen parametrisierbaren geometrischen Figuren in einem gegebenen Kantenbild (setzt somit eine Kantendetektion voraus).

HT → Hough-Transformation.

Hysteresis-Tresholding Schwellwertverfahren, bei dem Pixel erhalten bleiben, die einen oberen Schwellwert übersteigen sowie damit verbundene Pixel, wenn sie über einem unteren Schwellwert liegen.

Innere Orientierung Beschreibt die Abbildung der 3D-Punkte vom Kamerakoordinatensystem in das Bild. Setzt sich im einfachsten Fall aus den Koordinaten des Bildhauptpunktes H sowie der Kamerakonstanten c zusammen.

Interlacing „Zeilensprungverfahren"; dabei setzt sich ein (Video-) Bild aus 2 Halbbildern zusammen, wodurch aufeinander folgende Bilder versetzt erscheinen. Dieser Effekt ist vor allem bei Bildern zu beobachten, die sich stark voneinander unterscheiden oder bei sich schnell ändernden Objekten.

Kamera-Kalibrierung Mithilfe der Kamera-Kalibrierung stellt man den mathematischen Zusammenhang zwischen Welt- und Bildkoordinaten her, mit welchem sich anschließend Punkte der beiden Koordinatensysteme ineinander umrechnen lassen.

Kameramodell Ein Kameramodell stellt die Abstraktion einer Klasse realer Kameras dar.

Kantendetektor Bildverarbeitungsalgorithmus, der die in einem Bild vorhandenen Kanten extrahiert. Ergebnis ist meist das binäre Kantenbild.

Kontrastanpassung Verfahren zur Bildverbesserung, welches den Kontrast verbessert, indem der Grauwertbereich, der für ein Bild relevant ist, linear auf die gesamte Intensitätsverteilung abgebildet wird.

Kumulatives Histogramm Eindimensionales Feld $H(i)$, bei dem sich der Wert des Elementes i aus der Summe aller Elemente $j \leq i$ des zugehörigen Histogramms $h(j)$ ergibt.

Laplace-Filter Differenzfilter, der eine Schätzung der Laplace-Funktion realisiert und somit zur Kantenschärfung geeignet ist.

Laplace-Funktion Basiert auf der 2. Ableitung und ist im Zweidimensionalen definiert als: $\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$.

Lichtraumprofil Definierte Umgrenzungslinie, in dieser Arbeit eine Maske, welche die Höchstmaße eines Zuges im Querschnitt darstellt und sich orthogonal zur Fahrspur befindet. Stellt den *lichten Raum* dar, der frei von anderen Gegenständen sein muss.

Linearer Filter Verknüpft die Pixelwerte innerhalb der Filterregion in linearer Form, d.h. durch eine gewichtete Summe. Dadurch ist eine große Menge verschiedener Filter realisierbar, z.B. zum Glätten von Bildern.

Lochkameramodell Kameramodell, welche die Klasse der Kameras mit punktförmigem Projektionszentrum beschreibt.

LoG-Operator *Laplacian of Gaussian*; Laplace-Operator auf die Gauss-Funktion angewendet. Hat im Zweidimensionalen eine typische Mexikanerhut-Form und stellt somit einen Bandpassfilter dar. Wird u.a. zur Kantendetektion beim Marr-Hildreth-Kantendetektor eingesetzt.

Lookup-Tabelle Einfaches Array von Zahlen. Der „Lookup" eines Array-Elements liefert dabei dessen Wert. Findet z.B. Verwendung, um einem alten Grauwert, der als Index übergeben

wird, einen neuen Wert zuzuweisen, welcher sich aus dem entsprechenden Element der LUT ergibt.

LUT → Lookup-Tabelle.

Non-Maximum-Suppression Verfahren, mit dem lokale Maxima durch Unterdrückung nichtmaximaler Werte gefunden werden. So muss ein Pixel unter allen seinen Nachbarn den höchsten Wert haben, um als lokales Maximum zu gelten.

Optimaler Kantendetektor Filter, dessen Optimalität sich auf eine bestimmte mathematische Bedingung bezieht. Beispiele sind der Marr-Hildreth-, Canny- und Shen-Castan-Kantendetektor.

Orientierungs-Parameter Parameter der inneren sowie äußeren Orientierung (siehe auch 2.2 und 2.3).

Parameterraum (auch Houghraum oder $r\phi$-Raum) Bei der HT für Geraden ein zweidimensionaler Raum mit den Achsen r und ϕ (den Geradenparametern).

