Prüfer:	Prof. Dr. rer. nat. habil. Paul Levi
Betreuer:	Dipl.-Inform. Susanne Gerl Dipl.-Inform. Norbert Oswald

begonnen am:	01. Dez 1995
beendet am:	31. Mai 1996

CR-Klassifikation:	I.2.10, I.4.8, I.2.9, I.3.3

Diplomarbeit Nr. 1355

Robuste Extraktion sich bewegender Objekte aus einer mit dynamischer Kamera aufgenommenen Videosequenz

Marc Ebner

Fakultät Informatik
Institut für Parallele und
Verteilte Höchstleistungsrechner
Universität Stuttgart
Breitwiesenstraße 20–22
D–70565 Stuttgart

Zusammenfassung

In der vorliegenden Arbeit wurde ein robuster Ansatz, bewegte Objekte zu extrahieren, entwickelt. Schwingungen der Kamera und ungenaue Daten zur Berechnung der Eigenbewegung werden toleriert. Die Motivation zur Extraktion bewegter Objekte wird am Anfang der Arbeit vorgestellt. Die verschiedenen aus der Literatur bekannten Ansätze zur Extraktion bewegter Objekte und zur Detektion von Veränderungen aus einer Videosequenz werden besprochen. Der Ansatz, die Eigenbewegung der Kamera zu kompensieren und Veränderungen durch ein Differenzbild zu detektieren, wurde gewählt.

Das betrachtete Bild wird in der vorliegenden Arbeit als Fläche mit konstanter Entfernung zur Kamera modelliert. Die Entfernung zu den Objektpunkten wird durch Korrelation markanter Punkte abgeschätzt. Das aktuelle Bild wird aufgrund des vorangegangenen Bildes und der bekannten Transformationsmatrix vorhergesagt. Die Transformation wird aus den Statusinformationen der Kamera und des verwendeten Fahrzeuges berechnet. Die Modellierung der Kamera und des Fahrzeuges wird vorgestellt. Ein lineares Fehlermodell wurde eingeführt, um Schwingungen zu eliminieren. Durch Korrelation markanter Punkte wird ein Verschiebungsvektor berechnet, der die Prädiktion verbessert. Das Differenzbild aus der verbesserten Prädiktion des aktuellen Bildes und des tatsächlichen Bildes liefert Bildpunkte, an denen eine Veränderung stattgefunden hat. Verbleibende Störungen oder Fehler in der Kompensation der Eigenbewegung werden durch die Anwendung morphologischer Operatoren eliminiert.

Nun wird wie auf einer stationären Bildsequenz weitergearbeitet. Die detektierten Veränderungen werden zu bewegten Objekten zusammengefaßt. Eine zweite Literaturrecherche wurde betrieben, um nach geeigneten Verfahren zu suchen. Die Verfahren werden beschrieben und bewertet. Aus den vorgestellten Verfahren wurde ein Algorithmus entwickelt, Veränderungen des Differenzbildes zu Objekten zusammenzufassen. Es wird eine Heuristik eingeführt, um Bereiche, die durch die Anwendung morphologischer Operationen getrennt wurden, zu einem Objekt zusammenzufassen. Zwischen allen Regionen des vorangegangenen und des aktuellen Differenzbildes wird aufgrund des Vergleiches der Flächen der Regionen eine Korrespondenz hergestellt und der Bewegungsvektor berechnet. So ergeben sich neue Bewegungshypothesen, die mit den alten Bewegungshypothesen kombiniert werden. Sind die Bewegungshypothesen richtig, das heißt, es wird eine Veränderung im folgenden Differenzbild an der vermuteten Stelle gefunden, so wird die Hypothese validiert und es entsteht ein bewegtes Objekt. Die extrahierten bewegten Objekte werden dann ebenfalls eingesetzt, um Bereiche des Differenzbildes zu Objekten zusammenzufassen.

Die Benutzung des entwickelten Programms und der Bibliothek wird beschrieben. Die Grenzbereiche der Implementierung werden anhand einiger Videosequenzen analysiert. Laufzeitmessungen für verschiedene Bildgrößen werden ebenfalls gegeben. Schließlich wird die Ausgabe der Implementierung für einige Videosequenzen präsentiert.

Vorwort

Die vorliegende Arbeit wurde von der Friedrich-Naumann-Stiftung aus Mitteln des Bundesministeriums für Bildung und Wissenschaft gefördert. Ich möchte der Friedrich-Naumann-Stiftung an dieser Stelle ganz herzlich für die materielle und vor allem für ihre ideelle Förderung danken, die sie mir über ein Stipendium zukommen ließ.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

Die in der vorliegenden Arbeit verwendeten Symbole sind im Symbolverzeichnis in der Reihenfolge ihres Auftretens sortiert. So werden Symbole, die innerhalb desselben Kontextes verwendet werden, nicht auseinander gerissen. Programmvariablen und Symbole, die in Schreibmaschinenschrift gesetzt sind, sind nicht im Symbolverzeichnis eingetragen.

Kapitel 2

$\mathbf{P}_O$	Objektpunkt mit Koordinaten $[X, Y, Z]^T$
$\mathbf{P}_B$	Bildpunkt mit Koordinaten $[x, y, f]^T$
$\mathbf{P}_U$	Koordinatenursprung mit Koordinaten $[0, 0, 0]^T$
$\mathbf{P}_f$	Punkt im Zentrum der Bildebene mit Koordinaten $[0, 0, f]^T$
$\mathbf{P}_Z$	Projektion des Objektpunktes auf die Z-Achse des Koordinatensystems. Die Koordinaten sind $[0, 0, Z]^T$.
f	Brennweite
X	X-Koordinate eines Weltpunktes
Y	Y-Koordinate eines Weltpunktes
Z	Z-Koordinate eines Weltpunktes
x	X-Koordinate eines Bildpunktes
y	Y-Koordinate eines Bildpunktes
$\triangle\mathbf{P}_U\mathbf{P}_O\mathbf{P}_Z$	Dreieck, das durch die Punkte $\mathbf{P}_U$, $\mathbf{P}_O$ und $\mathbf{P}_Z$ beschrieben wird.
$\triangle\mathbf{P}_U\mathbf{P}_B\mathbf{P}_f$	Dreieck, das durch die Punkte $\mathbf{P}_U$, $\mathbf{P}_B$ und $\mathbf{P}_f$ beschrieben wird.
m	Skalierungsfaktor der orthogonalen Projektion. Mit m wird auch die Anzahl der Punkte eines Polygons bezeichnet.
$\bar{Z}$	Durchschnittliche Entfernung der Objektpunkte zum Hintergrund
$\mathbf{V}_O$	Geschwindigkeit des Objektpunktes
$\mathbf{V}_B$	Geschwindigkeit des Bildpunktes
$\hat{\mathbf{Z}}$	Einheitsvektor in Z-Richtung
t	Zeitpunkt
δt	kleine Zeitdifferenz
$E(x, y, t)$	Bestrahlungsstärke am Bildpunkt (x, y) zum Zeitpunkt t
u	X-Komponente des optischen Flusses
v	Y-Komponente des optischen Flusses
$u(x, y)$	X-Komponente des optischen Flusses im Punkt (x, y) Das selbe Sysmbol wird auch zur Bezeichnung des Realteils der Position ω eingesetzt.
$v(x, y)$	Y-Komponente des optischen Flusses im Punkt (x, y) Das selbe Sysmbol wird auch zur Bezeichnung des Imaginärteils der Position ω eingesetzt.

E_x	Partielle Ableitung von E nach x
E_y	Partielle Ableitung von E nach y
E_t	Partielle Ableitung von E nach t
∇E	Gradient von E
$\bar{u}(x,y)$	X-Komponente des durchschnittlichen optischen Flusses der Nachbarpunkte von (x,y)
$\bar{v}(x,y)$	Y-Komponente des durchschnittlichen optischen Flusses der Nachbarpunkte von (x,y)
λ	Gewichtungsfaktor
n	Zählvariable
B	Bild
$B(x,y,t)$	Grauwert des Bildes B an der Position (x,y) zum Zeitpunkt t
τ	Schwellwert
t_1, t_2	Zeitpunkte
DB	Differenzbild
$DB_{t_1,t_2}(x,y)$	Wert des Differenzbildes zweier Bilder, die zu den Zeiten t_1 und t_2 entstanden sind, an der Position (x,y)
PDB	Positives Differenzbild
$PDB_{t_1,t_2}(x,y)$	Wert des positiven Differenzbildes zweier Bilder, die zu den Zeiten t_1 und t_2 entstanden sind, an der Position (x,y)
NDB	Negatives Differenzbild
$NDB_{t_1,t_2}(x,y)$	Wert des negativen Differenzbildes zweier Bilder, die zu den Zeiten t_1 und t_2 entstanden sind, an der Position (x,y)
$AADB$	Angesammeltes, absolutes Differenzbild
$AADB_n(x,y)$	Wert des n-ten angesammelten, absoluten Differenzbildes zweier Bilder, die zu den Zeiten t_1 und t_2 entstanden sind, an der Position (x,y)
$APDB$	Angesammeltes, positives Differenzbild
$APDB_n(x,y)$	Wert des n-ten angesammelten, positiven Differenzbildes zweier Bilder, die zu den Zeiten t_1 und t_2 entstanden sind, an der Position (x,y)
$ANDB$	Angesammeltes, negatives Differenzbild
$ANDB_n(x,y)$	Wert des n-ten angesammelten, negativen Differenzbildes zweier Bilder, die zu den Zeiten t_1 und t_2 entstanden sind, an der Position (x,y)
BK	Bild mit bewegten Kanten
$BK(x,y,t)$	Pixelwert des Bildes mit bewegten Kanten
∇B	Gradientenbild
LB	Zweite zeitliche Ableitung des Bildes B
$LB(x,y,t)$	Pixelwert der zweiten zeitlichen Ableitung des Bildes B
C	Operator, der die Eckigkeit der betrachteten Objektpunkte ermittelt
$B(t)$	Bild zum Zeitpunkt t
R	Region
$R_{x,y}$	Region um den Punkt (x,y), definiert durch die Bildpunkte der Region
$\mu_{x,y,t}$	Mittelwert der um den Punkt (x,y) liegenden Grauwerte zum Zeitpunkt t
$\sigma^2_{x,y,t}$	Varianz der um den Punkt (x,y) liegenden Grauwerte zum Zeitpunkt t
H_0	Hypothese, daß die Regionen zum selben Objekt gehören
H_1	Hypothese, daß die Regionen zu verschiedenen Objekten gehören

$g(x, y, t)$	Grauwerte der Region um (x, y) zur Zeit t
$p(g(x, y, t)\|H_0)$	Wahrscheinlichkeit, daß die Grauwerte durch $g(x, y, t)$ gegeben sind, falls die Hypothese H_0 zutrifft.
$p(g(x, y, t)\|H_1)$	Wahrscheinlichkeit, daß die Grauwerte durch $g(x, y, t)$ gegeben sind, falls die Hypothese H_1 zutrifft.
a_j	Parameter der Grauwertfunktion einer Region $j \in \{1, .., 5\}$
NG	Bild mit normalisierten Grauwerten
$NG(x, y, t)$	Pixelwerte des Bildes mit normalisierten Grauwerten
D	Differenzmetrik
$D_{t_1,t_2}(x, y)$	Differenzwert am Punkt (x, y) zweier Bilder, die zum Zeitpunkt t_1 und t_2 aufgenommen wurden.
$I(t)$	Beleuchtung zum Zeitpunkt t
$S(x, y)$	Schattierungskoeffizient des Objektes am betrachteten Bildpunkt (x, y)
μ_{x,y,t_1,t_2}	Erwartungswert der Intensitätsverhältnisse der Grauwerte der Region um den Punkt (x, y) zweier Bilder, die zum Zeitpunkt t_1 und t_2 aufgenommen wurden.
FOE	Zentrum der Expansion (*Fokus of Expansion*)
(x_{FOE}, y_{FOE})	Koordinaten des Zentrums der Expansion
$[dx, dy, dz]^T$	Translation des Beobachters
EB	Eigenbewegungsbild, das durch die Eigenbewegung-Polar-Transformation entsteht
$EB(r, \theta, t)$	Pixelwert des Eigenbewegungsbildes EB mit den Koordinaten (r, θ) das zum Zeitpunkt t aufgenommen wurde
r	Abstand eines Bildpunktes zum Zentrum der Expansion
θ	Winkelposition, an der sich ein Bildpunkt befindet
$r(x, y)$	Abstand des Bildpunktes (x, y) zum Zentrum der Expansion
$\theta(x, y)$	Winkelposition, an der sich der Bildpunkt (x, y) befindet
z	Komplexe Zahl
i	$\sqrt{-1}$ (nur in Kapitel 2 steht i für $\sqrt{-1}$ sonst ist i eine Zählvariable)
$\omega(r, \theta)$	Position des Bildpunktes (r, θ) im komplex-logarithmischen Raum
$u(r, \theta)$	Realteil der Position $\omega(r, \theta)$
$v(r, \theta)$	Imaginärteil der Position $\omega(r, \theta)$
$(x(t), y(t))$	Position eines Bildpunktes zum Zeitpunkt t
α	Kleiner Drehwinkel um die Schwenkachse
γ	Kleiner Drehwinkel um die Nickachse

Kapitel 3

$\Theta(\nabla B)$	Richtung des Gradienten am Punkt (x, y) im Bild B
τ_1, τ_2, τ_3, τ_4	Schwellwerte
$v_{x,y}(\theta_v)$	Betrag der Geschwindigkeit eines Bildpunktes (x, y) in Abhängigkeit von der Richtung der Geschwindigkeit
θ_v	Richtung der Geschwindigkeit eines Bildpunktes
$(\hat{v}, \theta_{\hat{v}})$	Maxima im Parameterraum der Hough-Transformation
(x_0, y_0)	Punkt des Hintergrundes
$b(x, y)$	Notation die angibt, daß der Punkt (x, y) zum Hintergrund gehört

N	Nachbarschaftsrelation
$N(x,y)$	Punkte in der Nachbarschaft von Punkt (x,y)
G	Gradientenbild
G_t	Zeitliche Ableitung des Gradientenbildes
$p(b(x,y),B)$	Wahrscheinlichkeit, daß der Punkt (x,y) des Bildes B zum Hintergrund gehört.
$p(b(x,y)\|G_t(x,y), b(N(x,y)))$	Wahrscheinlichkeit, daß der Punkt (x,y) zum Hintergrund gehört, wenn bekannt ist, daß ein Punkt aus der Nachbarschaft von (x,y) zum Hintergrund gehört.
τ, μ	Parameter zur Approximation des Logarithmus der Wahrscheinlichkeit $p(b(x,y)\|G_t(x,y), b(N(x,y)))$
Typ O	Region, die durch das Bedecken des Hintergrundes mit dem Objekt entsteht.
Typ B	Region, die durch das Freiwerden des Hintergrundes mit dem Objekt entsteht.
Typ X	Region, die sowohl durch das Bedecken des Hintergrundes als auch durch das Freiwerden des Hintergrundes entsteht.
CC	Anzahl der Punkte auf der äußeren Kontur einer Region des Differenzbildes DB, die zugleich Kantenpunkte im aktuellen Bild sind.
CP	Anzahl der Punkte auf der äußeren Kontur einer Region des Differenzbildes DB, die zugleich Kantenpunkte im vorhergehenden Bild sind.

Kapitel 4

$\hat{\mathbf{X}}_\mathbf{i}$	Einheitsvektor des Koordinatensystems $\{i\}$ in X-Richtung $i \in \{R,0,1,2,3,4,C,W\}$
$\hat{\mathbf{Y}}_\mathbf{i}$	Einheitsvektor des Koordinatensystems $\{i\}$ in Y-Richtung $i \in \{R,0,1,2,3,4,C,W\}$
$\hat{\mathbf{Z}}_\mathbf{i}$	Einheitsvektor des Koordinatensystems $\{i\}$ in Z-Richtung $i \in \{R,0,1,2,3,4,C,W\}$
${}^iP$	Punkt im Koordinatensystem $\{i\}$ $(i \in \{R,0,1,2,3,4,C,W\})$
T	Transformationsmatrix
${}^i_jT$	Transformation von Koordinatensystem $\{j\}$ auf das Koordinatensystem $\{i\}$ $(i,j \in \{R,0,1,2,3,4,C,W\})$
r_{ij}	Rotationsmatrix der Transformationsmatrix $(i,j \in \{1,2,3\})$
ϱ_x, ϱ_y, ϱ_z	Translationsvektor der Transformationsmatrix
$Z(x,y)$	Entfernung des auf den Punkt (x,y) abgebildeten Punktes
a,b	Parameter der Abstandsfunktion $Z(x,y)$
h	Höhe des Bildes in Pixel
I_1, I_2, I_3, I_4	Grauwertvarianzen
$\breve{B}(t)$	Markante Punkte des Bildes $B(t)$
(x_1,y_1), (x_2,y_2)	Bildpunkte
(x_T,y_T)	Bildpunkt, der aufgrund der Transformationsgleichung T transformiert wurde
Δx	Horizontale Verschiebung des Bildes
Δy	Vertikale Verschiebung des Bildes

$(x_{T_\Delta}, y_{T_\Delta})$	Bildpunkt, der aufgrund der Transformationsgleichung T transformiert und durch die Fehlerkorrektur verschoben wurde
$B_{T_\Delta}(t)$	Bild, das aufgrund der Transformationsgleichung T transformiert und durch die Fehlerkorrektur verschoben wurde
w, h	Größe einer rechteckigen Maske in Pixel
M	Morphologisches Strukturelement
$\ominus$	Morphologische Operation Erosions
$\oplus$	Morphologische Operation Dilation
$\circ$	Morphologische Operation Opening
$\bullet$	Morphologische Operation Closing