Passpunkt (Referenzpunkt) Ein Passpunkt ist ein Objektpunkt, dessen Position $\mathcal{P}_w^i$ im Weltkoordinatensystem sowie die Lage seiner Abbildung $\mathcal{P}^i$ im Bildkoordinatensystem bekannt ist. Mit Hilfe einer Menge von Passpunkten lässt sich die Kamera-Kalibrierung durchführen.

Photogrammetrie Die Photogrammetrie beschäftigt sich mit der 3D-Rekonstruktion von Objekten aus perspektivischen Abbildungen wie Fotografien. Ihre Algorithmen lassen sich dazu verwenden, um berührungslose Messungen an Objekten und die Umrechnung von Objekt- in Bildkoordinaten (und entgegengesetzt unter der Annahme einer vorhandenen Bezugsebene, siehe auch 2.3) durchzuführen.

Punktoperation Operation, die einen Bildpunkt P so verändert, dass das Ergebnis nur von P abhängt und von keinen anderen Bildpunkten.

QR-Zerlegung Zerlegung einer Matrix A in das Produkt $A = Q * R$, wobei Q eine orthogonale, unitäre Matrix darstellt. Die QR-Zerlegung dient als stabiles numerisches Verfahren zur Ausgleichslösung linearer Gleichungssysteme.

Schnelle Fourier-Transformation Schneller Algorithmus zur Berechnung der DFT. Reduziert die Zeitkomplexität i. Allg. von $O(M^2)$ auf $O(M \cdot log_2 M)$.

Sinoide Sinusförmige Kurve.

Tiefpassfilter Linearer Filter, durch den das Bild geglättet wird (erlaubt tiefe Frequenzen, daher der Name).

Unsicherer Code (*unsafe code*) In C# ein Quelltext-Bereich, in dem mit Zeigern gearbeitet werden kann. Dadurch sind allerdings Speicherzugriffsfehler möglich.

Weltkoordinatensystem Das Weltkoordinatensystem spannt einen dreidimensionalen Raum auf, aus welchem die Abbildung auf 2D-Bildkoordinaten durchgeführt werden soll. Aus ihm ergibt sich durch Translation und Rotation das Kamera-Koordinatensystem und aus diesem schließlich die Abbildung in Bildkoordinaten.

B Testbilder

Hier soll anhand einiger Testbilder gezeigt werden, wo der entwickelte Algorithmus gut funktioniert, wo nur teilweise und vor allem auch wo er an seine Grenzen stößt. Die Teilbilder geben dabei jeweils an: links: Grauwertbild, Mitte: Kantenbild nach Shen-Castan, rechts: Erkannte Schiene und eingezeichnetes Lichtraumprofil. Als Vorverarbeitungsschritt wurde ein Gauß-Tiefpassfilter und eine automatische Kontrastanpassung angewandt (siehe Abschnitt 4.5). Mit einer zusätzlichen Filterung anhand der Gradientenrichtung (siehe 4.3.5.4) können die Ergebnisse teilweise noch verbessert werden. Abschnitt 4.8 geht zusätzlich auf einzelne dieser Bilder ein.