Kapitel 5

a_i	Kürzeste Entfernung zwischen $\hat{\mathbf{Z}}_i$ und $\hat{\mathbf{Z}}_{i+1}$ gemessen in Richtung von $\hat{\mathbf{X}}_i$
α_i	Winkel zwischen $\hat{\mathbf{Z}}_i$ und $\hat{\mathbf{Z}}_{i+1}$ gemessen um die Drehachse $\hat{\mathbf{X}}_i$
d_i	Entfernung von $\hat{\mathbf{X}}_i$ und $\hat{\mathbf{X}}_{i+1}$ gemessen in Richtung von $\hat{\mathbf{Z}}_i$
θ_i	Winkel zwischen $\hat{\mathbf{X}}_{i-1}$ und $\hat{\mathbf{X}}_i$ gemessen um die Drehachse $\hat{\mathbf{Z}}_i$
$\{R\}$	Koordinatensystem des Fahrzeuges
$\{0\}$	Koordinatensystem der Kamerabasis
$\{1\}$	Koordinatensystem der Kameraachse 1
$\{2\}$	Koordinatensystem der Kameraachse 2
$\{3\}$	Koordinatensystem der Kameraachse 3
$\{4\} = \{C\}$	Koordinatensystem der Kamera
$\{W\}$	Koordinatensystem der Welt
L_1, L_2, L_3, L_4	Kameraabmessungen
b_x, b_y, b_z	Position der Kamerabasis relativ zum Koordinatensystem des Fahrzeuges
$\theta'_1, \theta'_2, \theta'_3$	Bei der Berechnung der Transformationsmatrix bezeichnen die gestrichenen Winkel die Gelenkpositionen der Kamera zum Zeitpunkt t_2, die ungestrichenen Winkel die Positionen zum Zeitpunkt t_1
v	Lineare Geschwindigkeit des Fahrzeuges
w	Winkelgeschwindigkeit des Fahrzeuges
$R_X(\theta)$	Rotationsmatrix für Drehung um X-Achse um den Winkel θ
$R_Y(\theta)$	Rotationsmatrix für Drehung um Y-Achse um den Winkel θ
$R_Z(\theta)$	Rotationsmatrix für Drehung um Z-Achse um den Winkel θ
$D_X(x)$	Translationsmatrix für Translation in X-Richtung um x
$D_Y(y)$	Translationsmatrix für Translation in Y-Richtung um y
$D_Z(z)$	Translationsmatrix für Translation in Z-Richtung um z
d_x	Strecke, die das Fahrzeug in der Zeit δt zurücklegt
β	Winkel, um den sich das Fahrzeug in der Zeit δt dreht
Θ	Komplexitätsmaß
$BR(t)$	Bewegte Regionen zum Zeitpunkt t
$BH(t)$	Bewegungshypothesen zum Zeitpunkt t
$BO(t)$	Bewegte Objekte zum Zeitpunkt t
R	Bewegte Region
P	Polygon
(s_x, s_y)	Schwerpunkt

(s_x, s_y)	Schwerpunkt
(v_x, v_y)	Bewegungsvektor
A_R	Fläche der Region
$B_T(t)$	Bild $B(t)$ das durch die Transformation T transformiert wurde
$\dot{\theta}_i(t)$	Geschwindigkeiten der Gelenke $i \in \{1, 2, 3\}$ zum Zeitpunkt t
c	Korrelationswert zweier Regionen um markante Punkte
w_B, h_B	Breite und Höhe eines Bildes B
n_1, n_2	Zählvariablen
Z_x	Z-Koordinate eines Objektpunktes berechnet aufgrund seiner Verschiebung in x-Richtung
Z_y	Z-Koordinate eines Objektpunktes berechnet aufgrund seiner Verschiebung in y-Richtung
$(\hat{x}, \hat{y})$	Koordinaten des durch T transformierten Bildpunktes (x_1, y_2), wenn sich der zugehörige Objektpunkt im Abstand von 1m von der Kamera befunden hat
$(\tilde{x}, \tilde{y})$	Koordinaten des durch T transformierten Bildpunktes (x_1, y_2), wenn die translatorische Bewegung der Kamera vernachlässigt wird
L_B	Liste mit Abstandswerten markanter Punkte von Bild B zur Kamera
L_z	Liste mit Abstandswerten vorhergehender Bilder
l	Anzahl der Werte in der Liste L_z
L_x	Liste mit Differenzen korrespondierender markanter Punkte in x-Richtung
L_y	Liste mit Differenzen korrespondierender markanter Punkte in y-Richtung
$\beta(B)$	Bild mit Konturen der Regionen in Bild B
K	Menge der äußeren Konturen eines Bildes mit Regionen
A, B	Konturen zweier Regionen
nahe(A, B)	Prädikat, das angibt, ob die beiden Regionen A und B nahe beieinanderliegen
$\sim'$	Relation naher Regionen
R_A, R_B	Punktmenge des Bereiches, der von der Kontur A bzw. B umschlossen wird
$\sim$	Reflexive und transitive Hülle der Relation $\sim$
$K/\sim$	Äquivalenzklassen von K, die durch die Relation $\sim$ erzeugt werden
O	Objekt
$BR_{T_\Delta}(t)$	Bewegte Regionen zum Zeitpunkt t, die durch T transformiert und anschließend verschoben wurden
R_1, R_2	Bewegte Regionen
A_1, A_2	Flächen der bewegten Regionen
$\|\|P\|\|$	Anzahl der Punkte des Polygons P
O_b	Bewegtes Objekt
O_h	Bewegungshypothese
H	Bewegungshypothese
$BO_{T_\Delta}(t)$	Bewegte Objekte, die auf ihre Position zum Zeitpunkt t gebracht wurden, dann durch T transformiert und anschließend verschoben wurden
$BH_{T_\Delta}(t)$	Bewegungshypothesen, die auf ihre Position zum Zeitpunkt t gebracht wurden, dann durch T transformiert und anschließend verschoben wurden

Kapitel 1

Einleitung

Zunächst wird hier erklärt, welche Motivation hinter dem Vorhaben, bewegte Objekte aus einer Bildsequenz extrahieren, steckt. Im Anschluß daran wird erklärt, wie die vorliegende Arbeit gegliedert ist, und warum sie in der folgenden Form gegliedert ist.

1.1 Motivation

Warum ist es sinnvoll, bewegte Objekte aus einer bewegten Umgebung zu extrahieren? Sobald ein Roboter sich gemeinsam in einem Umfeld mit anderen Robotern oder Menschen bewegt, ist der Roboter gezwungen auf seine Umgebung zu reagieren. Dies soll an einem einfachen Beispiel verdeutlicht werden. Ein mobiler Roboter könnte die Zustellung von Post innerhalb eines Gebäudes übernehmen. Der Roboter muß also an einem Ort die Post aufnehmen und den Empfängern zustellen. In der Regel wird solch ein Roboter dieselben Gänge wie die Menschen benutzen. Vor allem zum Schutz der Menschen ist es wichtig, daß sie nicht von einem mobilen Fahrzeug angefahren werden. Der Roboter sollte Kollisionen vermeiden und braucht dazu die Information der Objekte in seiner Umgebung, die sich selbst bewegen.

Mobile Fahrzeuge werden in vielen Bereichen eingesetzt. In Fabriken liefern sie zum Beispiel Bauteile an einen Arbeitsplatz [McKerrow 91]. Diese Roboter haben jedoch nicht nur ihre Aufgabe zu erfüllen, sondern sie müssen auch auf eine sich ständig verändernde Umwelt reagieren. Dies ist umso nötiger, je stärker die Fahrzeuge aus einem kontrollierten Umfeld herausgenommen werden, und sich in einer alltäglichen Umgebung zurechtfinden müssen. Kollisionen mit anderen bewegten Objekten aus der Umgebung müssen vermieden werden.

Bei einem Spaziergang durch eine belebte Fußgängerzone kann leicht überprüft werden, wie gut wir Menschen diese Fähigkeit besitzen. Auch auf einer belebten Fußgängerzone sind wir in der Lage, uns in die gewünschte Richtung zu bewegen, ohne dabei gegen andere Menschen zu stoßen. Um ein Fahrzeug mit ähnlichen Fähigkeiten auszustatten, ist es noch ein weiter Weg. Als ersten Schritt in diese Richtung müssen die Objekte aus der Umwelt bestimmt werden, die sich selbst bewegen.

Ihre Umwelt nehmen mobile Roboter über Sensoren wahr [Jones et al. 93]. Mobile Roboter sind in der Regel mit den verschiedensten Sensoren ausgestattet. Mit Ultraschall-Sensoren kann zum Beispiel der Abstand zu einem Hindernis gemessen werden. Doch diese Information kann aufgrund von Reflexionen sehr ungenau sein [McKerrow 91]. Mit einer Kamera ausgestattete Roboter sind in der Lage ein Bild ihrer Umwelt aufzunehmen und daraus eine Vielzahl von Informationen zu extrahieren.

Bereits ein einzelnes Bild liefert eine Fülle von Informationen [Marr 82]. Kanten können extrahiert werden und zu Objekten zusammengefaßt werden. Die Objekte im Bild können zum Beispiel aufgrund ihrer Form bekannten Objekten zugeordnet und somit erkannt werden. Eine Bildsequenz liefert jedoch weit mehr Informationen als ein einzelnes Bild. Aus einer Bildsequenz kann zum Beispiel festgestellt werden, ob sich Objekte bewegt haben oder ob sich die Beleuchtung (z.B. von Lampen oder die Lichter einer Ampel) geändert hat. Wird die Bildsequenz mit einer Kamera aufgenommen, die sich während der Aufnahme translatorisch bewegt, so kann auch die Entfernung zu den Objekten im Bild bestimmt werden.

Ziel ist es in der vorliegenden Arbeit, Objekte aus einer Bildsequenz zu extrahieren, die sich selbst bewegen. Es wird ein mit einer Kamera ausgestattetes Fahrzeug eingesetzt, das Bilder aus der Umgebung des Fahrzeuges liefert. Das Fahrzeug und die Kamera werden von einem Algorithmus gesteuert, der eine bestimmte Aufgabe zu erfüllen hat. Die Steuerung der Kamera und des Fahrzeuges ist also durch einen Kontrollalgorithmus vorgegeben und wird nicht beeinflußt. Die bewegten Objekte werden also von einem passiven Algorithmus extrahiert werden. Nur so ist eine gemeinsame Nutzung der Ressourcen möglich. Dies steht im Gegensatz zum Ansatz von Daniilidis et al. [Daniilidis et al. 95], die eine aktive Steuerung der Kamera vornehmen um ein bewegtes Objekt zu verfolgen.

Natürlich wäre es denkbar, die Ressourcen nur für ein kurzes Zeitintervall zu nutzen und dann an den nächsten Nutzer abzugeben. Doch dann könnte es passieren, daß ein anderer Algorithmus das Fahrzeug oder die Kamera in eine andere Position gebracht hat und diese Bewegung rückgängig gemacht werden müßte. Kurz gesagt, es ist nur ein Fahrzeug und nur eine Kamera vorhanden. Ein Fahrzeug kann nicht gleichzeitig nach links und rechts fahren. Ebenso hat eine Kamera nur eine einzige Blickrichtung. Die Kontrollalgorithmen müßten also alle in etwa das gleiche Ziel verfolgen. Da dies relativ unwahrscheinlich ist (immerhin sind es unterschiedliche Algorithmen), sollte entweder ein globales Ziel definiert werden, das dann alle Algorithmen befolgen, oder die Algorithmen können für einen hinreichend langen Zeitraum ihr Ziel verfolgen, ohne daß dabei die anderen Algorithmen behindert werden. In der vorliegenden Arbeit wird die Bewegung der Kamera und des Fahrzeuges als gegeben hingenommen. Eine Steuerung erfolgt nicht. Somit kann der Algorithmus zusammen mit jeglichen anderen Verfahren eingesetzt werden. Daher ist dieser Ansatz für einen kooperativen Einsatz in einer komplexen Architektur eines Robotersystems [Rausch et al. 95] geeignet.

In der vorliegenden Arbeit wurde ein robuster Algorithmus entwickelt. Robust heißt hier (wie bei Nelson [Nelson 91]), daß der Algorithmus Fehler in der Eingabe toleriert und auch mit ungenauen Daten arbeitet.

1.2 Gang der Arbeit

Zunächst wurde in diesem Kapitel die Motivation dargelegt, einen Algorithmus zu entwickeln, der bewegte Objekte aus einer Videosequenz extrahiert.

Im folgenden Kapitel werden die Ergebnisse der Literaturrecherche zusammengefaßt, die zu Beginn der Arbeit durchgeführt wurde. Es stellte sich zunächst die Frage, welche Ansätze es gibt, Informationen aus einer Videosequenz zu extrahieren. Hierbei kommt es noch nicht darauf an, vollständige Objekte zu extrahieren, sondern jegliche Art von Veränderungen. Es werden Ansätze beschrieben, die für die vorliegende Arbeit relevant sind. Neben diesen Ansätzen wird aber auch auf Ansätze eingegangen, die in der vorliegenden Arbeit nicht eingesetzt werden. Diese Ansätze werden erklärt, um sie vorzustellen und dann gemeinsam mit den verwendeten

Ansätzen am Schluß von Kapitel 2 zu bewerten.

Schließlich wurde der Ansatz, die Eigenbewegung der Kamera zu kompensieren und dann die bewegten Objekte zu extrahieren, gewählt. Daher stellte sich die Frage, wie Objekte aus einer mit stationärer Kamera aufgenommenen Sequenz extrahiert werden können. In Kapitel 3 sind deshalb die Ergebnisse einer weiteren Literaturrecherche zusammengefaßt. Diese zweite Literaturrecherche setzt auf den Ergebnissen von Kapitel 2 auf. Es werden hier Verfahren betrachtet, die es ermöglichen einzelne Regionen eines Differenzbildes zu Objekten zusammenzufassen.

In Kapitel 4 werden die Ideen der in Kapitel 2 und Kapitel 3 gefundenen Ansätze zusammengefaßt, die für die vorliegende Arbeit relevant sind. Die Erweiterungen und Ergänzungen der Ansätze werden hier ebenfalls beschrieben. In Kapitel 4 wird noch nicht auf eine konkrete Implementierung Bezug genommen. Hier werden die theoretischen Hintergründe zu dem entwickelten Algorithmus geschildert. So wurden die Details der Implementierung von den Ideen getrennt.

In Kapitel 5 wird die Implementierung des Algorithmuses besprochen. Hier wird auf alle Details eingegangen, die die verwendete Hard- und Software betreffen. Die Modellierung der Hardware wird hier ebenfalls erleutert. Schließlich werden in Kapitel 5 alle Details der Implementierung beschrieben. Der tatsächlich implementierte Algorithmus wird hier schrittweise erklärt. Da der Algorithmus sehr umfangreich ist, wird hier auf Pseudo-Code verzichtet. Stattdessen werden die in Kapitel 4 verwendeten Ideen in Symbole mit exakter Bedeutung gefaßt, und anhand dieser Symbole die Funktionsweise des Algorithmuses beschrieben. Neben der Funktionsweise wird hier auch auf die Benutzung der hier entwickelten Software eingegangen.

Die durchgeführten Experimente und Untersuchungen der vorgenommenen Implementierung sind in Kapitel 6 zusammengefaßt. Hier wird überprüft, ob die Implementierung in der Lage ist, bewegte Objekte aus einer Videosequenz zu extrahieren. Situationen, in denen der Algorithmus an seine Grenzen stößt, werden hier ebenfalls besprochen. Die durchgeführten Zeitmessungen sind in Kapitel 6 zusammengefaßt.

In Kapitel 7 sind nochmals die wesentlichen Ergebnisse der vorliegenden Arbeit zusammengefaßt. Möglichkeiten zur Erweiterung der vorliegenden Arbeit werden aufgezeigt.

Im Anhang ist schließlich das verwendete Bildformat, mit dem die Videosequenzen abgespeichert wurden, beschrieben. Daher wird die Möglichkeit gegeben, die Bilder der hier aufgenommenen Videosequenzen weiter zu nutzen. Ein Hilfsprogramm, das zur Betrachtung der Videosequenzen geschrieben wurde, wird hier ebenfalls beschrieben. Die für die vorliegende Arbeit entwickelte Bibliothek, mit der die Bilder visualisiert wurden, wird erklärt. Schließlich sind einige Videosequenzen aufgeführt, bei denen der Algorithmus bewegte Objekte erfolgreich extrahierte.

Kapitel 2

Bewegte Bilder

Bewegte Bilder vermitteln eine Fülle von Informationen. Verschiedene Methoden zur Analyse von Bildsequenzen werden ausführlich in [Huang 81] oder [Huang 83] beschrieben. Einen Überblick geben auch die Artikel von Nagel [Nagel 86] und Aggarwal [Aggarwal 86]. Wie bewegte Bilder vom visuellen System des Menschen wahrgenommen werden, beschreibt Ullman [Ullman 79]

Zunächst werden in diesem Kapitel die Ergebnisse einer zu Beginn der vorliegenden Arbeit durchgeführten Literaturrecherche besprochen. Hier werden verschiedene Verfahren besprochen, Veränderungen in einer Bildsequenz zu detektieren. Einige der vorgestellten Verfahren werden in der vorliegenden Arbeit eingesetzt. Es werden aber auch Verfahren besprochen, die in der vorliegenden Arbeit nicht verwendet werden. Die Gründe hierfür sind am Ende des Kapitels in einer Bewertung der Ansätze zusammengefaßt.

2.1 Entstehung des Bildes

Im folgenden wird zunächst besprochen, wie aus der von der Kamera betrachteten Szene ein Bild entsteht. Die physikalischen Hintergründe sind in Horn [Horn 86] und Levi [Levi 93] ausführlich beschrieben. Es werden zwei Modelle besprochen, mit denen Objektpunkte auf die korrespondierenden Bildpunkte abgebildet werden können. Beide Verfahren sind ausführlich in Horn [Horn 86] beschrieben.

2.1.1 Perspektivische Projektion

Der perspektivischen Projektion liegt das Modell einer Lochkamera zugrunde. Es wird der Objektpunkt $\mathbf{P}_O$ auf den Bildpunkt $\mathbf{P}_B$ abgebildet. Das Licht, das vom Punkt $\mathbf{P}_O$ ausgeht, trifft also am Punkt $\mathbf{P}_B$ auf dem Bild auf. Das Licht verläuft dabei in einer Gerade direkt durch die Blende der Kamera. Die Blende der Kamera befinde sich im Koordinatenursprung $\mathbf{P}_U$ und die Fläche, auf die das Bild projeziert wird, befinde sich in der XY-Ebene im Abstand f vom Koordinatenursprung. Der Objektpunkt $\mathbf{P}_O$ habe die Koordinaten $[X, Y, Z]^T$. Der Bildpunkt $\mathbf{P}_B$ habe die Koordinaten $[x, y, f]^T$. Die Projektion des Punktes $\mathbf{P}_O$ auf die Z-Achse sei $\mathbf{P}_Z$. Mit $\mathbf{P}_f$ werde der Punkt im Abstand f von Ursprung auf der Bildebene bezeichnet. Aus der Ähnlichkeit der Dreiecke $\triangle \mathbf{P}_U\mathbf{P}_O\mathbf{P}_Z$ und $\triangle \mathbf{P}_U\mathbf{P}_B\mathbf{P}_f$ ergeben sich die Koordinaten des Bildpunktes $\mathbf{P}_B$ in der Bildebene:

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z} \tag{2.1}$$

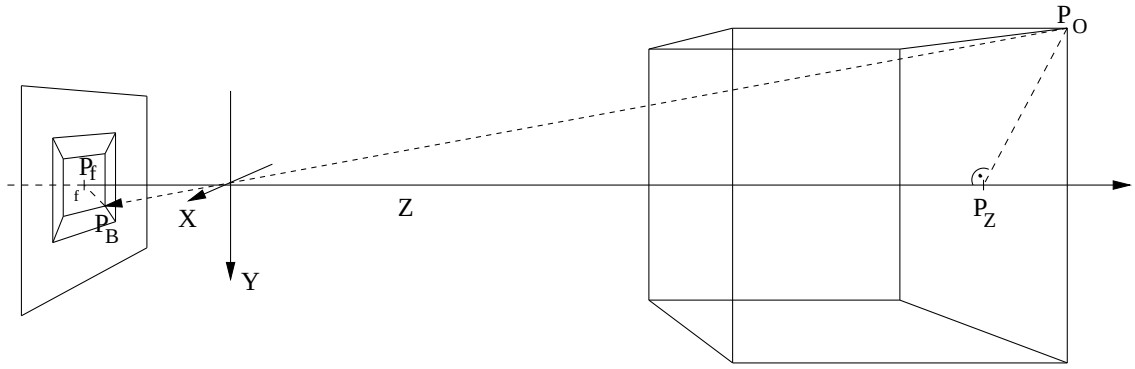

Abbildung 2.1: Perspektivische Projektion eines ein Drahtgerüstes in der Form eines Würfels.

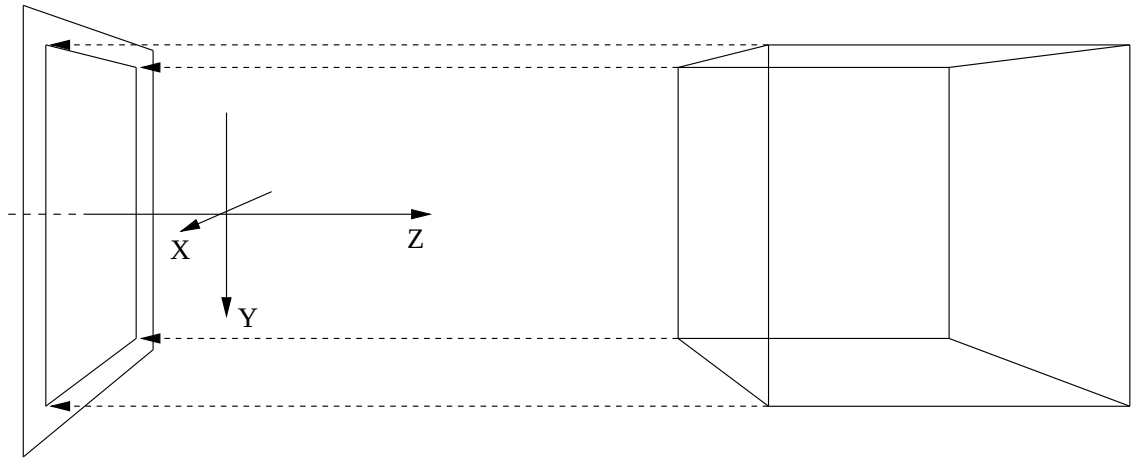

Abbildung 2.2: Orthogonale Projektion eines ein Drahtgerüstes in der Form eines Würfels.

In Abbildung 2.1 ist die perspektivische Projektion für ein Drahtgerüst in der Form eines Würfels dargestellt. Bei der perspektivischen Projektion ist darauf zu achten, daß das Bild, das auf der Projektionsfläche entsteht, ein Spiegelbild der durch die Lochkamera aufgenommenen Szene darstellt.

2.1.2 Orthogonale Projektion

Bei der orthogonalen Projektion wird jeder Punkt $\mathbf{P}_O$ eines Objektes senkrecht auf die Bildebene projeziert. Der Skalierungsfaktor m bestimmt die Größe des projezierten Bildes. Die Koordinaten (x, y) des Bildpunktes P_B in der Bildebene ergeben sich aus den Koordinaten $[X, Y, Z]$ des Objektpunktes $\mathbf{P}_O$ nach

$$x = mX \qquad y = mY.$$

Sind die Tiefenunterschiede im betrachteten Bild klein in Relation zum Abstand von der Kamera zu den betrachteten Objekten, so unterscheidet sich die orthogonale Projektion kaum von der perspektivischen Projektion. Der Skalierungsfaktor kann durch $m = \frac{f}{\bar{Z}}$ approximiert werden. $\bar{Z}$ gibt dabei die durchschnittliche Entfernung zum Hintergrund an. In Abbildung 2.2 ist die orthographische Projektion eines würfelförmigen Drahtgerüstes zu sehen.

2.2 Bewegungsfeld

In der vorliegenden Arbeit wird das Bewegungsfeld berechnet, um die Eigenbewegung der Kamera zu kompensieren. Daher wird hier kurz das Bewegungsfeld besprochen. Bewegt sich ein Punkt eines Objektes im Sichtbereich der Kamera, so verursacht diese Bewegung in der Regel eine Bewegung des entsprechenden Bildpunktes. Die Geschwindigkeitsvektoren der Bildpunkte definieren das Bewegungsfeld [Horn 86]. Die Bewegung eines Bildpunktes kann direkt aus der Bewegung eines Objektpunktes berechnet werden. Hierbei wird angenommen, daß der Objektpunkt im Sichtbereich der Kamera bleibt, also nicht verdeckt wird.

Das Bewegungsfeld soll nun berechnet werden. Der Objektpunkt $\mathbf{P}_O$ werde mit perspektivischer Projektion (Gleichung 2.1) auf den Bildpunkt $\mathbf{P}_B$ abgebildet. Der Ursprung dieser Vektoren sei im Zentrum der Projektion. Der Abstand der Bildebene zur Linse sei f. Die Geschwindigkeit $\mathbf{V}_O$ des Objektpunktes $\mathbf{P}_O$ und die Geschwindigkeit $\mathbf{V}_B$ des Bildpunktes $\mathbf{P}_B$ ergeben sich aus

$$\mathbf{V}_O = \frac{d\mathbf{P}_O}{dt} \tag{2.2}$$

und

$$\mathbf{V}_B = \frac{d\mathbf{P}_B}{dt}. \tag{2.3}$$

Aufgrund der perspektivischen Projektion gilt für die beiden Vektoren $\mathbf{P}_B$ und $\mathbf{P}_O$ folgende Abhängigkeit.

$$\frac{\mathbf{P}_B}{f} = \frac{\mathbf{P}_O}{\mathbf{P}_O \cdot \hat{\mathbf{Z}}} \tag{2.4}$$

Dabei sei $\hat{\mathbf{Z}}$ der Einkeitsvektor senkrecht zur Bildebene. Differenziert man beide Seiten von Gleichung 2.4 nach der Zeit, so folgt mit Gleichung 2.2 und Gleichung 2.3 [Horn 86].

$$\frac{d}{dt}\left(\frac{\mathbf{P}_B}{f}\right) = \frac{d}{dt}\left(\frac{\mathbf{P}_O}{\mathbf{P}_O \cdot \hat{\mathbf{Z}}}\right) \tag{2.5}$$

$$\frac{\mathbf{V}_B}{f} = \frac{(\mathbf{P}_O \cdot \hat{\mathbf{Z}})\mathbf{V}_O - (\mathbf{V}_O \cdot \hat{\mathbf{Z}})\mathbf{P}_O}{(\mathbf{P}_O \cdot \hat{\mathbf{Z}})^2} \tag{2.6}$$

$$= \frac{(\mathbf{P}_O \times \mathbf{V}_O) \times \hat{\mathbf{Z}}}{(\mathbf{P}_O \cdot \hat{\mathbf{Z}})^2} \tag{2.7}$$

Also läßt sich aus den Geschwindigkeitsvektoren $\mathbf{V}_O$ und den Ortsvektoren $\mathbf{P}_O$ der Objektpunkte das Bewegungsfeld der Bildpunkte berechnen.

2.3 Optischer Fluß

Der optische Fluß gibt wie das Bewegungsfeld für jeden Bildpunkt einen Geschwindigkeitsvektor an. Der optische Fluß basiert aber im Gegensatz zum Bewegungsfeld nur auf der Bewegung der Bildpunkte [Horn 86]. Er beschreibt, mit welcher Geschwindigkeit sich ein Pixel im Bild bewegt, ohne zu wissen, wodurch die Bewegung im Bild verursacht wurde.

Das Bewegungsfeld ist im allgemeinen nicht dasselbe wie der optische Fluß. Ein Beispiel soll dies verdeutlichen [Horn 86]. Der optische Fluß einer rotierenden, einfarbigen Kugel unter konstanter Beleuchtung ist Null. Die Grauwerte des Bildes zeigen also keine Veränderung auf. Da sich die Punkte der Kugel aber bewegen, ergibt sich aus Gleichung 2.7 ein Bewegungsfeld ungleich Null. In einem weiteren Versuch stehe die Kugel still und die Beleuchtung ändere sich. Dann ist das Bewegungsfeld nach Gleichung 2.7 Null. Der optische Fluß ist aber ungleich Null, weil sich die Grauwerte im Bild der Kugel verändern.

In der vorliegenden Arbeit wird der optische Fluß nur für markante Punkte des Bildes berechnet. Wird der optische Fluß für das gesamte Bild berechnet, so können die bewegten Objekte durch die Segmentierung des optischen Flusses extrahiert werden. Die Diskontinuitätsgrenzen können zur Trennung der Szene in bewegte und stationäre Objekte herangezogen werden. Smith et al. [Smith et al. 95] verfolgen diesen Ansatz. Den optischen Fluß berechnen Smith et al. [Smith et al. 94] durch den Vergleich von Merkmalen des Bildes (sogenannten *Features*). Bouthemy et al. [Bouthemy et al. 93] berechnen den optischen Fluß implizit und segmentieren die Bildsequenz mittels eines zweidimensionalen affinen Bewegungsmodells, um bewegte Objekte für beliebige Kamerabewegungen zu extrahieren.

Von Horn et al. [Horn et al. 81] wurde ein iteratives Verfahren entwickelt um den optischen Fluß zu berechnen. Im folgenden steht $E(x, y, t)$ für die Bestrahlungsstärke [Levi 93] zum Zeitpunkt t. Horn nimmt von einem Bildpunkt (x, y) an, daß sich dieser in der Zeit δt mit der Geschwindigkeit u in x-Richtung und v in y-Richtung bewegt. Dann gilt

$$E(x + u\delta t, y + v\delta t, t + \delta t) = E(x, y, t).$$

Diese Annahme führt zu der Gleichung

$$E_x u + E_y v + E_t = 0$$

wobei

$$E_x = \frac{\partial E}{\partial x}, \quad E_y = \frac{\partial E}{\partial y}, \quad E_t = \frac{\partial E}{\partial t}, \quad u = \frac{dx}{dt} \quad \text{und} \quad v = \frac{dy}{dt}.$$

Diese Gleichung alleine reicht aber nicht aus, um den optischen Fluß zu berechnen. Es kann lediglich der optische Fluß in Richtung des Gradienten ∇E bestimmt werden. Die Komponente des optischen Flusses senkrecht dazu kann nicht aus obiger Gleichung bestimmt werden. Horn [Horn 86] nimmt an, daß sich der optische Fluß nur stetig ändert. Mit Hilfe des Variationskalküls entwickelte Horn ein Verfahren um den optischen Fluß einer Bildfolge zu berechnen. Für den diskreten Fall läßt sich der optische Fluß durch

$$u^{n+1}(x, y) = \bar{u}(x, y) - \lambda \frac{E_x \bar{u}^n(x, y) + E_y \bar{v}^n(x, y) + E_t}{1 + \lambda(E_x^2 + E_y^2)} E_x$$

und

$$v^{n+1}(x, y) = \bar{v}(x, y) - \lambda \frac{E_x \bar{u}^n(x, y) + E_y \bar{v}^n(x, y) + E_t}{1 + \lambda(E_x^2 + E_y^2)} E_y$$

aus dem optischen Fluß $(u^n(x,y), v^n(x,y))$ der vorhergehenden Iteration n bestimmen. $(\bar{u}, \bar{v})$ gibt den durchschnittlichen Fluß der benachbarten Bildpunkte an. Mit λ wird die Abweichung von der Stetigkeitsbedingung gewichtet.

Francois et al. [Francois et al. 91] berechnen den optischen Fluß implizit durch Definition einer Energiefunktion und erreichen so eine Segmentierung der Bildsequenz für beliebige Kamerabewegungen. Ein ähnlicher Ansatz, allerdings für eine stationäre Kamera, wird von [Bouthemy et al. 93] beschrieben.

2.4 Detektion von Veränderungen im Bild

Im folgenden werden einige Verfahren betrachtet, mit denen Veränderungen aus einer mit stationärer Kamera aufgenommen Bildsequenz extrahiert werden können. Veränderungen im Bild können durch bewegte Objekte verursacht worden sein. Veränderungen zeigen sich aber auch, wenn sich die Beleuchtung ändert.

Zunächst werden Differenzbilder aufeinander folgender Bilder betrachtet. Differenzbilder stellen eine relativ einfache Möglichkeit dar, Veränderungen zu detektieren. Bewegte Kanten und Ecken können ebenfalls aus aufeinander folgenden Bildern extrahiert werden. Ein robustes Verfahren, Veränderungen zu detektieren, wird ebenfalls besprochen. Schließlich wird ein Verfahren besprochen, daß eine von der Beleuchtung der Szene unabhängige Detektion von Veränderungen ermöglicht.

2.4.1 Differenzbilder

Das Differenzbild ist die diskrete Version der zeitlichen Ableitung und gibt die Veränderung der Grauwerte im Zeitraum δt des Bildes B an [Murray et al. 94a]:

$$\frac{\partial B(x,y,t)}{\partial t} \approx \frac{B(x,y,t) - B(x,y,t-\delta t)}{\delta t}.$$

Um Rauschen zu unterdrücken, werden gewöhnlich nur Differenzen, die größer als ein Schwellwert sind, als Veränderungen gewertet. Jain et al. [Jain et al. 95] definieren die folgenden Arten von Differenzbildern wobei τ ein Schwellwert ist.

- Differenzbild DB

$$DB_{t_1,t_2}(x,y) = \begin{cases} 1 & \text{falls} \quad |B(x,y,t_1) - B(x,y,t_2)| > \tau \\ 0 & \text{sonst} \end{cases}$$

- Positives Differenzbild PDB

$$PDB_{t_1,t_2}(x,y) = \begin{cases} 1 & \text{falls} \quad B(x,y,t_1) - B(x,y,t_2) > \tau \\ 0 & \text{sonst} \end{cases}$$

- Negatives Differenzbild NDB

$$NDB_{t_1,t_2}(x,y) = \begin{cases} 1 & \text{falls} \quad B(x,y,t_1) - B(x,y,t_2) < \tau \\ 0 & \text{sonst} \end{cases}$$

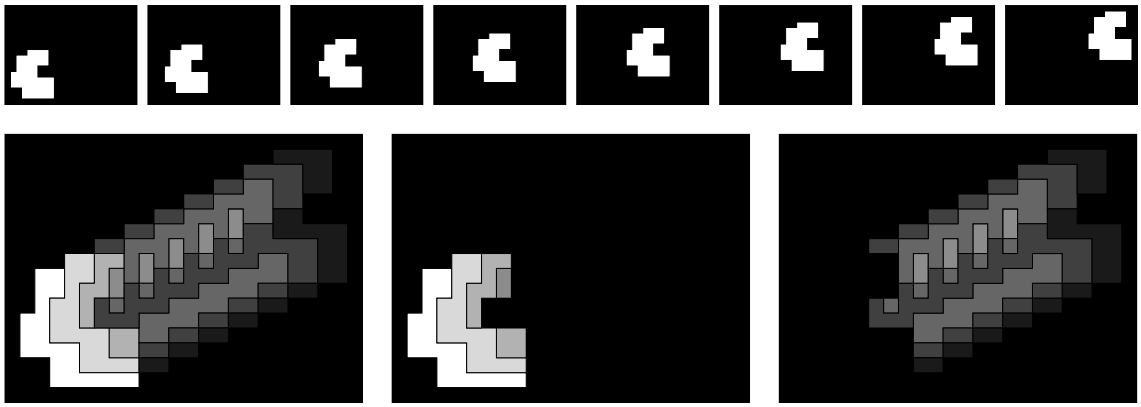

Abbildung 2.3: Angesammeltes, absolutes Differenzbild $AADB$, angesammeltes, positives Differenzbild $APDB$, angesammeltes, negatives Differenzbild $ANDB$ für ein Objekt, das sich von links unten nach rechts oben bewegt. Das Objekt wird durch hohe Pixelwerte repräsentiert.