B.1 Gute Erkennungen

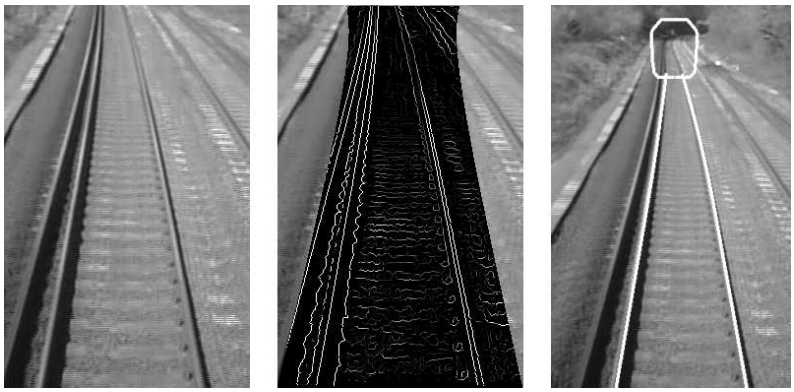

Abbildung B.1: Perfekte Erkennung

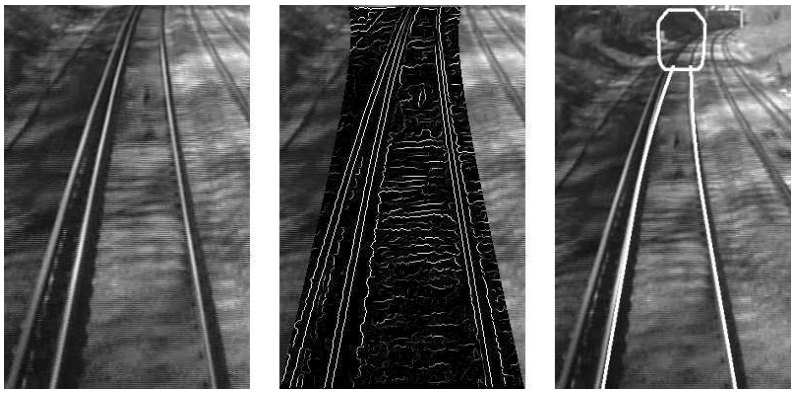

Abbildung B.2: Gute Erkennung in schattierten Situationen

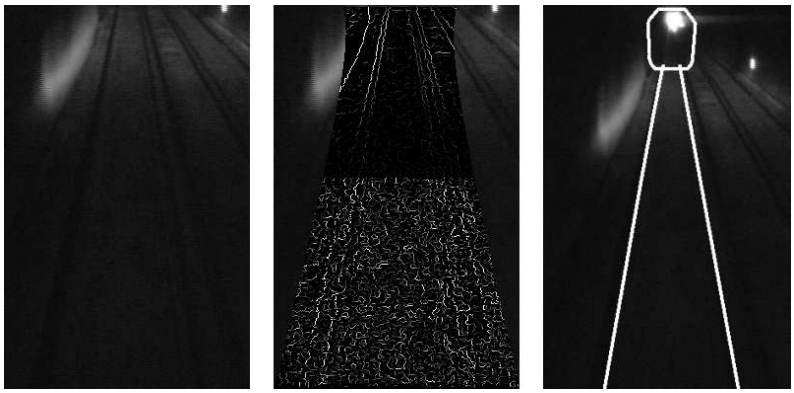

Abbildung B.3: Gute Erkennung im Tunnel, wenn sich die Schiene vom Untergrund abhebt