- Angesammeltes, absolutes Differenzbild $AADB$

$$\begin{aligned} AADB_0(x,y) &= 0 \\ AADB_n(x,y) &= AADB_{n-1}(x,y) + DB_{1,n}(x,y) \end{aligned}$$

- Angesammeltes, positives Differenzbild $APDB$

$$\begin{aligned} APDB_0(x,y) &= 0 \\ APDB_n(x,y) &= APDB_{n-1}(x,y) + DB_{1,n}(x,y) \end{aligned}$$

- Angesammeltes, negatives Differenzbild $ANDB$

$$\begin{aligned} ANDB_0(x,y) &= 0 \\ ANDB_n(x,y) &= ANDB_{n-1}(x,y) + DB_{1,n}(x,y) \end{aligned}$$

Jain [Jain 84] und Jain et al. [Jain et al. 95] verwenden die angesammelten positiven und negativen Differenzbilder $APDB$ und $ANDB$ um bewegte Objekte zu extrahieren. Ein Objekt wird aus dem Referenzbild extrahiert, wenn die Region im angesammelten Differenzbild aufgehört hat zu wachsen. Das Wachstum der Region ist abgeschlossen, sobald sich das Objekt vollständig von seinem Platz bewegt hat. Also muß solange gewartet werden, bis sich das Objekt vollständig von seinem Platz bewegt hat. Erst dann kann das Objekt extrahiert werden.

In Abbildung 2.3 ist das $AADB$, das $APDB$ und das $ANDB$ für ein Objekt gezeigt, daß sich von links unten nach rechts oben bewegt. Helle Grauwerte sind durch hohe Pixelwerte

repräsentiert. Das Objekt kann in diesem Beispiel nach vier Schritten aus dem $APDB$ extrahiert werden. Nach dem dritten Schritt hat es seine volle Größe erreicht. Dies kann nach einem weiteren Schritt festgestellt werden.

Aus welchem angesammelten Differenzbild (positives oder negatives) das Objekt extrahiert werden muß, hängt von der Farbe bzw. vom Grauwert des Objektes ab. Daher müssen beide angesammelten Differenzbilder beobachtet werden. In Abbildung 2.3 hätte das Objekt bei einer Umkehr der Pixelwerte aus dem $ANDB$ extrahiert werden müssen.

2.4.2 Bewegte Kanten und Ecken

Verfahren, die bewegte Kanten extrahieren, werden oft eingesetzt um Bewegung in einer Bildsequenz zu detektieren und gleichzeitig die Datenmenge zu reduzieren. Bewegte Kanten werden von Murray et al. [Murray et al. 94a], Picton [Picton 89] und Shah et al. [Shah et al. 84] extrahiert. Nach Jain et al. [Jain et al. 95] wird eine bewegte Kante wie folgt definiert: eine bewegte Kante ist eine Kante im Bild *und* sie bewegt sich.

Die Und-Verknüpfung kann durch eine Multiplikation realisiert werden. Also ergibt sich das Bild mit den bewegten Kanten BK durch Multiplikation des Kanten- und des Differenzbildes.

$$BK(x,y,t) = \frac{\partial B(x,y,t)}{\partial t} \cdot \nabla B(x,y,t)$$

Shah et al. [Shah et al. 84], Picton [Picton 89] und Jain et al. [Jain et al. 95] weisen darauf hin, daß es von Vorteil ist, das Bild erst nach Durchführung der Multiplikation und nicht früher (etwa bei der Extraktion der Kanten oder der Berechnung des Differenzbildes) zu binarisieren. Dann werden schwache Kanten, die sich schnell bewegen oder starke Kanten, die sich langsam bewegen, trotzdem berücksichtigt.

Sugimoto et al. [Sugimoto et al. 86] verwenden die diskrete Version der zweiten zeitlichen Ableitung der Bildsequenz, um bewegte Kanten zu extrahieren. Das Bild der zweiten zeitlichen Ableitung wird mit LB bezeichnet.

$$\begin{aligned} LB(x,y,t) &= \frac{\partial^2}{\partial t^2} B(x,y,t) \\ &= \frac{1}{\delta t^2} \left(B(x,y,t+\delta t) - B(x,y,t)\right) - \left(B(x,y,t) - B(x,y,t-\delta t)\right) \\ &= \frac{1}{\delta t^2} \left(B(x,y,t+\delta t) - 2B(x,y,t) + B(x,y,t-\delta t)\right) \end{aligned}$$

Sugimoto binarisieren das Bild LB, indem sie einen Schwellwert setzen. Die Nulldurchgänge in LB geben jedoch die exakte Position der bewegten Kanten zum Zeitpunkt t an. In Abbildung 2.4 ist die Ausgabe des Operators am Beispiel eines bewegten Rechtecks verdeutlicht. Das Rechteck bewegt sich von links oben nach rechts unten. Helle Grauwerte sind durch hohe Pixelwerte und dunkle durch kleine Pixelwerte dargestellt. Im Differenzbild und in der zweiten zeitlichen Ableitung hat Grau den Pixelwert Null.

Dubuisson et al. [Dubuisson et al. 95] extrahieren bewegte Kanten durch Multiplikation von Differenzbildern. Sie glätten das Bild zuerst, um eine glatte Steigung an den Kanten zu erhalten. Dann multiplizieren sie aufeinander folgende Differenzbilder DB um die Kanten für das mittlere Bild zu erhalten. Für den eindimensionalen Fall ist dies in Abbildung 2.5 dargestellt.

$$BK(x,y,t) = DB_{t-\delta t,t}(x,y) \cdot DB_{t,t+\delta t}(x,y)$$

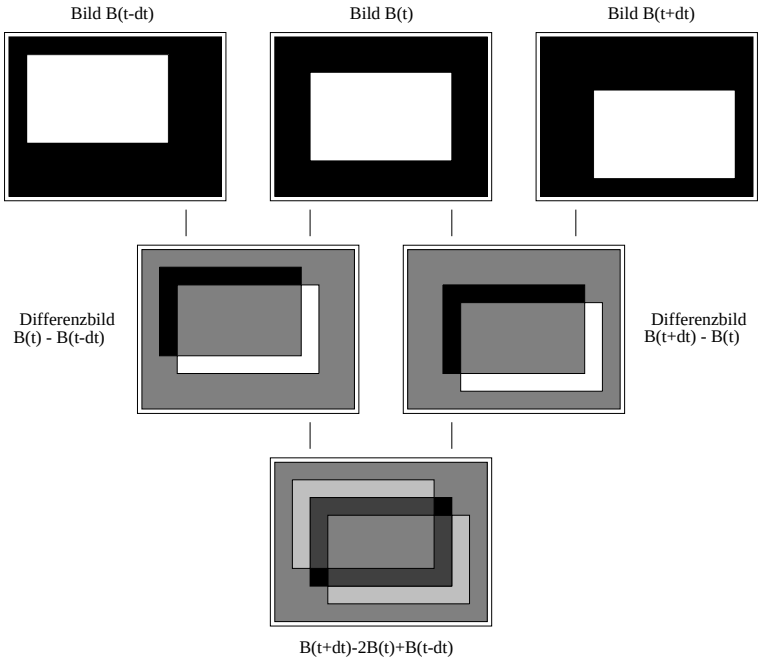

Abbildung 2.4: Zweite zeitliche Ableitung eines bewegten Rechtecks.

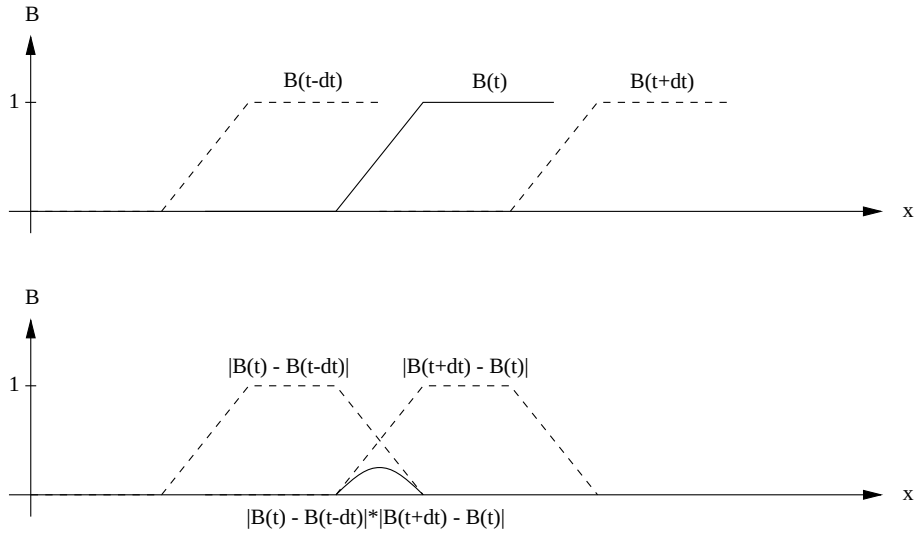

Abbildung 2.5: Extraktion einer bewegten Kante für den eindimensionalen Fall (angepaßt nach [Dubuisson et al. 95]).

Shah et al. [Shah et al. 84] extrahieren bewegte Ecken aus einer Bildsequenz. Sie verwenden die Und-Verknüpfung wie bei der Extraktion bewegter Kanten um bewegte Ecken zu lokalisieren.

$$BE(x,y,t) \quad = \quad C(B(x,y,t)) \cdot \frac{\partial B(x,y,t)}{\partial t}$$

Dabei ist C ein Operator, der "Eckigkeit" der betrachteten Objektpunkte ermittelt. Für jeden Punkt berechnet der Operator also, in welchem Maß der Punkt ein Eckpunkt des Bildes ist. Wie bei den bewegten Kanten wird eine Binarisierung mit einem Schwellwert durchgeführt, um die bewegten Ecken zu finden.

Lee et al. [Lee 90] extrahieren Ecken aus einer Bildsequenz und setzen drei Bilder der Sequenz ein, um eine Korrespondenz zwischen den Ecken herzustellen. Für jeden Eckpunkt (x_1, y_1) des ersten Bildes stellen sie eine Korrespondenz mit allen Eckpunkten, die sich in einem Fenster um (x_1, y_1) im zweiten Bild befinden, her. Für diese Punktpaare berechnen sie einen Bewegungsvektor. Anhand des Bewegungsvektors berechnen Lee et al. die vermutete Position eines Eckpunktes im dritten Bild. Innerhalb eines Fensters um die Prädiktion der Position der Ecke suchen sie nach tatsächlich extrahierten Ecken. So entstehen mehrere mögliche Bewegungsbahnen einer Ecke. Durch ein iteratives, auf Wahrscheinlichkeiten und Relaxation basierendes Verfahren bestimmen Lee et al. schließlich eindeutig die Bewegungsbahnen der Ecken.

2.4.3 Robuste Detektion von Veränderungen

Anstatt die zeitliche Ableitung auf einzelnen Pixel zu berechnen, kann auch für jeden Pixel eine kleine Umgebung R definiert werden [Jain et al. 95]. Die so definierten Regionen auf dem vorhergehenden und aktuellen Bild werden dann miteinander verglichen. Dies macht die Erkennung von Veränderungen robuster. Als Test, ob eine Veränderung stattgefunden hat, kann der folgende Vergleich, der auf der Wahrscheinlichkeitsverteilung der Grauwerte zweier Regionen basiert, herangezogen werden [Jain et al. 95].

$$\frac{\left[\frac{\sigma^2_{x,y,t_1}+\sigma^2_{x,y,t_2}}{2} + \left(\frac{\mu_{x,y,t_1}-\mu_{x,y,t_2}}{2}\right)^2\right]^2}{\sigma^2_{x,y,t_1}\sigma^2_{x,y,t_2}} > \tau$$

Dabei ist $\mu_{x,y,t}$ der Mittelwert und $\sigma^2_{x,y,t}$ die Varianz der Grauwerte der Bildpunkte in der Region $R_{x,y}$ um den Punkt (x, y) zum Zeitpunkt t und τ ist ein Schwellwert. Die Region $R_{x,y}$ bestehe aus n Punkten. Ist der berechnete Wert größer als der Schwellwert, so hat eine Veränderung an der Position (x, y) stattgefunden. Der Mittelwert $\mu_{x,y,t}$ und die Varianz $\sigma^2_{x,y,t}$ werden wie folgt berechnet.

$$\begin{aligned}
\mu_{x,y,t} &= \frac{1}{n} \sum_{(x',y') \in R_{x,y}} B(x', y', t) \\
\sigma^2_{x,y,t} &= \frac{1}{n} \sum_{(x',y') \in R_{x,y}} [\mu_{x,y,t} - B(x', y', t)]^2
\end{aligned}$$

Der Vergleich basiert auf der Wahrscheinlichkeitsverteilung der Grauwerte zweier Regionen. Dabei werden zwei Hypothesen miteinander verglichen:

$$\begin{aligned} H_0 &= \text{Die Regionen gehören zu demselben Objekt.} \\ H_1 &= \text{Die Regionen gehören zu verschiedenen Objekten.} \end{aligned}$$

Die Grauwerte der beiden Objekte seien konstant und mit additivem gaußschen Rauschen überlagert. Daher seien die Grauwerte gaußverteilt. $g(x,y,t)$ geben im folgenden die Grauwerte der Region $R_{x,y}$ zum Zeitpunkt t an. Dann gibt $p(g(x,y,t)|H_0)$ die Wahrscheinlichkeit an, daß die Region $R_{x,y}$ zum Zeitpunkt t die Grauwerte $g(x,y,t)$ besitzt, unter der Annahme, daß die Hypothese H_0 gilt. $p(g(x,y,t)|H_1)$ gibt die Wahrscheinlichkeit an, daß die Region $R_{x,y}$ die Grauwerte $g(x,y,t)$ besitzt, unter der Annahme, daß die Hypothese H_1 gilt. Dann ist das Verhältnis der Wahrscheinlichkeiten $p(g(x,y,t)|H_1)$ und $p(g(x,y,t)|H_0)$

$$\frac{p(g(x,y,t)|H_1)}{p(g(x,y,t)|H_0)} = \frac{\left[\frac{\sigma^2_{x,y,t_1}+\sigma^2_{x,y,t_2}}{2} + \left(\frac{\mu_{x,y,t_1}-\mu_{x,y,t_2}}{2}\right)^2\right]^n}{\sigma^n_{x,y,t_1}\,\sigma^n_{x,y,t_2}}$$

Hsu et al. [Hsu et al. 84] modellieren das Bild mit einer stückweise linearen oder quadratischen bivariaten Grauwertfunktion. Sie verwenden den obigen Test um die Regionen zu vergleichen wobei

$$\sigma^2_{x,y,t} = \frac{1}{n} \sum_{(x',y')\in R_{x,y}} [a_0 + a_1 x' + a_2 y' - B(x',y',t)]^2$$

bei der Modellierung mit einer linearen Grauwertfunktion und

$$\sigma^2_{x,y,t} = \frac{1}{n} \sum_{(x',y')\in R_{x,y}} [a_0 + a_1 x' + a_2 y' + a_3 x'^2 + a_4 y'^2 + a_5 x' y' - B(x',y',t)]^2$$

bei der Modellierung mit einer quadratischen Grauwertfunktion ist. Die Parameter a_i werden auf der Region R mit der Methode der kleinsten Quadrate bestimmt.

2.4.4 Beleuchtungsunabhängige Bewegungserkennung

Eine Möglichkeit, Bewegungserkennung unabhängig von der Beleuchtung zu realisieren, besteht in der Normalisierung der Grauwerte [Skifstad et al. 89]. Werden zwei Regionen $R_{x,y}$ um den Punkt (x,y) Zeitpunkt t_1 und t_2 miteinander verglichen, so werden die Grauwerte der Region $R_{x,y}$ aus dem Bild zum Zeitpunkt t_2 zuerst normalisiert. Dazu wird jeder Grauwert $B(x,y,t_2)$ aus der Region $R_{x,y}$ entsprechend der folgenden Gleichung transformiert. Dies ergibt das Bild mit den normalisierten Grauwerten NG.

$$NG(x,y,t_2) = \frac{\sigma_{x,y,t_1}}{\sigma_{x,y,t_2}}(B(x,y,t_2) - \mu_{x,y,t_2}) + \mu_{x,y,t_1}$$

Die Grauwerte werden durch obige Skalierung so normalisiert, daß sie die gleiche Varianz und den gleichen Mittelwert besitzen. Auf den normalisierten Regionen kann dann ein Verfahren zur Detektion von Veränderungen durchgeführt werden.

Von Skifstad et al. [Skifstad et al. 89] wurden zwei Verfahren entwickelt, um Veränderungen unabhängig von der Beleuchtung zu erkennen. Das erste Verfahren wird *Derivative-Model-Method* genannt. Sie modellieren die Grauwerte $g(x, y, t)$ einer Region mit einer quadratischen bivariaten Funktion.

$$g(x, y, t) = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2$$

Um nun eine Veränderung in der Region $R_{x,y}$ zu erkennen, verwenden sie die folgende Differenzmetrik D.

$$D_{t_1,t_2}(x, y) = \sum_{(x',y') \in R_{x,y}} \left(\frac{\partial g(x', y', t_2)}{\partial x} + \frac{\partial g(x', y', t_2)}{\partial y} \right) - \left(\frac{\partial g(x', y', t_1)}{\partial x} + \frac{\partial g(x', y', t_1)}{\partial y} \right)$$

Die quadratische bivariate Funktion wird verwendet, um die Grauwerte möglichst exakt zu modellieren. Durch die Verwendung der partiellen Ableitung des Grauwertmodells soll die Erkennung von Veränderungen unabhängig von der Beleuchtung werden. Denn durch eine Veränderung in der Beleuchtung werden die Grauwerte oft nur verschoben. Allerdings ist der Ansatz nicht für große Beleuchtungsänderungen geeignet [Skifstad et al. 89].

Das zweite Verfahren wird von Skifstad et al. [Skifstad et al. 89] *Shading-Model-Method* genannt. Aus der Computergrafik verwenden sie ein Schattierungsmodell. Die Intensität eines Bildpunktes sei $B(x, y, t)$, die Beleuchtung zur Zeit t sei $I(t)$ und $S(x, y)$ sei ein Schattierungskoeffizient, der vom betrachteten Punkt (x, y) abhängt. Dann wird die Intensität eines Punktes durch

$$B(x, y, t) = S(x, y)I(t)$$

berechnet. Der Schattierungskoeffizient hängt vom betrachteten Objekt ab. Innerhalb einer kleinen Region sei er aber konstant. Nun wird dieselbe Region R in zwei Bildern betrachtet, die zu unterschiedlichen Zeitpunkten entstanden sind. Für das Verhältnis der Intensitäten der Punkte der beiden Bilder gilt

$$\frac{I(t_1)}{I(t_2)} = \frac{B(x, y, t_1)}{B(x, y, t_2)} = \frac{B(x', y', t_1)}{B(x', y', t_2)} \qquad (x, y), (x', y') \in R$$

falls sich das betrachtete Objekt nicht verändert hat. Der Schattierungskoeffizient kann gekürzt werden, da er innerhalb der betrachteten Region konstant ist. Das heißt, das Verhältnis der Intensitäten zweier Regionen ist innerhalb der Region konstant. Somit ist die Varianz des Verhältnisses gleich Null, wenn sich keine Veränderung ergeben hat. Bei einer Varianz größer Null hat eine Veränderung stattgefunden. Also wird der folgende Test zur Erkennung von Veränderungen verwendet.

$$\frac{1}{n} \sum_{(x',y') \in R_{x,y}} \left(\frac{B(x', y', t_1)}{B(x', y', t_2)} - \mu_{x,y,t_1,t_2} \right)^2 > \tau$$

Dabei ist

$$\mu_{x,y,t_1,t_2} = \frac{1}{n} \sum_{(x',y') \in R_{x,y}} \frac{B(x', y', t_1)}{B(x', y', t_2)}$$

der Erwartungswert der Intensitätsverhältnisse der Grauwerte zu den Zeitpunkten t_1 und t_2 in der Region $R_{x,y}$ um (x, y). Die Region $R_{x,y}$ habe n Punkte. Es wird ein Schwellwert τ definiert, ab dem die Varianz als ungleich Null betrachtet wird. Skifstad et al. [Skifstad et al. 89]

verwenden eine 5×5 Region um die Statistikdaten zu berechnen. Die Größe der Region sollte so groß sein, daß die berechneten Statistiken repräsentativ für die Beschaffenheit der Region ist.

Skifstad et al. [Skifstad et al. 89] weisen daraufhin, daß das vorgeschlagene Verfahren eine Veränderung der Helligkeit eines Objektes nicht feststellen kann. Wenn also von einem Bild zum nächsten ein dunkler Ball im Sichtbereich der Kamera durch einen hellen ersetzt wird, so kann dies mit obigem Verfahren nicht festgestellt werden.

2.5 Extraktion bewegter Objekte für translatorische Kamerabewegungen

2.5.1 Eigenbewegung-Polar-Transformation

Für den Fall, daß sich der Betrachter nur durch eine Translation, nicht aber durch Rotation bewegt, können bewegte Objekte mit der Eigenbewegung-Polar-Transformation bewegte Objekte extrahiert werden [Jain et al. 95]. Die Translation des Beobachters $[dx, dy, dz]^T$ sei bekannt. Dann läßt sich das Zentrum der Expansion (*Fokus of Expansion FOE*) berechnen.

$$\begin{aligned} x_{FOE} &= f\frac{dx}{dz} \\ y_{FOE} &= f\frac{dy}{dz} \end{aligned}$$

Da dz im Nenner steht, muß eine kleine Bewegung nach vorne oder hinten, also in Blickrichtung, erfolgen. Jetzt kann das Bild B durch die Eigenbewegung-Polar-Transformation transformiert werden. Das transformierte Bild wird mit EB für Eigenbewegungsbild bezeichnet.

$$EB(r, \theta, t) = B(x, y, t)$$

mit

$$\begin{aligned} r(x, y) &= \sqrt{(x - x_{FOE})^2 + (y - y_{FOE})^2} \\ \theta(x, y) &= \tan^{-1}\left(\frac{y - y_{FOE}}{x - x_{FOE}}\right) \end{aligned}$$

Also gibt r den Abstand zum FOE an und θ gibt den Winkel an, an dessen Position sich der Punkt (x, y) befindet. Stationäre Punkte des Hintergrundes bewegen sich im Bild EB nur in Richtung der r-Koordinate bei einer Translation des Beobachters. Lediglich ihr Radius zum FOE verändert sich. Bei einer Vorwärtsbewegung nimmt der Radius zu, bei einer Rückwärtsbewegung nimmt der Radius ab. Bewegte Objekte können sich auch in Richtung der θ-Koordinate im Eigenbewegung-Polar-Transformations-Raum bewegen, wenn ihre Bahn nicht in Richtung des FOEs liegt.

2.5.2 Komplex-logarithmische Transformation

Bei der komplex-logarithmischen Transformation [Jain et al. 95] wird die Position eines Bildpunktes (x, y) als Komplexe Zahl z betrachtet.

$$z = x + iy = r(\cos\theta + i\sin\theta) = re^{i\theta}$$

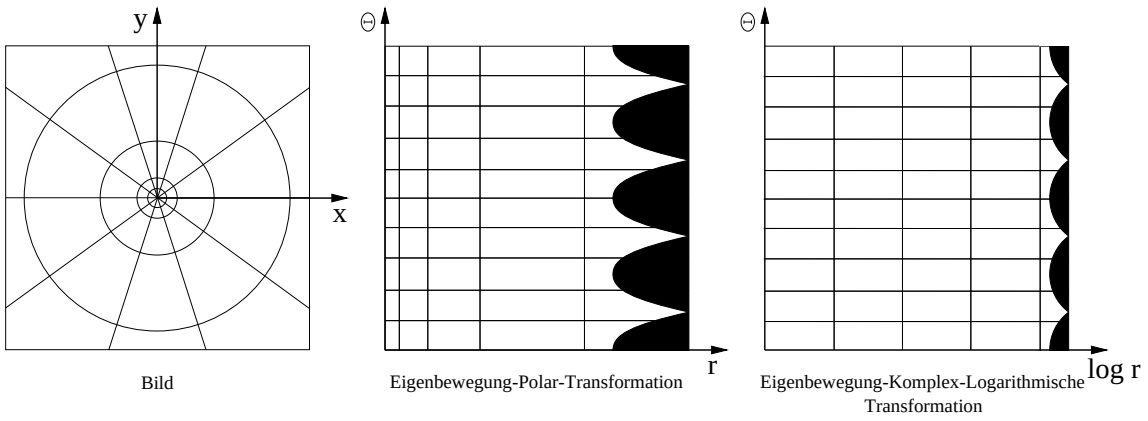

Abbildung 2.6: Kamerabild, Eigenbewegung-Polar-Transformation des Kamerabildes und Eigenbewegung-Komplex-Logarithmische-Transformation des Kamerabildes (angepaßt nach [Frazier et al. 90]). Das Zentrum der Expansion befinde sich im Zentrum des Bildes.

Die Bildpunkte z werden gemäß der folgenden Abbildung auf die Position ω transformiert. Der Ursprung befinde sich dazu in der Bildmitte.

$$\begin{aligned} \omega(r,\theta) &= \ln z \\ &= \ln(re^{i\theta}) \\ &= \ln r + i\theta \\ &= u(r,\theta) + iv(r,\theta) \end{aligned}$$

Also werden die Koordinaten durch

$$\begin{aligned} u(x,y) &= u(r(x,y),\theta(x,y)) = \ln r(x,y) = \ln\sqrt{x^2+y^2} \\ v(x,y) &= v(r,\theta) = \theta = \tan^{-1}\left(\frac{y}{x}\right) \end{aligned}$$

transformiert. Bewegt sich die Kamera in Blickrichtung, so gilt [Jain et al. 95] mit perspektivischer Projektion (Gleichung 2.1) für die Bewegung eines Punktes, dessen Objektpunkt sich im Abstand Z zur Kamera befindet, im Raum der komplex-logarithmischen Transformation

$$\frac{du}{dz} = -\frac{1}{Z}$$

und

$$\frac{dv}{dz} = 0.$$

Daher hängt die Bewegung eines Punktes im Raum der komplex-logarithmischen Transformation in Richtung der reellen Achse nur von der Entfernung Z ab, wenn sich die Kamera in Blickrichtung bewegt.

2.5.3 Eigenbewegung-Komplex-Logarithmische Transformation

Bei der Eigenbewegung-Komplex-Logarithmischen Transformation [Jain et al. 95] wird die Transformation der Bildpunkte in Bezug auf den FOE vorgenommen.

$$\begin{aligned}\omega(r,\theta) &= \ln z(x - x_{FOE}, y - y_{FOE}) \\ &= \ln(re^{i\theta}) \\ &= \ln r + i\theta \\ &= u(r,\theta) + iv(r,\theta)\end{aligned}$$

Bewegt sich die Kamera nur translatorisch, so hängt die Bewegung der Punkte auf der reellen Achse nur von der Z-Koordinate ab. Für die Eigenbewegung-Komplex-Logarithmische Transformation gilt bei beliebigem Translationsvektor $[dx, dy, dz]^T$ der Kamera mit $dz \neq 0$ [Jain et al. 95]

$$\frac{du}{dz} = -\frac{1}{Z}$$

und

$$\frac{dv}{dz} = 0.$$

Die Punkte des Eigenbewegung-Komplex-Logarithmischen-Raums, die sich in v-Richtung bewegen, werden also von einem bewegten Objekt verursacht. Ein Bild, das durch die Eigenbewegung-Komplex-Logarithmische Transformation transformiert wurde, ist in Abbildung 2.6 gezeigt. Die runden Bereiche am rechten Rand des transformierten Bildes entsprechen den Kanten des Originalbildes. Für ein rundes Originalbild würden die schwarzen Bereiche verschwinden.

2.6 Extraktion bewegter Objekte für rotatorische Kamerabewegungen

Murray et al. [Murray et al. 94a] kompensieren die Eigenbewegung einer Schwenk/Nick-Kamera. Die von Murray et al. durchgeführten Berechnungen werden hier nicht wiederholt, da die Modellierung speziell auf eine Schwenk/Nick-Kamera zugeschnitten ist. Für jeden Bildpunkt des aktuellen Bildes $(x(t_2), y(t_2))$ berechnen sie den korrespondierenden Bildpunkt des vorhergehenden Bildes $(x(t_1), y(t_1))$ entsprechend der folgenden Gleichungen. Dabei bezeichnet f die Brennweite der Kamera, θ bezeichnet den Nickwinkel der Kamera, α bezeichnet einen kleinen Drehwinkel um die Schwenkachse und γ bezeichnet einen kleinen Drehwinkel um die Nickachse.

$$\begin{aligned}x(t_1) &= f\frac{x(t_2) + y(t_2)\alpha\sin\theta + f\alpha\cos\theta}{-x(t_2)\alpha\cos + y(t_2)\gamma + f} \\ y(t_1) &= f\frac{-x(t_2)\alpha\sin\theta + y(t_2) + f\gamma}{-x(t_2)\alpha\cos + y(t_2)\gamma + f}\end{aligned}$$

So berechnen sie eine Prädiktion des aktuellen Bildes aus dem vorhergehenden Bild. Dann subtrahieren Murray et al. die Prädiktion vom aktuellen Bild und berechnen den Betrag der Subtraktion.

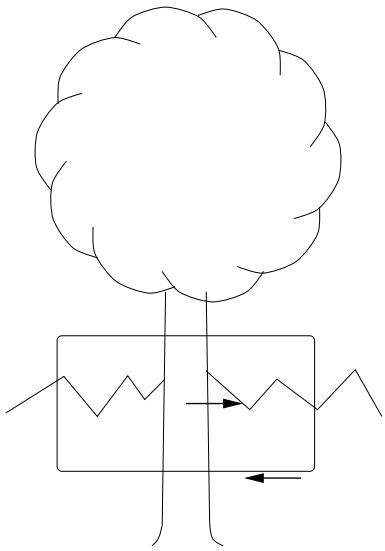

Abbildung 2.7: Betrachtet man einen Baum durch ein bewegtes Zugfenster, so scheint der Baum sich relativ zum Hintergrund zu bewegen. Dem Betrachter fehlt eine Verbindung des Baumes zum Hintergrund. Allerdings weiß der Betrachter natürlich, daß der Baum kein bewegtes Objekt darstellt.

Da die Positionsinformationen der Kamera nicht immer exakt sind, verwenden sie morphologische Operatoren, um Ungenauigkeiten in der Kompensation der Eigenbewegung zu beseitigen. Den morphologischen Operator Opening wenden Murray et al. auf das Absolutbild an, um kleine Bereiche zu beseitigen. Murray et al. weisen darauf hin, daß nicht nur Ungenauigkeiten in der Kompensation der Eigenbewegung durch die Anwendung des morphologischen Operators beseitigt werden, sondern eventuell auch bewegte Objekte. Dies ist der Fall, wenn die Bewegung der Objekte im Bild kleiner ist, als die Größe des verwendeten Strukturelements. Bei den Ergebnissen setzten Murray et al. eine Größe von 11×11 ein.

Nachdem Murray et al. Veränderungen aufeinander folgender Bilder detektiert haben, extrahieren sie bewegte Kanten. Dazu multiplizieren sie das Bild, das durch den Opening-Operator bearbeitet wurde mit dem Kantenbild des aktuellen Bildes. Erst dann binarisieren sie das Bild mit den bewegten Kanten durch Setzen eines Schwellwertes.

2.7 Bewertung der Ansätze

Ist die Entfernung der Objektpunkte zur Kamera und die Geschwindigkeit der Objektpunkte bekannt, so kann das Bewegungsfeld der Bildpunkte berechnet werden. Die Bildbewegung kann so relativ schnell berechnet werden. Dieser Ansatz ist für eine effiziente Implementierung geeignet. Werden geeignete Annahmen über den von der Kamera betrachteten stationären Hintergrund getroffen, dann kann das Bewegungsfeld der Bildpunkte aus der Eigenbewegung der Kamera berechnet werden. Dieser Ansatz wurde in der vorliegenden Arbeit verfolgt. Der Ansatz ist auch für große Kamerabewegungen geeignet.

Im Gegensatz dazu setzt die Berechnung des optischen Flusses in der Regel kleine Kame-

rabewegungen voraus [Weng et al. 92]. Auch Verfahren, die auf der Berechnung des optischen Flusses basieren, können getäuscht werden. Wird nur ein Bildausschnitt betrachtet, so kann oft nicht festgestellt werden, ob wirklich ein bewegtes Objekt extrahiert wird. Dies soll an einem Beispiel verdeutlicht werden. Bei großen Tiefenunterschieden in der betrachteten Szene scheint der Vordergrund sich relativ zum Hintergrund zu bewegen. Dies ist vor allem dann der Fall, wenn nur ein Bildausschnitt betrachtet wird. In Abbildung 2.7 ist ein Baum dargestellt, der durch ein Zugfenster betrachtet wird. Bewegt sich das Zugfenster nach links, dann scheint sich der Baum relativ zur Landschaft nach rechts zu bewegen. Dem Betrachter fehlt eine Verbindung des Baumes zum Hintergrund. Das Problem, bewegte Objekte robust zu extrahieren, erfordert Informationen über die Struktur der betrachteten Szene oder über die Kamerabewegung [Thompson et al. 87].

In der vorliegenden Arbeit wird ein einfaches Differenzbild zur Detektion von Veränderungen eingesetzt. Das Differenzbild läßt sich relativ schnell berechnen. Bewegte Objekte können auch, wie in Abschnitt 2.4.1 beschrieben, mittels angesammelter Differenzbilder extrahiert werden; allerdings muß solange gewartet werden, bis sich das Objekt vollständig von seinem Platz bewegt hat. Bei den Ansätzen zur robusten Detektion von Veränderungen müssen die Veränderungen auf der Ebene der Superpixel [Jain et al. 95] erkannt werden. Die beleuchtungsunabhängige Bewegungserkennung hat den Nachteil, daß Farbveränderungen nicht erkannt werden. Interessant ist der Ansatz, bewegte Objekte mit der Eigenbewegung-Komplex-Logarithmischen Transformation zu extrahieren. Allerdings kann dieser Ansatz nur für translatorische Kamerabewegungen eingesetzt werden.

Kapitel 3

Zusammenfassen von Regionen zu Objekten

Für die vorliegende Arbeit wird zunächst die Eigenbewegung der Kamera kompensiert und dann wie auf einer mit stationärer Kamera aufgenommenen Videosequenz gearbeitet. Bewegungen werden durch Bildung eines Differenzbildes detektiert. Dabei entstehen einzelne Bereiche. Diese Bereiche müssen zu einem Objekt zusammengefaßt werden. Es wurden eine Reihe von Ansätzen betrachtet, wie die Bereiche zusammengefaßt werden können. Die verschiedenen Ansätze sollen nun besprochen werden.

3.1 Bestimmung des stationären Hintergrundes

Bei einer mit stationärer Kamera aufgenommenen Videosequenz ist es relativ einfach, den stationären Hintergrund zu bestimmen. Für jeden Bildpunkt kann der stationäre Hintergrund aus dem Mittelwert der häufigsten Grauwerte, die an dem Bildpunkt in der Videosequenz aufgetreten sind, berechnet werden. So kann ein Bild konstruiert werden, in dem keine bewegten Objekte enthalten sind. Dieser Ansatz wird zum Beispiel von Shio et al. [Shio et al. 91] eingesetzt, um Personen aus einer Viseosequenz zu extrahieren. Ein zeitlicher Tiefpaßfilter kann ebenfalls eingesetzt werden, um die Pixelwerte des Hintergrunds zu bestimmen [Donohoe et al. 1988].

Durch das Bilden des Differenzbildes aus einem Bild der Sequenz und dem Bild mit dem stationären Hintergrund, können alle Veränderungen, die durch bewegte Objekte oder durch Beleuchtungsänderungen hervorgerufen wurden, extrahiert werden. Doch auch mit diesem Ansatz ist ein wesentliches Problem noch nicht gelöst. Hat ein Punkt eines bewegten Objektes zufällig den gleichen Grauwert wie der stationäre Hintergrund an dieser Stelle, dann wird die Veränderung im Differenzbild nicht detektiert.