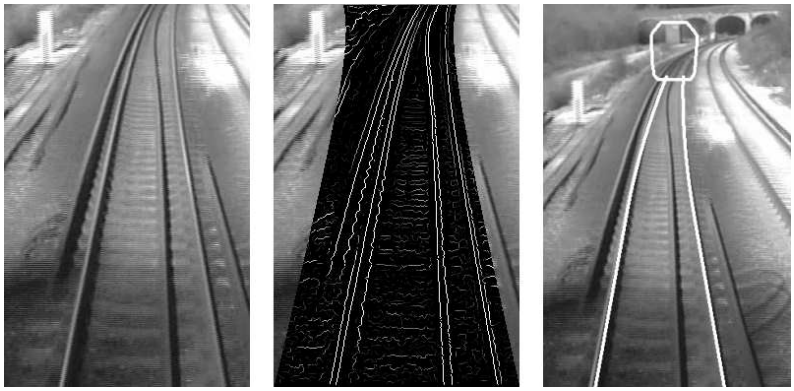

Abbildung B.4: Gute Erkennung bei nahezu maximaler Kurvenfahrt

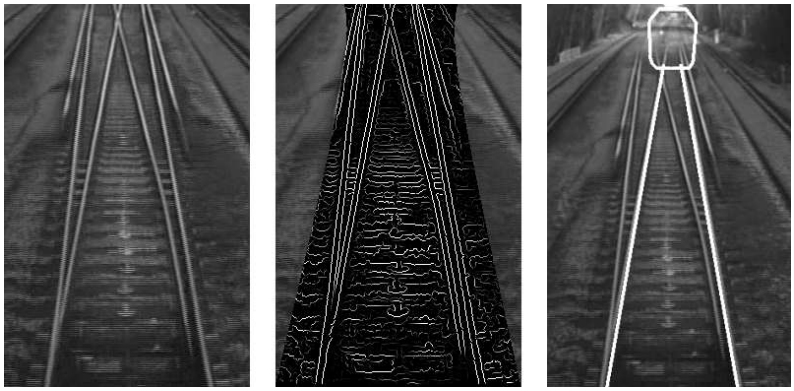

Abbildung B.5: Gute Selektion bei mehreren Schienen

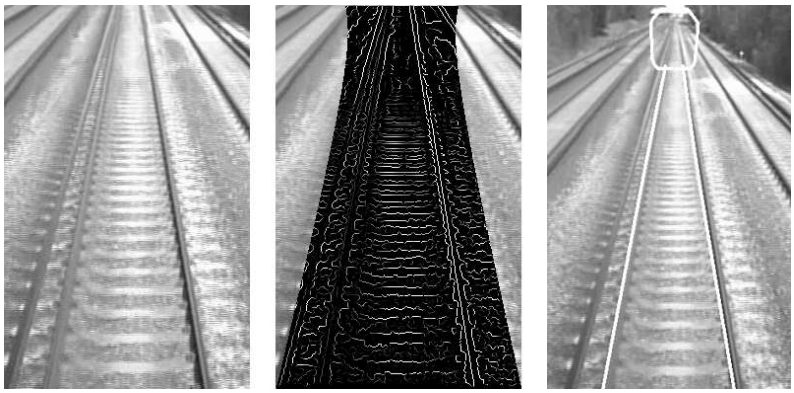

Abbildung B.6: Gute Erkennung, wenn nur eine Schiene von der HT erkannt werden konnte