3.2 Morphologische Operationen

Wenn ein Differenzbild zur Detektion von Veränderungen eingesetzt wird, kann es vorkommen, daß Bereiche innerhalb eines Objektes keine Veränderungen zeigen, obwohl sich das Objekt bewegt hat. Dieser Fall tritt auf, wenn sich der Grauwert eines Bildpunktes trotz Bewegung des Objektes nicht verändert hat. Diese Punkte fehlen im Differenzbild. Bei einer Binarisierung entstehen so Löcher in den Flächen. Diese Löcher könnten durch die Anwendung von

morphologischen Operationen aufgefüllt werden. Die Funktionsweise morphologischer Operationen ist ausführlich in Lehrbüchern von Jain et al. [Jain et al. 95] oder Gonzales et al. [Gonzales et al. 92] beschrieben.

Für die Anwendung morphologischer Operationen muß jedoch ein Strukturelement definiert werden. Daher stellt sich die Frage, wie groß dieses Strukturelement sein soll. Es muß groß genug sein, damit die Löcher geschlossen werden. Allerdings sollte es nicht zu groß sein, damit benachbarte Objekte nicht miteinander verschmolzen werden. Ist die Größe der betrachteten bewegten Objekte bekannt, so kann eventuell ein geeignetes Strukturelement gefunden werden.

3.3 Kontrast- und geschwindigkeitsbasierter Ansatz

Thompson [Thompson 80] verwendet Kontrast- und Geschwindigkeitsinformationen um einzelne Regionen des Bildes zu einem Objekt zusammenzufassen. Die Geschwindigkeit einzelner Bildpunkte werden durch ein von Fennema et al. [Fennema et al. 79] entwickeltes Verfahren berechnet.

Fennema et al. [Fennema et al. 79] setzen die Hough-Transformation [Bässmann et al. 89] ein, um Geschwindigkeiten von bewegten Objekten zu bestimmen (*Gradient-Intensity Transform Method*). Das Verfahren ist jedoch nur für translatorisch bewegte starre Körper geeignet. Für das Verfahren setzen sie voraus, daß der Gradient eines Punktes (x, y) sich mit der Zeit kaum verändert. Daher glätten Fennema et al. das Bild mit einem Tief-Paß-Filter und betrachten nur Punkte, für die gilt:

$$\begin{aligned} ||\nabla B(x,y,t_1)| - |\nabla B(x,y,t_2)|| &< \tau_1 \\ |\Theta(\nabla B(x,y,t_1)) - \Theta(\nabla B(x,y,t_2))| < \tau_2 \quad &\text{oder} \\ 2\pi \quad - \quad &|\Theta(\nabla B(x,y,t_1)) - \Theta(\nabla B(x,y,t_2))| < \tau_2 \\ \left|\frac{\partial}{\partial t} B(x,y,t)\right| &> \tau_3 \end{aligned}$$

Dabei gibt $\Theta(\nabla B(x, y, t))$ die Richtung des Gradienten am Punkt (x, y) zum Zeitpunkt t an [Bässmann et al. 89]. τ_1, τ_2 und τ_3 sind Schwellwerte. Die möglichen Geschwindigkeiten $v_{x,y}$ eines Bildpunktes (x, y), liegen dann entlang der durch

$$|v_{x,y}(\theta_v)| = -\frac{\frac{\partial}{\partial t} B(x,y,t)}{|\nabla B(x,y,t)| \cos(\Theta(\nabla B(x,y,t)) - \theta_v)}, \quad 0 \leq \theta_v \leq 2\pi$$

beschriebenen Kurve. Dabei gibt θ_v die Richtung der Geschwindigkeit an. Für verschiedene Werte der Gradientenrichtung ergibt sich jeweils eine Kurve. Die Schnittmenge der Kurven geben die Geschwindigkeit des Objektes an. Nachdem alle Bildpunkte transformiert wurden, geben die Maxima im Parameterraum der Hough-Transformation die Geschwindigkeiten der Objekte im Bild an. Damit die Geschwindigkeiten der Objekte exakt bestimmt werden können, müssen die Maxima deutlich hervortreten.

Nachdem die Maxima aus dem Parameterraum ermittelt wurden, weisen Fennema et al. [Fennema et al. 79] den einzelnen Punkten ihre Geschwindigkeit zu. Für jedes Maxima $\hat{v}$ aus dem Parameterraum weisen sie den Punkten (x, y) für die gilt:

$$\left| |\hat{v}| \cos(\Theta(\nabla B(x,y,t)) - \theta_{\hat{v}}) + \frac{\frac{\partial}{\partial t} B(x,y,t)}{|\nabla B(x,y,t)|} \right| < \tau_4$$

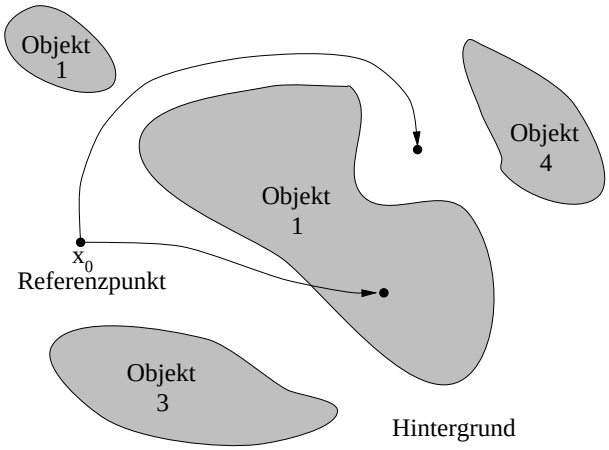

Abbildung 3.1: Vier Objekte vor einem Hintergrund. Der Referenzpunkt x_0 gehört zum Hintergrund (angepaßt nach [Bichsel 1994]).

die Geschwindigkeit $\hat{v}$ zu, wobei τ_4 ein Schwellwert ist.

Nachdem die Geschwindigkeiten einzelner Punkte bestimmt wurde, faßt Thompson [Thompson 80] einzelne Grauwertregionen zu bewegten Objekten zusammen. Dabei berücksichtigt Thompson sowohl die Grauwerte der Regionen als auch die Geschwindigkeit. Regionen mit unterschiedlicher Geschwindigkeit werden nur dann zusammengefaßt, wenn sie aufgrund ihrer Grauwerte ähnlich genug sind. Besitzen zwei benachbarte Regionen eine Geschwindigkeit ungleich Null, so werden sie unabhängig von ihrem Grauwert zusammengefaßt, wenn beide Geschwindigkeiten gleich sind.

3.4 Wahrscheinlichkeitsbasierter Ansatz

Von Bichsel [Bichsel 1994] wurde ein iteratives Verfahren entwickelt, um bewegte Objekte aus einer mit stationärer Kamera aufgenommenen Szene zu extrahieren. Es basiert auf der experimentellen Tatsache, daß das Histogramm einer Ableitung eines Bildes eine Laplace-Verteilung zeigt. Das Verfahren setzt voraus, daß für die Bildsequenz folgendes gilt:

- Einfach miteinander verbundene Objekte bewegen sich vor einem einzigen, verbundenen und stationären Hintergrund.
- Von mindestens einem Punkt ist bekannt, daß dieser zum Hintergrund gehört.

Durch die folgenden Konditionen kann eindeutig entschieden werden, ob ein Punkt (x, y) Teil eines Objektes ist, oder ob er Teil des Hintergrundes ist. Abbildung 3.1 dient zur Visualisierung der Konditionen.

- Der Punkt (x_0, y_0) gehöre zum Hintergrund
- Für jeden Punkt des Hintergrundes (x_0, y_0) existiert mindestens ein Pfad von (x_0, y_0) nach (x, y), der keinen Punkt des Objektes berührt.

- Jeder Pfad von (x_0, y_0) zu jedem Punkt (x, y) eines Objektes berührt mindestens einen Punkt des Objektes. Der erste Punkt des Objektes, der auf diesem Pfad liegt, ist ein Punkt auf der Kontur des Objektes. Somit hat dieser Punkt mindestens einen Punkt des Hintergrundes als Nachbar.

Als Punkte, von denen bekannt ist, daß sie zum Hintergrund gehören, wählte Bichsel für die von ihm betrachtete Videosequenz den oberen Rand des Bildes.

Das von Bichsel [Bichsel 1994] entwickelte Verfahren berechnet die Wahrscheinlichkeiten $p(b(x,y)|B)$. Mit $b(x,y)$ wird angegeben, daß der Punkt (x,y) zum Hintergrund gehört. Daher ist $p(b((x,y))|B)$ die Wahrscheinlichkeit, daß der Punkt (x,y) zum Hintergrund des Bildes B gehört. $G_t(x,z,t) = G(x,y,t+1) - G(x,y,t)$ sei die zeitliche Ableitung des Gradientenbildes $G = |\nabla B|$. Die Nachbarschaftsrelation der Bildpunkte wird durch N beschrieben. Bichsel verwendet eine vierer Nachbarschaft in horizontaler und vertikaler Richtung. Dann gibt $b(N(x,y))$ an, daß ein Punkt aus der Nachbarschaft von (x,y) zum Hintergrund gehört. Die Wahrscheinlichkeit $p(b(x,y)|G_t(x,y), b(N(x,y)))$ gibt an, daß der Punkt (x,y) zum Hintergrund gehört, wenn bekannt ist, daß die zeitliche Ableitung den Wert $G_t(x,y)$ hat, und mindestens ein Punkt aus der Nachbarschaft von (x,y) zum Hintergrund gehört. Die Wahrscheinlichkeiten $p(b(x,y)|B)$ werden iterativ nach folgendem Verfahren berechnet.

- Berechnung der zeitlichen Ableitung G_t des Gradientenbildes G.
- Initialisierung der lokalen Hintergrundwahrscheinlichkeiten mit

$$\ln(p(b(x,y)|G_t(x,y), b(N(x,y)))) \approx \begin{cases} -|\mu| \cdot (|G_t| - \tau) & \text{falls} \quad |G_t| \geq \tau \\ 0 & \text{falls} \quad |G_t| \leq \tau \end{cases}.$$

  Der Logarithmus der Wahrscheinlichkeit $p(b(x,y)|G_t(x,y), b(N(x,y)))$ wind von Bichsel durch eine Funktion aus zwei Geraden approximiert. Die zweite Gerade beginnt an der Position τ und hat die Steigung $-|\mu|$. τ bestimmt also den Übergangspunkt und kann aus experimentellen Daten bestimmt werden. Der Parameter μ stellt lediglich eine Skalierung der berechneten Wahrscheinlichkeiten dar und kann in die unten folgende Binarisierung integriert werden.

- Iterativ werden die lokalen Hintergrundwahrscheinlichkeiten $p(b(x,y)|B)$, die auf globalen Informationen des Bildes basieren, entsprechend

$$p(b(x,y)|B) = \max_{(x',y') \in N(x,y)} \ln(p(b(x',y')|B)) + \ln(p(b(x,y)|G_t(x,y), b(N(x,y))))$$

  angepaßt.

Nachdem einige Iterationen des Algorithmuses erfolgt sind, wird durch einen Schwellwert bestimmt, welche Punkte zum Hintergrund und welche Punkte zu einem Objekt gehören. Bichsel berichtet von typischen fünf Iterationen, falls ein Gauß-Seidel-Verfahren angewandt wird, bei dem die neuen berechneten Wahrscheinlichkeiten bereits in derselben Iteration weiterverwendet werden.

Westberg [Westberg 92] beschreibt ein hierarchisches, kontur-basiertes Verfahren, die Bildpunkte eines Bildes in drei Klassen einzuteilen. Als Klassen definiert Westberg: Innen, Rand und Außen. Das Verfahren basiert auch auf Wahrscheinlichkeiten und setzt voraus, daß die Bildsequenz mit stationärer Kamera aufgenommen wurde.

3.5 Zuordnung der Regionen zu Bildern

Das Differenzbild zeigt Regionen, in denen eine Veränderung stattgefunden hat. Verschiedene Regionen können durch ein bewegtes Objekt erzeugt worden sein. Im folgenden wird ein Verfahren besprochen, mit dem die Regionen den Objekten zugeordnet werden können. Zunächst werden die verschiedenen Typen der Regionen besprochen, die im Differenzbild entstehen können. Jain et al. [Jain et al. 79a] teilen die Regionen des Differenzbildes wie folgt ein.

- Typ O (*Object Growing*)

  Die Region, die durch das Bedecken des Hintergrundes mit dem Objekt entsteht, wird als Typ O bezeichnet, weil das Objekt in diese Region "hineinwächst".

- Typ B (*Background Growing*)

  Die Region, die durch das Freiwerden des Hintergrundes entsteht, wird als Typ B bezeichnet, weil dort der Hintergrund "wächst" oder größer wird.

- Statisch oder dynamisch

  Hat sich das Objekt während der Zeit, in der die Differenzbilder aufgenommen wurden, soweit bewegt, daß es sich im Differenzbild nicht überlappt, so werden die entstehenden Regionen in statisch und dynamisch eingeteilt. Als statisch wird die Region des Differenzbildes bezeichnet, die zur Objektposition im vorhergehenden Bild gehört. Als dynamisch wird die Region des Differenzbildes bezeichnet, die zur Objektposition im aktuellen Bild gehört.

Jain et al. [Jain et al. 79b] definieren zu den Regionen O und B noch den Typ X auf dem Differenzbild DB.

- Typ X (Background Growing and Object Growing)

  Die Region, die sowohl durch das Bedecken des Hintergrundes mit dem Objekt als auch durch das Freiwerden des Hintergrundes entsteht, wird als Typ X bezeichnet.

Jain et al. finden die Art der Region durch die Berechnung des folgenden Verhältnisses heraus.

$$\frac{CC}{CP}$$

mit

CC = Anzahl der Punkte auf der äußeren Kontur einer Region des Differenzbildes DB, die zugleich Kantenpunkte im aktuellen Bild sind.

CP = Anzahl der Punkte auf der äußeren Kontur einer Region des Differenzbildes DB, die zugleich Kantenpunkte im vorhergehenden Bild sind.

Die Region R läßt sich wie folgt zuordnen [Jain et al. 79b] (Abbildung 3.2).

$$\text{Typ}(R) = \begin{cases} B & \text{falls} \quad \frac{CC}{CP} \ll 1 \\ X & \text{falls} \quad \frac{CC}{CP} \approx 1 \\ O & \text{falls} \quad \frac{CC}{CP} \gg 1 \end{cases}$$

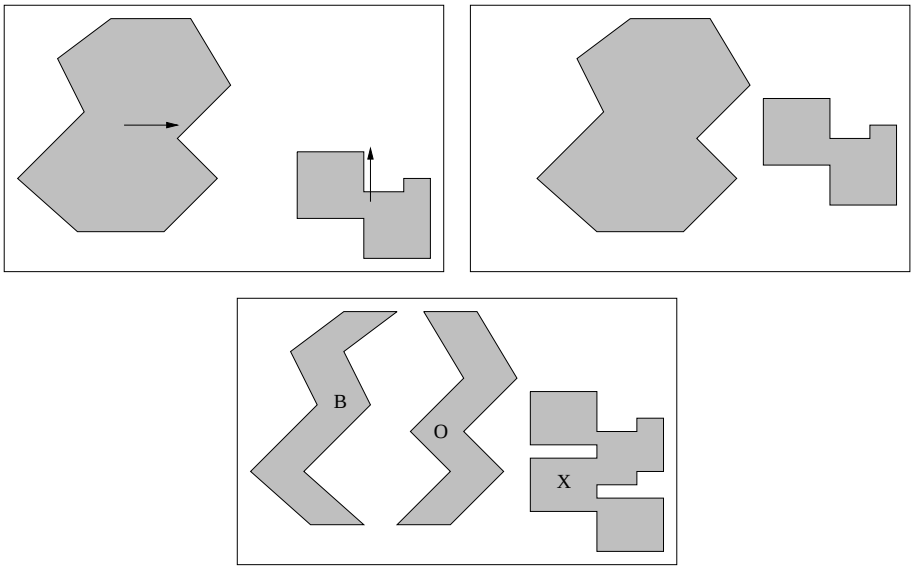

Abbildung 3.2: Zuordnung von Regionen des Differenzbildes zu einem von Jain et al. [Jain et al. 79b], [Jain et al. 79a] definierten Typ.

Jain et al. extrahieren dann das Objekt O aus dem vorhergehenden Bild, indem sie die Region R mit Typ$(R) = B$ durch Grauwertsegmentierung im vorhergehenden Bild vergrößern. Für eine Region R mit Typ$(R) = O$ wenden sie die Segmentierung auf dem aktuellen Bild an und extrahieren das Objekt dort. Regionen R mit Typ$(R) = X$ werden verkleinert und so auf das Objekt im Bild angepaßt. Dabei wird die Verkleinerung der Kontur ausgehend von den Punkten der Kontur, die nicht mit einer Kante im Bild zusammenfallen, bis zur Kante des Objektes vorgenommen. Die Kontur wird so in horizontaler und vertikaler Richtung verkleinert. Das Verfahren setzt voraus, daß der Hintergrund relativ homogen ist. Anstatt eine Segmentierung pixelweise vorzunehmen, könnten die Regionen auch durch Setzen eines Schwellwertes [Milgram 79] extrahiert werden.

3.6 Geometrischer Ansatz

Yalamanchili et al. [Yalamanchili et al. 82] verwenden geometrische Eigenschaften der Regionen eines Differenzbildes, um die einzelnen Regionen zu bewegten Objekten zusammenzufassen. Von den Objekten wird angenommen, daß es Polygone sind. Das Zeitintervall zwischen zwei Bildern sei so klein, daß sich kein Objekt vollständig von seinem Platz bewegt. Die Grauwerte eines Polygons seien homogen. Yalamanchili et al. [Yalamanchili et al. 82] unterscheiden zwei Arten der Bewegung.

- Bewegung des Objektes erfolgt unabhängig von der Form und Orientierung des Objektes.
- Bewegung des Objektes erfolgt abhängig von der Form und Orientierung des Objektes.

Für beide Fälle haben Yalamanchili et al. [Yalamanchili et al. 82] Verfahren entwickelt, die Bereiche des Differenzbildes zusammenzufassen. Die Verfahren werden nacheinander auf das

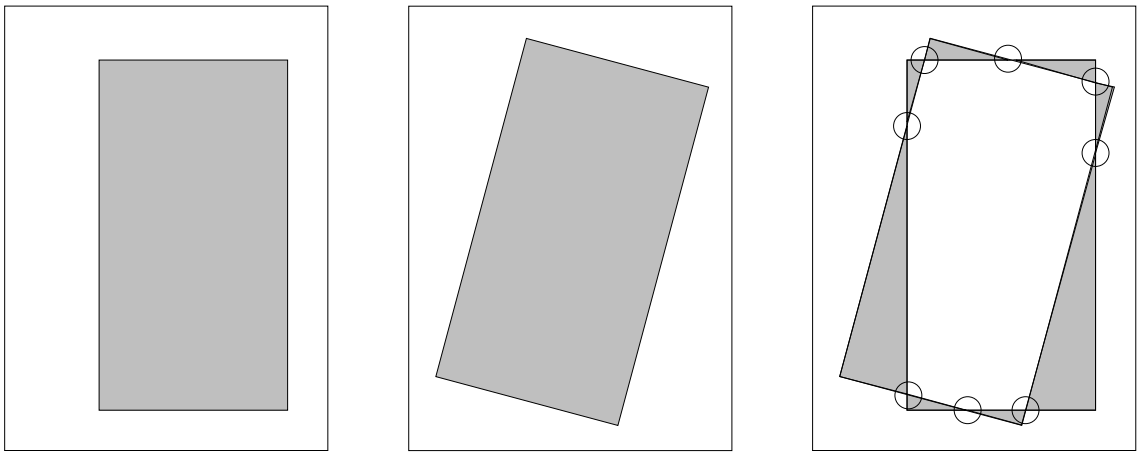

Abbildung 3.3: Bewegung unabhängig von Form und Orientierung des Objektes. Die durch Kreise markierten Punkte sind nicht im Differenzbild enthalten, da sie Teil des Objektes im vorhergehenden und im aktuellen Bild sind (angepaßt nach [Yalamanchili et al. 82]).

Differenzbild angewandt.

Bewegt sich das Objekt unabhängig von seiner Form und Orientierung, dann ist keine Seite des Objektes parallel zur Bewegungsrichtung. Das Differenzbild dieses bewegten Objektes ist dann fast eine geschlossene Kontur. Allerdings ist die Kontur an einigen Punkten unterbrochen. Für ein drehendes Objekt ist dies in Abbildung 3.3 dargestellt. Die durch Kreise markierten Punkte sind nicht im Differenzbild enthalten. Diese Punkte liegen auf der Kante des Objektes und sie gehören zum Objekt im vorhergehenden und im aktuellen Bild. Daher sind die einzelnen Regionen an diesen Stellen voneinander getrennt. Um die Regionen aus dem Differenzbild an diesen Stellen wieder zusammenzufügen, werden die Punkte aus der Schnittmenge der Kanten betrachtet. Falls nötig, werden sie zum Differenzbild hinzugefügt. Schließlich wird die Kontur der so zusammengefaßten Regionen bestimmt und diese Regionen aus dem Differenzbild entfernt. Die verbleibenden Regionen werden mit dem folgenden Verfahren zusammengefaßt. Bei ihnen wird eine Bewegung des Objektes in Abhängigkeit von der Form und Orientierung des Objektes angenommen.

Bewegt sich das Objekt abhängig von seiner Form und Orientierung, dann ist mindestens eine Kante des Objektes parallel zur Bewegungsrichtung. Ein Beispiel für ein Objekt, das sich parallel zu zwei Kanten des Objektes bewegt, ist in Abbildung 3.4 gegeben. Es entstehen in diesem Fall zwei getrennte Regionen im Differenzbild. Diese Regionen werden von Yalamanchili et al. [Yalamanchili et al. 82] aufgrund geometrischer Überlegungen zu einem Objekt zusammengefaßt. Dazu wird im Differenzbild nach colinearen Kanten im Differenzbild gesucht. Die Länge dieser Kanten ist ein Maß für die Geschwindigkeit des Objektes. Daher werden die zwei colineare Kanten nur dann zusammengefaßt, wenn die Kanten ungefähr gleich lang sind. In Abbildung 3.4 werden also die gestrichelten Linien eingefügt und so die Kontur des Objektes extrahiert.

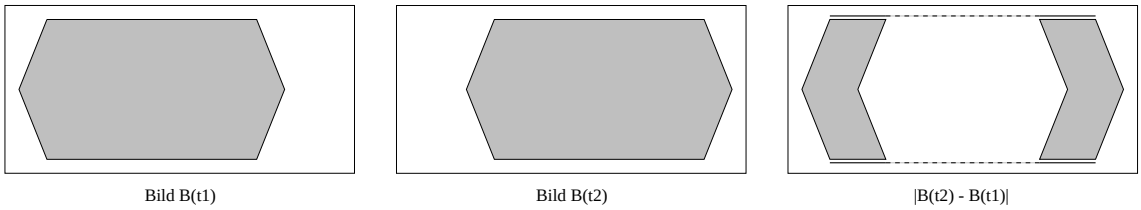

Abbildung 3.4: Bewegung abhängig von Form und Orientierung des Objektes. Die Regionen des Differenzbildes können durch Verbinden colinearer, gleichlanger Kanten zusammengefaßt werden (angepaßt nach [Yalamanchili et al. 82]).

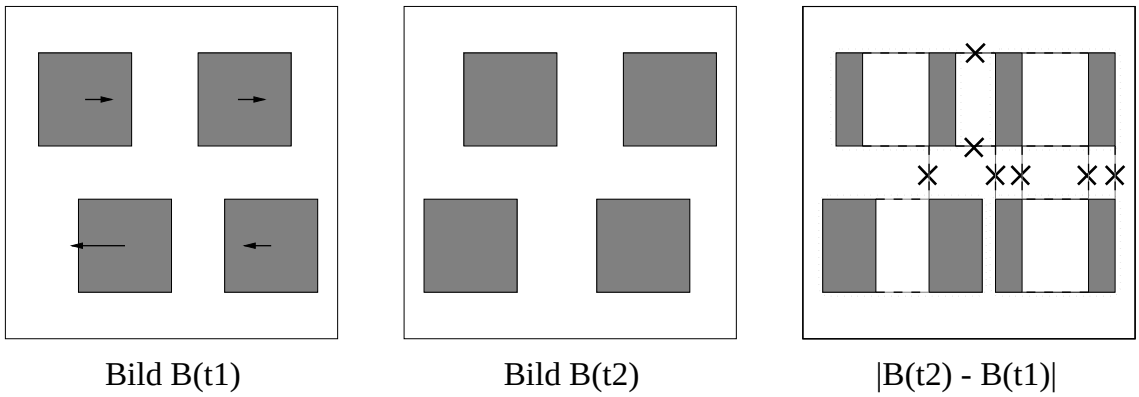

Abbildung 3.5: Konstruiertes Beispiel, bei dem der Ansatz von Yalamanchili [Yalamanchili et al. 82] fälschlicherweise Verbindungslinien zieht. Die falschen Verbindungslinien sind im Differenzbild durchgestrichen.

Nachdem zusammenhängende Regionen extrahiert wurden, verwenden Yalamanchili et al. [Yalamanchili et al. 82] noch Region-Growing um die exakte Kontur des Objektes zu extrahieren.

Der Ansatz, Regionen des Differenzbildes durch geometrische Überlegungen zusammenzufassen, ist für einen effizienten Algorithmus geeignet. Es wird die Anzahl der betrachteten Bildpunkte reduziert. Yalamanchili et al. [Yalamanchili et al. 82] verwenden noch einen Region-Growing-Schritt in ihrem Algorithmus. Region-Growing ist in der Regel eine sehr zeitintensive Operation. In diesem Fall ist sie jedoch nur auf die Bereiche der bewegten Objekte beschränkt. Also muß nicht das gesamte Bild segmentiert werden.

In Abbildung 3.5 ist noch ein konstruiertes Beispiel aufgeführt. Es enthält vier bewegte Objekte. Der Algorithmus von Yalamanchili et al. würde die einzelnen Regionen des Differenzbildes entlang der gestrichelten Linien miteinander verbinden. Die Verbindungslinien, die aufgrund der tatsächlichen Zugehörigkeit der Regionen zu den Objekten nicht hätten eingezeichnet werden dürfen, sind durchgestrichen. Der Algorithmus hätte diese Verbindungslinien aber fälschlicherweise gezogen.

3.7 Zusammenfassen von Regionen durch Prädiktion der Objektbewegung

Sugimoto et al. [Sugimoto et al. 86] verwenden die Prädiktion der Objektbewegung, um eine Region verschiedenen Objekten zuzuordnen. So können sie mehrere bewegte Objekte verfolgen, die sich auch gegenseitig verdecken können. Zunächst berechnen Sugimoto et al. die Position der bewegten Objekte aufgrund ihres Bewegungsvektors. Der Bewegungsvektor gibt die Bewegung des Objektes in der Bildebene an.

Bei einer Verdeckung zweier oder mehrerer Objekte trennen sie die Region, indem sie die Prädiktion der Objektbewegung nutzen, den Bereich zu bestimmen, der durch die Überlappung der Objekte entstanden ist. Sugimoto et al. vergrößern die Fläche der vorhergesagten Objektpositionen, falls notwendig, um auch Bereiche, die außerhalb der Prädiktion liegen, den Objekten zuzuordnen.

3.8 Bewertung der Ansätze

Nun soll besprochen werden, welche der in diesem Kapitel vorgestellten Verfahren für die vorliegende Arbeit geeignet sind, die Regionen des Differenzbildes zu Objekten zusammenzufassen. Die Subtraktion des Hintergrundes ist vor allem dann geeignet, wenn die Videosequenz mit einer stationären Kamera aufgenommen wurde. In der vorliegenden Arbeit sollen aber bewegte Objekte aus einer mit bewegter Kamera aufgenommenen Videosequenz extrahiert werden. Zwar wird die Eigenbewegung der Kamera kompensiert, doch ist es nicht möglich, den stationären Hintergrund aus einigen wenigen Bildern zu berechnen. Vor allem bei großen Kamerabewegungen ist dies schwierig. Dann ist ein stationärer Punkt in nur wenigen Bildern der Videosequenz im Sichtbereich der Kamera. Daher sind nicht genügend Werte für einen Bildpunkt vorhanden, um daraus den stationären Hintergrund zu bestimmen. Außerdem ist eine völlig exakte Kompensation der Eigenbewegung nicht möglich, wenn mit ungenauen Statusinformationen gearbeitet wird.

Morphologische Operationen könnten eingesetzt werden, um Löcher aus den Regionen des Differenzbildes zu beseitigen. Doch hier stellt sich das Problem der Größe des Strukturelementes. Sequentielle Verfahren, die auf der Segmentierung des Bildes basieren, benötigen sehr viel Rechenzeit. Daher werden in der vorliegenden Arbeit die Bilder nicht segmentiert. Ein einfacher Segmentierungs Algorithmus, wie er zum Beispiel in Horn [Horn 86] beschrieben wird, hätte noch durch Aufteilen und Zusammenfügen der Regionen wie es zum Beispiel von Jain et al. [Jain et al. 95] beschrieben wird, verbessert werden müssen. Doch dies benötigt zusätzliche Rechenzeit.

Der Ansatz von Bichsel [Bichsel 1994] setzt voraus, daß von mindestens einem Punkt bekannt ist, daß dieser zum Hintergrund gehört. Die Verfahren von Bichsel [Bichsel 1994] und Westberg [Westberg 92] sind nur für eine mit stationärer Kamera aufgenommenen Videosequenz geeignet. In der vorliegenden Arbeit wird die Eigenbewegung der Kamera zwar kompensiert. Trotzdem verbleiben Ungenauigkeiten in der Prädiktion der Bildbewegung. Der Ansatz, die Regionen des Differenzbildes den Regionen des vorhergehenden bzw. des aktuellen Bildes zuzuordnen, ist ebenfalls nur bei exakter Kompensation der Eigenbewegung der Kamera geeignet.

Der von Yalamanchili et al. [Yalamanchili et al. 82] entwickelte geometrische Ansatz ist vor allem für eine effiziente Implementierung geeignet. Er wird in abgewandelter Form auch in der

vorliegenden Arbeit eingesetzt werden. Der Ansatz von Sugimoto et al. [Sugimoto et al. 86], die Prädiktion der Objektbewegung zu nutzen, wird auch hier verfolgt.

Kapitel 4

Extraktion bewegter Objekte

Im folgenden wird nun der für die vorliegende Arbeit in Zusammenarbeit mit den Betreuern Gerl und Oswald [Oswald et al. 96] entwickelte Algorithmus beschrieben. Entsprechend dem zielorientierten Ansatz [Aloimonos 93] werden nur die Daten berechnet, die zur Lösung des Problems erforderlich sind. Es wird davon ausgegangen, daß die Kamera, die die Bildsequenz aufnimmt, für einen bestimmten Zweck eingesetzt wird. Die Kamera besitzt also einen Kontrollalgorithmus, der ihre Bewegung steuert. Somit stehen die gewünschten oder tatsächlichen Geschwindigkeiten oder Positionen, der bei der Steuerung eingesetzten Motoren, zur Verfügung. Aus diesen Daten kann die Eigenbewegung der Kamera approximiert werden. Ist die Eigenbewegung der Kamera bekannt, so kann auf der Bildsequenz wie bei einer stationären Kamera weitergearbeitet werden. Da bei einer bewegten Kamera Schwingungen auftreten können, wird noch eine Fehlerkorrektur eingesetzt, um diese zu eliminieren.

4.1 Kompensation der Eigenbewegung

Wenn die Eigenbewegung der Kamera erfolgreich kompensiert werden kann, dann wird die Extraktion bewegter Objekte auf das Problem reduziert, die Objekte aus einer mit stationärer Kamera aufgenommenen Videosequenz zu extrahieren. Interessant ist, daß einige Tiere auch ihre Eigenbewegung kompensieren [Carpenter 88]. Ein Huhn oder eine Ente kompensieren ihre Eigenbewegung, indem sie versuchen, ihren Kopf bei einer Vorwärtsbewegung ruhig zu halten, und ihn dann schließlich mit einer schnellen Bewegung vorschnellen zu lassen. Da die Augen seitwärts gerichtet sind, entsteht ohne Kompensation der Eigenbewegung eine kontinuierliche Verschiebung des Bildes. Bei Augen, die nach vorne ausgerichtet sind, ist dieses Problem nicht so groß.

Einige Tiere haben einen speziellen Gang entwickelt, um die Eigenbewegung zu kompensieren. Da in der vorliegenden Arbeit kein Einfluß auf die Bewegung der Kamera genommen wird, erfolgt die Kompensation der Eigenbewegung der Kamera durch Herausrechnen der Bewegung des Bildes. Komprimierungsalgorithmen für Bildsequenzen führen ebenfalls eine Vorhersage des Bildes durch und versuchen die Eigenbewegung zu kompensieren [Gall 91]. Dazu modellieren sie die Bildbewegung lokal als einfache Translation. Ist die Eigenbewegung der Kamera und die Entfernung der Objektpunkte des Bildes bekannt, so kann die Bewegung des Bildes aufgrund dieser Eigenbewegung vorhergesagt werden (Gleichung 2.7). Die Prädiktion der Bildbewegung soll nun besprochen werden.

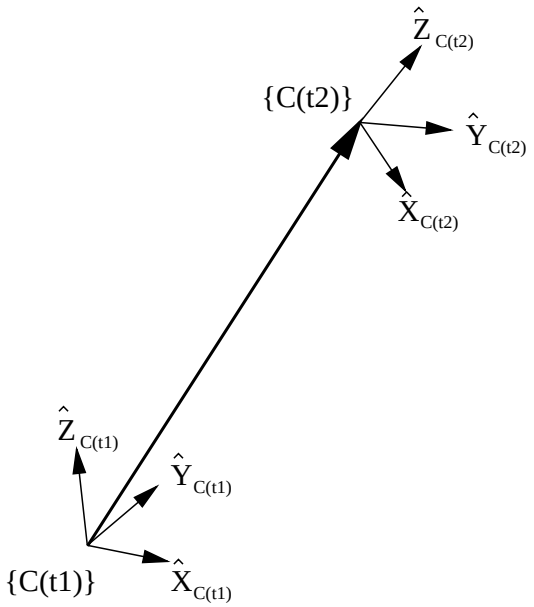

Abbildung 4.1: Translatorische und rotatorische Kamerabewegung.

4.1.1 Prädiktion der Bewegung des stationären Hintergrundes

Es wird eine Transformationsvorschrift für einen Punkt, der sich zum Zeitpunkt t_1 an der Position $(x(t_1), y(t_1))$ im Bild befindet, auf die Position zum Zeitpunkt t_2 berechnet. Der Punkt werde durch ein stationäres Objekt, also durch den Hintergrund erzeugt. Der Punkt an der Position $(x(t_1), y(t_1))$ werde durch den Punkt ${}^{W}\mathbf{P} = [X, Y, Z]^T$ erzeugt. Zum Zeitpunkt t_2 hat sich der Punkt durch die Bewegung der Kamera auf den Punkt an der Position $(x(t_2), y(t_2))$ im Bild bewegt. Im Koordinatensystem der Kamera zum Zeitpunkt t_1 hat dieser Punkt die Koordinaten ${}^{C(t_1)}\mathbf{P} = [X(t_1), Y(t_1), Z(t_1)]^T$. Zum Zeitpunkt t_2 hat derselbe Punkt die Koordinaten ${}^{C(t_2)}\mathbf{P} = [X(t_2), Y(t_2), Z(t_2)]^T$.