B.2 Teilweise Erkennungen

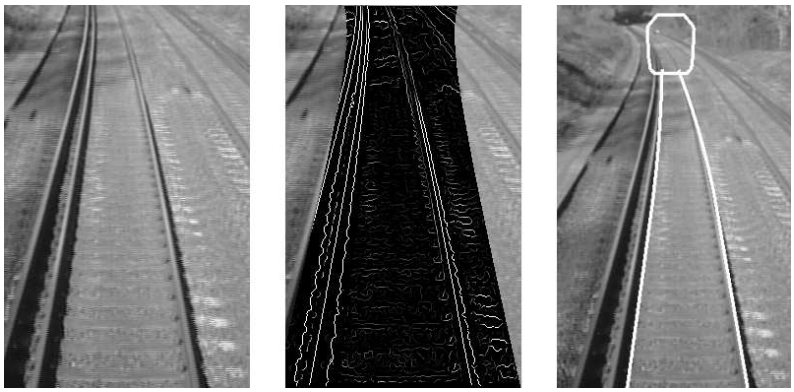

Abbildung B.7: Im vorderen Bereich Schatten als Schiene erkannt

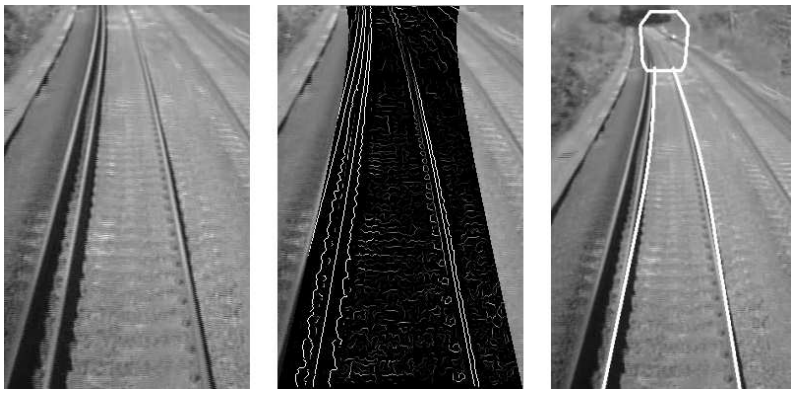

Abbildung B.8: Abweichung im mittleren Teil

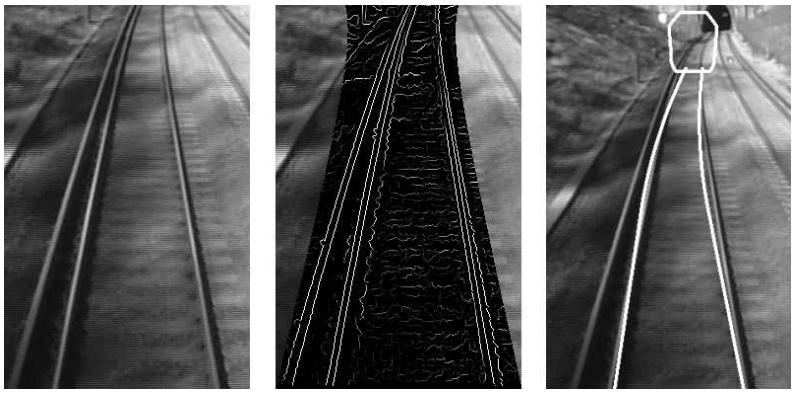

Abbildung B.9: Temporäre Falsch-Erkennung

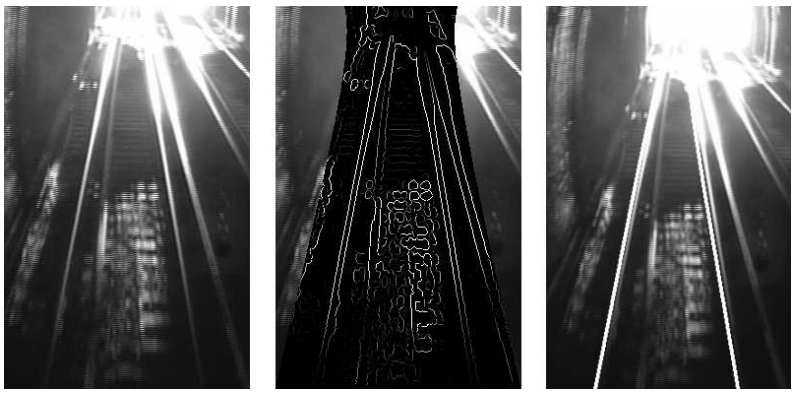

Abbildung B.10: Erkennung im oberen Teil durch gleißendes Licht nicht möglich

Abbildung B.11: Kurzzeitige Fehlerkennung durch Bildstörungen

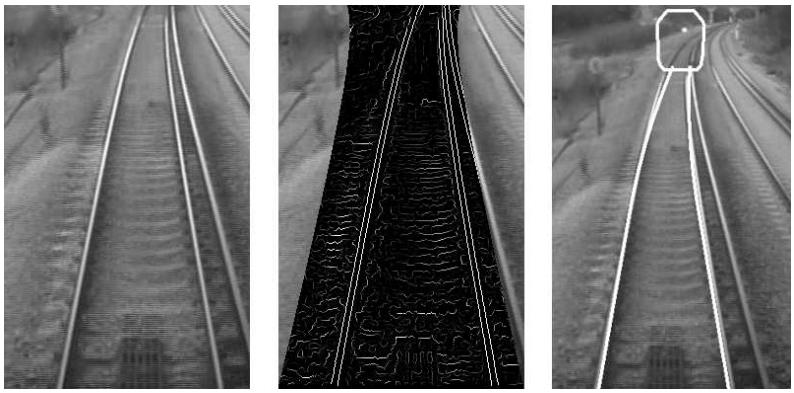

Abbildung B.12: Erkennung des Schattens als Schiene im hinteren Bereich

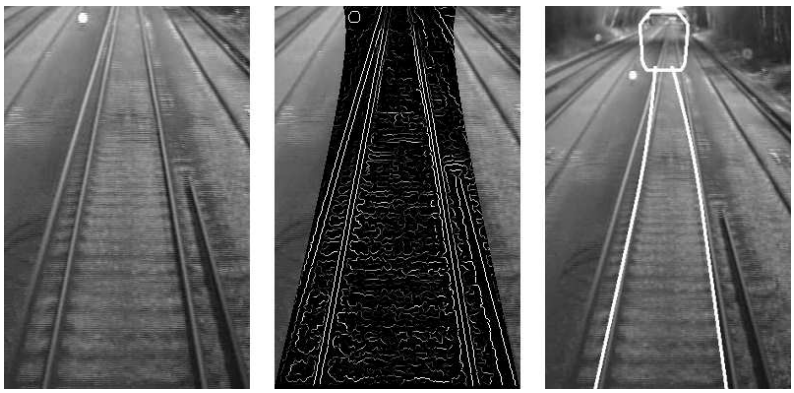

Abbildung B.13: Teilweise Abweichung vom Optimum durch parallele Schienen

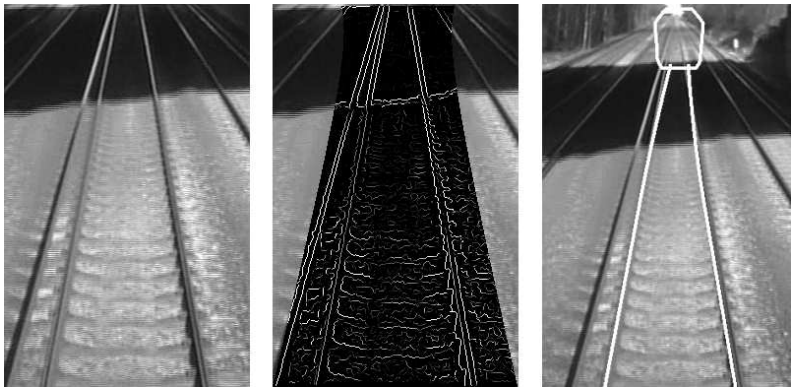

Abbildung B.14: Erkennung beim Unterfahren einer Brücke

B.3 Fehl-Erkennungen

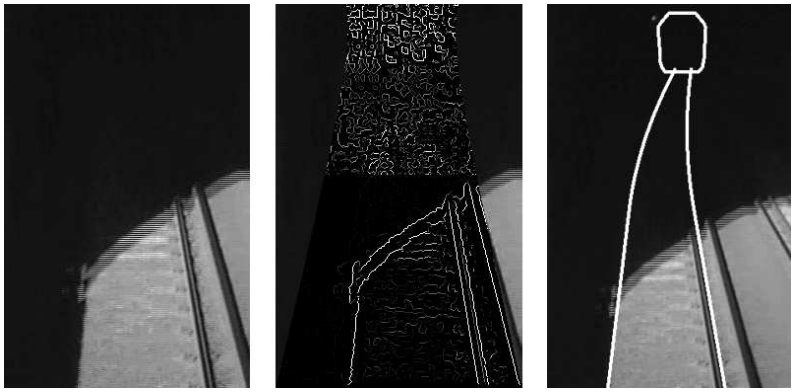

Abbildung B.15: Keine Erkennung beim Übergang in den Tunnel

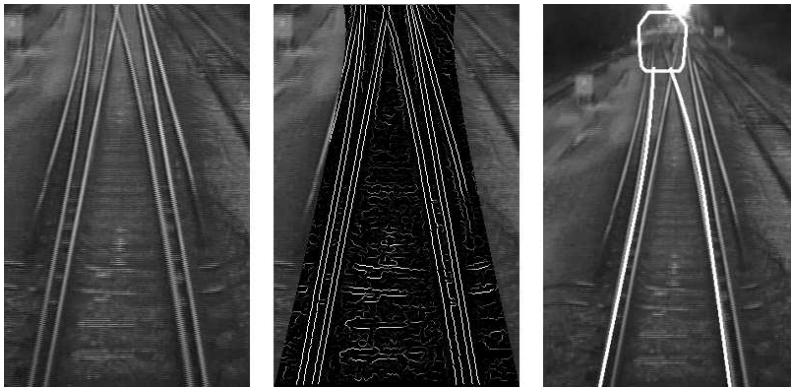

Abbildung B.16: Temporäre Fehlerkennung bei Weichen

C Klassenverteilung auf einzelne Dateien

Dateiname	Enthaltene Klassen
Calibration.cs	`Calibration, Point2D, Point3D, PassPoint`
Complex.cs	`Complex`
DxPlay.cs	`DxPlay`
Filter.cs	`Filter (abstract)`
FormDemoApp.cs	`FormDemoApp`
FormStartProcessing.cs	`FormStartProcessing`
GradientFilter.cs	`GradientFilter, GradientFilters`
HistOps.cs	`HistOps`
HoughTransform.cs	`HoughTransform`
ImageComplex.cs	`ImageComplex`
ImageGray.cs	`ImageGray`
ImageLimiter.cs	`ImageLimiter`
Line.cs	`Line, HoughLine, CandidateLines, WinningCandidate`
LinearFilter.cs	`LinearFilter, LinearFilters, AdaptiveGaussFilter`
MathCalc.cs	`MathCalc`
OptimalEdgeFilters.cs	`MarrHildrethFilter, CannyFilter, ShenCastanFilter, OptimalEdgeFilters`
ParabolaModel.cs	`ParabolaModel (abstract), Parabola, LinearParabola`
RailProcessing.cs	`RailProcessing`

Tabelle C.1: Klassenverteilung

D XML-Strukturen

D.1 XML-Struktur für Kalibrierungsdaten

Die Kalibrierungsdaten des Videos werden üblicherweise nicht direkt mit der Schienenerkennungs-Applikation erstellt, sondern über vorhandene Entwicklungen der Firma CCC GmbH ermittelt. Daher wird ein bestehendes XML-Schema für kalibrierte Punkte nach folgender Struktur verwendet:

```
<?xml version="1.0"?>
<CCCMeasure schemaVersion="1.0">
  <measureData schemaVersion="1.0">
    <calibrations>
      <calibration name="Videobeschreibung">
        <calibData unit="0">
          <point info="Punktbeschreibung">
            <world x="..." y="..." z="..."/>
            <img x="..." y="..."/>
          </point>
          <point ...>
            ...
          </point>
          ...
        </calibData>
      </calibration>
    </calibrations>
  </measureData>
</CCCMeasure>
```

`<calibrations>` enthält eine Liste von Kalibrierungen, von denen durch das Programm die erste verwendet wird. Unter `<calibration>\<calibData>` sind dann die eigentlichen Kalibrierungspunkte in den `<point>`-Tags enthalten. Diese umfassen einen Passpunkt, der sich über `<world>` als Weltpunkt und `<img>` als Bildpunkt definiert. Für die DLT sind wie in Kapitel 2 besprochen mindestens 6 solcher Passpunkte erforderlich. Es sollte erwähnt werden, dass die zu übergebenden Bildpunkte auf eine Bildbreite von 640 Pixeln normiert sein müssen. Über das Attribut `unit`

des `<calibData>`-Tags lässt sich zudem die Maßeinheit der Weltpunkte festlegen: 0=keine (mm), 1=cm, 2=dm, 3=m, 4=km, 5=höchste Maßeinheit (hier: km).

D.2 XML-Struktur für Lichtraumprofile

Hier wird ein einfacher Linienzug definiert, der die Größe des Lichtraumprofils im Weltkoordinatensystem angibt. Bei einer frontalen Maske gilt stets $y = 0$, wie es der Normalfall sein sollte. Der Punkt $(0, 0, 0)$ stellt den mittleren Fußpunkt des Lichtraumprofils dar, der am Ende in der Mitte zwischen den erkannten Schienen aufsetzen soll. Folgende Struktur kommt zum Einsatz:

```
<?xml version="1.0"?>
<clearanceDiagram  schemaVersion="1.0">
  <linePath unit="0">
    <point x="..." y="..." z="..." />
    <point ... />
    ...
  </linePath>
</clearanceDiagram>
```

Das `<linepath>`-Tag kann ein Attribut `unit` enthalten, welches wie in D.1 die Maßeinheit der Weltpunkte angibt. Standard ist dabei *mm*. Es folgt eine Liste von `<point>`-Elementen, welche die einzelnen Punkte des Lichtraumprofils darstellen. Im Programm wird der Linienzug geschlossen, daher muss der letzte Punkt *nicht* mit dem ersten übereinstimmen, eine Gerade zwischen beiden wird automatisch erzeugt.

E Inhalt der CD

Neben der ausführbaren Software und der CD-Inhaltsbeschreibung im Stammverzeichnis sind folgende Ordner vorhanden:

Bachelorarbeit:	Enthält dieses Dokument als PDF-Datei.
DirectShow .NET:	Quellcodes, Beispiele und Tutorials zu DirectShow .NET v1.2.
Dokumentation:	Dokumentation der Testsoftware.
Hilfsprogramme:	Benötigte und optionale Software: .NET Framework 2.0, DirectX 9.0c, AviSynth, VirtualDubMod.
Literatur:	Verwendete Literatur, die im PDF-Format zugänglich war.
Quelltext:	Enthält den Quelltext der entwickelten Software.
Testmaterial:	Enthält das Testvideo als WMV-Datei sowie seine Kalibrierung und das Lichtraumprofil als XML-Datei.

Literaturverzeichnis

[1] *Anlagen zur Eisenbahn-Bau- und Betriebsordnung.* `http://www.wedebruch.de/gesetze/betrieb/eboanl.htm`, letzter Zugriff: 13.09.2006

[2] *DirectShow .NET.* `http://directshownet.sourceforge.net/index.html`, letzter Zugriff: 10.06.2006

[3] *Eisenbahn-Bau- und Betriebsordnung.* `http://www.wedebruch.de/gesetze/betrieb/ebo1.htm`, letzter Zugriff: 13.09.2006

[4] *Photogrammetrie.* `http://de.wikipedia.org/wiki/Photogrammetrie`, letzter Zugriff: 10.04.2006

[5] Canny, J.F.: A computational approach to edge detection. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8 (1986), S. 679–698

[6] Cláudio Rosito Jung, Christian Roberto K.: A Robust Linear-Parabolic Model for Lane Following. In: *Proceedings of SIBGRAPI 2004 (17th Brazilian Symposium on Computer Graphics and Image Processing)* (2004), S. 72–79

[7] Haberäcker, Peter: *Praxis der Digitalen Bildverarbeitung und Mustererkennung.* Carl Hanser Verlag München Wien, 1995. – ISBN 3–446–15517–1

[8] Jain, Anil K.: *Fundamentals Of Digital Image Processing.* Prentice-Hall, Inc., 1989. – ISBN 0–13–336165–9

[9] John B. McDonald, Simon M. Charles Markham M. Charles Markham: Selected Problems in Automated Vehicle Guidance / Signals and Systems Group, Department of Computer Science, National University of Ireland, Maynooth, Ireland. Version: 2001. `http://citeseer.csail.mit.edu/502502.html`. 2001 (NUIM/SS/01/05). – Forschungsbericht

[10] Klette, Schlüns Koschan: *Computer Vision - Räumliche Information aus digitalen Bildern.* Vieweg Verlag, 1996. – ISBN 3–528–06625–3

[11] Kraus, Karl: *Photogrammetrie - Band 2: Verfeinerte Methoden und Anwendungen.* 3. Auflage. Ferd. Dümmerls Verlag, Bonn, 1996. – ISBN 3–427–78653–6

[12] Kraus, Karl: *Photogrammetrie - Band 1: Grundlagen und Standardverfahren.* 6. Auflage. Ferd. Dümmerls Verlag, Bonn, 1997. – ISBN 3–427–78646–3

[13] Lacmann, Otto: *Die Photogrammetrie in ihrer Anwendung auf nicht-topographischen Gebieten.* S. Hirzel Verlag Leipzig, 1989

[14] Massima Bertozzi, et. a. Alberto Broggi B. Alberto Broggi: Artificial Vision in Road Vehicles. In: *Proceedings of the IEEE* 90 (2002), 1258–1271. `http://citeseer.csail.mit.edu/614403.html`

[15] Mühlmann, Karsten: *Design und Implementierung eines Systems zur schnellen Rekonstruktion dreidimensionaler Modelle aus Stereobildern*, Universität Mannheim, Dissertation, 2002. `http://bibserv7.bib.uni-mannheim.de/madoc/volltexte/2002/56/pdf/56_1.pdf`

[16] Moré, John: *Direkte Lineare Transformation (DLT).* `http://www.fpk.tu-berlin.de/~john/pdf/dlt.pdf`, letzter Zugriff: 10.04.2006

[17] Parker, J.R.: *Algorithms for image processing and computer vision.* John Wiley and Sons Inc., 1997. – ISBN 0–471–14056–2

[18] Rafael C. Gonzalez, Richard E. W.: *Digital Image Processing.* Addison-Wesley Publishing Company, 1992. – ISBN 0–201–50803–6

[19] Regensburger, Karl: *Photogrammetrie.* VEB Verlag für Bauwesen, Berlin, 1990. – ISBN 3–345–00461–5

[20] Rüger, Regensburger Pietschner: *Photogrammetrie - Verfahren und Geräte zur Kartenherstellung.* 5. Auflage. VEB Verlag für Bauwesen, Berlin, 1987. – ISBN 3–345–00196–9

[21] Schwidefsky, Ackermann: *Photogrammetrie - Grundlagen, Verfahren, Anwendungen.* 7. Auflage. P.G. Teubner Stuttgart, 1976. – ISBN 3–519–13401–2

[22] Steinbrecher, Rainer: *Bildverarbeitung in der Praxis.* R. Oldenburg Verlag GmbH München, 1993. – ISBN 3–489–22372–0

[23] Torre, V. ; Poggio, T. A.: On Edge Detection. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8 (1986), 147–163. `http://citeseer.ist.psu.edu/torre84edge.html`

[24] Tsai, Roger Y.: A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. In: *IEEE Journal of Robotics and Automation* (August 1987), S. 323–344

[25] Vetterlein, Stefan: *Photogrammetrische Messungen in Bildern und Bildfolgen*, HTWK Leipzig, Diplomarbeit, 2004

[26] Wilhelm Burger, Mark James B.: *Digitale Bildverarbeitung - Eine Einführung mit Java und ImageJ*. 2., überarbeitete Auflage. Springer-Verlag Berlin Heidelberg, 2006. – ISBN 3–540–30940–3

[27] Yue Wang, Dinggang S. ; Teoh, Eam K.: Lane Detection Using Catmull-Rom Spline. In: *Proceedings of Intelligent Vehicles 1998 Symposium* (1998), 51–57. `https://www.rad.upenn.edu/sbia/dgshen/papers/IV98.pdf`

[28] Zhang, Zhengyou: A Flexible New Technique for Camera Calibration / Microsoft Research. 1998 (MSR-TR-98-71). – Forschungsbericht

[29] Ziou, D. ; Tabbone, S.: Edge Detection Techniques - An Overview. In: *International Journal of Pattern Recognition and Image Analysis* 8 (1998), 537–559. `http://citeseer.ist.psu.edu/ziou98edge.html`

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Leipzig, den 28.09.2006

Matthias Jauernig