Die Transformationsmatrix ${}^{C(t_2)}_{C(t_1)}T$ wird als bekannt vorausgesetzt (Abbildung 4.1). Die Koordinaten des Punktes ${}^{C(t_1)}\mathbf{P}$ transformieren sich also wie folgt [Craig 89]:

$$ {}^{C(t_2)}\mathbf{P} = {}^{C(t_2)}_{C(t_1)}T \,.\, {}^{C(t_1)}\mathbf{P} \tag{4.1} $$

$$ \begin{bmatrix} X(t_2) \\ Y(t_2) \\ Z(t_2) \\ 1 \end{bmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \varrho_x \\ r_{21} & r_{22} & r_{23} & \varrho_y \\ r_{31} & r_{32} & r_{33} & \varrho_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} X(t_1) \\ Y(t_1) \\ Z(t_1) \\ 1 \end{bmatrix} \tag{4.2} $$

Die Parameter r_{ij} bilden die 3×3 Rotationsmatrix der Transformation. Der Vektor $[\varrho_x, \varrho_y, \varrho_z]$ ist der Translationsvektor der Transformation. Nach Gleichung 4.2 gilt:

$$ \begin{aligned} X(t_2) &= r_{11}X(t_1) + r_{12}Y(t_1) + r_{13}Z(t_1) + \varrho_x \\ Y(t_2) &= r_{21}X(t_1) + r_{22}Y(t_1) + r_{23}Z(t_1) + \varrho_y \\ Z(t_2) &= r_{31}X(t_1) + r_{32}Y(t_1) + r_{33}Z(t_1) + \varrho_z \end{aligned} $$

Mit perspektivischer Projektion (Gleichung 2.1) und der Brennweite f ergibt sich folgende Abbildung [Murray et al. 94a].

$$\begin{aligned} x(t_2) &= f\frac{X(t_2)}{Z(t_2)} \\ &= f\frac{r_{11}X(t_1)+r_{12}Y(t_1)+r_{13}Z(t_1)+\varrho_x}{r_{31}X(t_1)+r_{32}Y(t_1)+r_{33}Z(t_1)+\varrho_z} \\ &= f\frac{r_{11}x(t_1)+r_{12}y(t_1)+fr_{13}+f\frac{\varrho_x}{\bar{Z}(t_1)}}{r_{31}x(t_1)+r_{32}y(t_1)+fr_{33}+f\frac{\varrho_z}{\bar{Z}(t_1)}} \end{aligned}$$

$$\begin{aligned} y(t_2) &= f\frac{Y(t_2)}{Z(t_2)} \\ &= f\frac{r_{21}X(t_1)+r_{22}Y(t_1)+r_{23}Z(t_1)+\varrho_x}{r_{31}X(t_1)+r_{32}Y(t_1)+r_{33}Z(t_1)+\varrho_z} \\ &= f\frac{r_{21}x(t_1)+r_{22}y(t_1)+fr_{23}+f\frac{\varrho_y}{\bar{Z}(t_1)}}{r_{31}x(t_1)+r_{32}y(t_1)+fr_{33}+f\frac{\varrho_z}{\bar{Z}(t_1)}} \end{aligned}$$

Ist die Bewegung der Kamera bekannt, so wird zur Transformation der Bildpunkte noch deren Abstand zur Kamera benötigt. Für große Entfernungen $Z(t_1)$ der Punkte des Hintergrundes kann die Translation völlig vernachlässigt werden. Für $Z(t_1) \longrightarrow \infty$ folgt:

$$\begin{aligned} \lim_{Z(t_1)\to\infty} x(t_2) &= f\frac{r_{11}x(t_1)+r_{12}y(t_1)+fr_{13}}{r_{31}x(t_1)+r_{32}y(t_1)+fr_{33}} \\ \lim_{Z(t_1)\to\infty} y(t_2) &= f\frac{r_{21}x(t_1)+r_{22}y(t_1)+fr_{23}}{r_{31}x(t_1)+r_{32}y(t_1)+fr_{33}} \end{aligned}$$

Diese Transformationsgleichungen sind exakt für den Fall, daß die Kamera nur eine Rotationsbewegung und keine Translationsbewegung durchführt. Murray et al. [Murray et al. 94a] führten diese Berechnungen zur Kompensation der Rotation einer Schwenk/Nick-Kamera durch.

In der vorliegenden Arbeit wird jedoch der Abstand der Punkte des Hintergrundes zur Kamera als bekannt vorausgesetzt. Die Entfernung der Kamera zum Hintergrund wird als unabhängig vom einzelnen Bildpunkt angenommen. Der Hintergrund befinde sich also zum Zeitpunkt t im Abstand $\bar{Z}(t)$ zur Kamera. Daher wird folgende Transformationsgleichung verwendet.

$$x(t_2) = f\frac{r_{11}x(t_1)+r_{12}y(t_1)+fr_{13}+f\frac{\varrho_x}{\bar{Z}(t_1)}}{r_{31}x(t_1)+r_{32}y(t_1)+fr_{33}+f\frac{\varrho_z}{\bar{Z}(t_1)}} \tag{4.7}$$

$$y(t_2) = f\frac{r_{21}x(t_1)+r_{22}y(t_1)+fr_{23}+f\frac{\varrho_y}{\bar{Z}(t_1)}}{r_{31}x(t_1)+r_{32}y(t_1)+fr_{33}+f\frac{\varrho_z}{\bar{Z}(t_1)}} \tag{4.8}$$

Diese Transformationsgleichung gibt das Bewegungsfeld [Horn 86] einer Ebene an, die sich im Abstand $\bar{Z}(t)$ zur Kamera befindet. Welche Auswirkungen diese Transformation auf das Bild

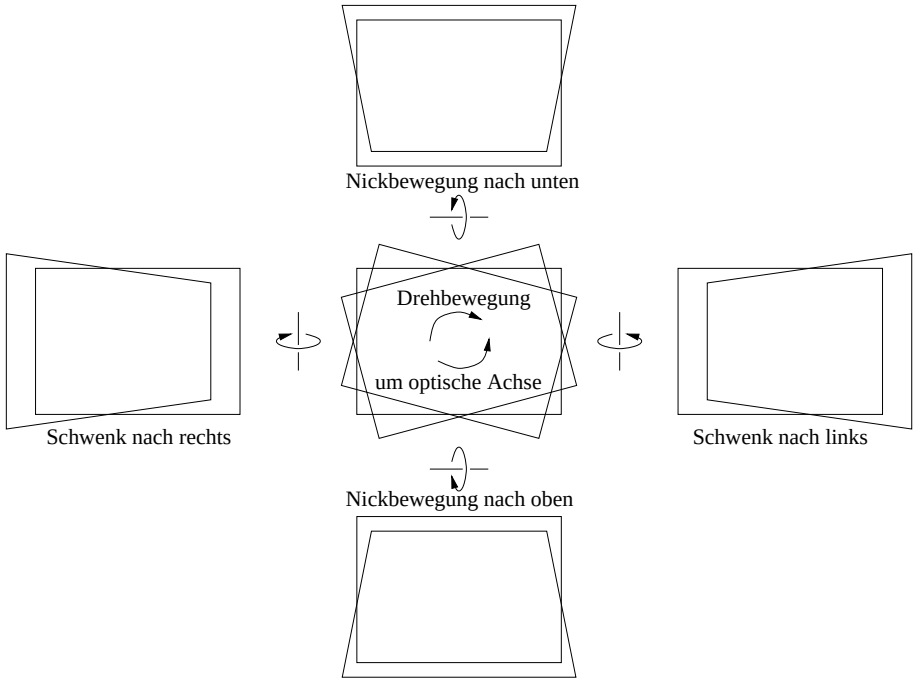

Abbildung 4.2: Transformation des Kamerabildes bei einer Drehbewegung um die optische Achse, bei einer Schwenkbewegung und bei einer Nickbewegung der Kamera.

hat, ist anhand einiger Beispiele in Abbildung 4.2 dargestellt. Zu sehen sind die Auswirkungen der Transformation auf das Bild in der Bildmitte bei einer Drehbewegung um die optische Achse. Bei einer Schwenkbewegung transformiert sich das Bild wie links und rechts in Abbildung 4.2 dargestellt. Oben und unten in Abbildung 4.2 sind die Auswirkungen der Transformation auf eine Nickbewegung der Kamera zu sehen.

4.1.2 Elimination von Oszillationen der Kamera

Die in Abschnitt 4.1.1 beschriebene Transformation ist aufgrund der als konstant angenommenen Entfernung nur eine Näherung der tatsächlichen Bewegung der Punkte des Hintergrundes. Es wurde ferner vorausgesetzt, daß die Transformationsmatrix T exakt berechnet werden kann. Die Transformationsmatrix wird aus Informationen berechnet, die durch den Kontrollalgorithmus des Fahrzeuges zur Verfügung stehen. Da diese Informationen nur eine bestimmte Genauigkeit besitzen, kann die Transformationsmatrix nicht völlig exakt berechnet werden.

Zudem ist die Kamera oft auf einem Fahrzeug oder Manipulator angebracht, der während seiner Bewegung kleine Schwingungen ausführt. Dieser Effekt ist besonders dann gegeben, wenn sich die Kamera am Ende eines Manipulators befindet, der wiederum auf einem Fahrzeug angebracht ist. Fährt das Fahrzeug über eine kleine Unebenheit im Boden, so beginnt der Manipulator auf dem Fahrzeug zu schwingen.

Um horizontale und vertikale Schwingungen der Kamera zu eliminieren wird ein lineares Fehlermodell verwendet. Die Prädiktion des Bildes wird noch um den Wert $(\Delta x, \Delta y)$ verschoben, um somit eine verbesserte Prädiktion auf das tatsächliche Bild zu erreichen. Durch diese

Art der Fehlerkorrektur werden Schwingungen eliminiert, aber auch ungenaue Daten. Denn für große Entfernungen $\bar{Z}$ ist die Verzerrung des Bildes bei einem Schwenk der Kamera oder einer Nickbewegung der Kamera relativ klein. Daher kann bei diesen Bewegungen eine einfache Verschiebung auch hier helfen, die Prädiktion zu verbessern. Rotationen um die optische Achse der Kamera können durch eine einfache Verschiebung des Bildes nicht korrigiert werden[1].

Translationen der Kamera horizontal und vertikal zur Blickrichtung der Kamera verschieben das Bild. Eine Translation der Kamera in Richtung der optischen Achse vergrößert oder verkleinert das Bild. Durch die Fehlerkorrektur wird nur die Translation in horizontaler oder vertikaler Richtung korrigiert. Eine Korrektur ist hier besonders dann wichtig, wenn die geschätzte Entfernung zum Hintergrund nicht mit der tatsächlichen Entfernung übereinstimmt. Ist die geschätzte Entfernung zu groß, so wird das Bild durch die Prädiktion nicht weit genug verschoben. Ist die geschätzte Entfernung zu nah, so wird das Bild über die tatsächliche Position hinaus verschoben. Durch die Fehlerkorrektur soll dieser Effekt verringert werden. Damit kann der Algorithmus bis zur Anpassung der geschätzten Entferung zum Hintergrund (bei einer erfolgreichen Berechung der erforderlichen Verschiebung des Bildes) weiterhin die Kamerabewegung kompensieren.

Zunächst wurde versucht, die Verschiebung des Bildes durch zeilen- und spaltenweise Korrelation zu bestimmen. Es wurde also eine eindimensionale Korrelation [Ancona et al. 95] durchgeführt. Dies Verfahren konnte nicht verwendet werden, da bei einem großen bewegten Objekt das Objekt und nicht der Hintergrund zur Deckung gebracht wurde.

Der Verschiebungsvektor der Prädiktion wird in der vorliegenden Arbeit aus dem Median der Verschiebung von korrespondierenden markanten Punkten berechnet. Dies hat sich als robuster Mechanismus zur bestimmung der erforderlichen Verschiebung des Bildes herausgestellt. Durch die Berechnung des Medians werden Ausreißer eliminiert [Jain et al. 95]. Dies setzt voraus, daß mehr markante Punkte, für die eine Korrespondenz hergestellt wurde, auf dem Hintergrund als auf den bewegten Objekten in der betrachteten Szene sind. Je mehr markante Punkte sich auf dem Hintergrund als auf den bewegten Objekten befinden, desto wichtiger ist die Fehlerkorrektur um die Prädiktion zu verbessern. Die Fehlerkorrektur kann nur angewendet werden, wenn markante Punkte in aufeinander folgenden Bildern existieren.

4.1.3 Berechnung der Entfernung zum Hintergrund

Bewegt sich eine Kamera, so kann aus der aufgenommenen Bildsequenz die Entfernung der Bildpunkte bestimmt werden [Williams 80]. Williams [Williams 80] modelliert die Szene im Blickfeld der Kamera als horizontale und vertikale Flächen. Dazu wird zunächst durch einen Segmentierungsprozess ein erstes Modell der Szene erstellt. Dieses Modell wird dann durch einen Suchprozess, der die Höhe der horizontalen Flächen und den Abstand zu den vertikalen Flächen anpaßt, verbessert.

Anstatt für jeden Punkt des Bildes die Tiefe zu berechnen, könnte eine Funktion verwendet werden, mit der für jeden Bildpunkt die Entfernung des zugehörigen Objektpunktes definiert wird. Ist die Kamera auf eine Wand gerichtet und befindet sich der Boden im Sichtbereich der Kamera, so könnten die Entfernungen der Objektpunkte zum Beispiel durch folgende Funktion

[1] Rotationen des Bildes um eine Achse senkrecht zur Bildebene treten bei der für die vorliegende Arbeit eingesetzten Kamera, nicht so häufig auf. Eine Rotation des Bildes tritt zum Beispiel dann auf, wenn die Kamera seitwärts ausgerichtet ist und eine Nickbewegung durchführt. Wird die Kamera auf die Decke ausgerichtet und dann eine Drehbewegung um die Schwenkachse ausführt, so tritt ebenfalls eine Drehung des Bildes um eine Achse senkrecht zur Bildebene auf. Allerdings ist diese Achse nicht im Zentrum des Bildes.

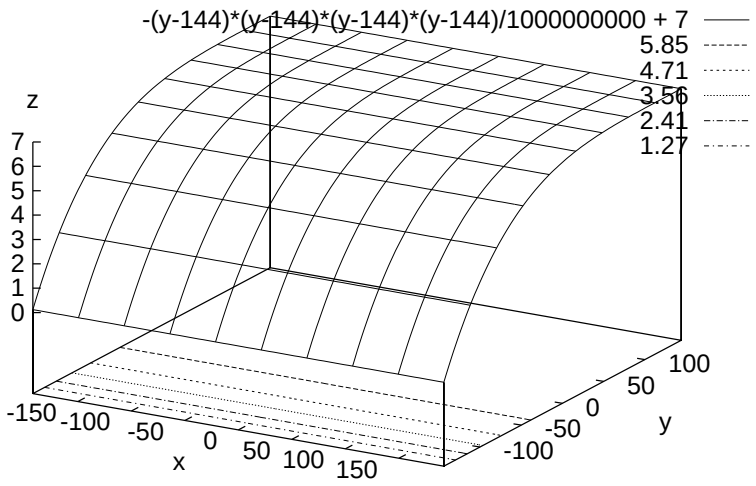

Abbildung 4.3: Abstandsfunktion für eine Wand, die jedem Bildpunkt (x,y) eine Entfernung zuordnet. Höhenlinien sind gestrichelt eingezeichnet.

modelliert werden.

$$Z(x,y) = a\left(y - \frac{h}{2}\right)^4 + b$$

Der Parameter h gibt die Höhe des Bildes an. Die Parameter a und b sind je nach betrachteter Szene zu bestimmen. Die Funktion ist für ein 384×288 großes Bild mit den Parametern $a = -1/1000000000$ und $b = 7$ in Abbildung 4.3 dargestellt.

Ein Korridor könnte zum Beispiel mit folgender Funktion modelliert werden

$$Z(x,y) = a(x^2 + y^2) + b$$

Die Parameter a und b sind je nach betrachteter Szene zu bestimmen. Für ein 384×288 großes Bild ist die Funktion in Abbildung 4.4 mit den Parametern $a = -1/5000$ und $b = 12$ dargestellt.

Für die vorliegende Arbeit wird ein einfacheres Modell des Hintergrundes angenommen. Der Hintergrund bestehe aus einer Fläche, die sich im konstanten Abstand $\bar{Z}$ zur Kamera befinde. Daher ist nur ein Parameter zu bestimmen. Dieses relativ einfache Modell wird hier verwendet, da es besonders für einen robusten Ansatz geeignet ist. Ist der Abstand der betrachteten Objektpunkte von der Kamera groß im Vergleich zur Tiefenvariation der Objektpunkte, dann können die Entfernungen der Objektpunkte durch eine Ebene approximiert werden. Der Abstand $\bar{Z}$ der Ebene kann aus der Bewegung markanter Punkte bestimmt werden. Dies setzt voraus, daß markante Punkte existieren und zwischen ihnen eine Korrespondenz hergestellt werden kann. Zur Berechnung der Entfernung ist eine möglichst genaue Transformationsmatrix T nötig. Die Transformationsmatrix kann aus den Statusinformationen des Kontrollalgorithmuses, der die Kamera steuert, berechnet werden. $(x(t_1), y(t_1))$ sei ein Punkt der betrachteten

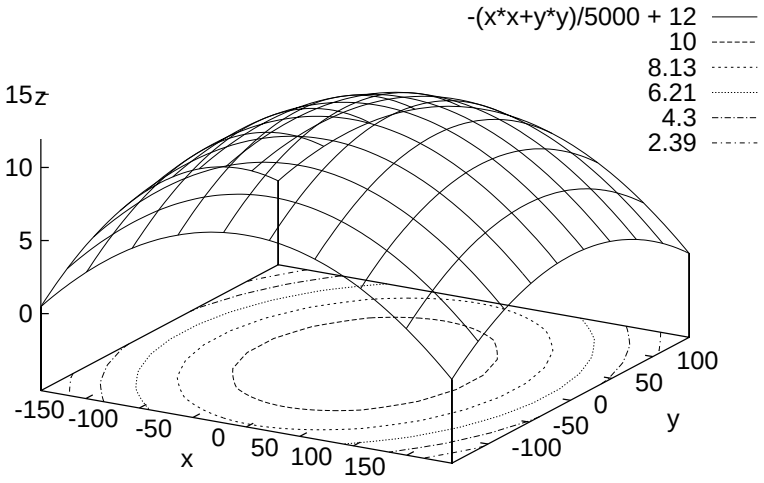

Abbildung 4.4: Abstandsfunktion für einen Gang, die jedem Bildpunkt (x,y) eine Entfernung zuordnet. Höhenlinien sind gestrichelt eingezeichnet

Ebene zum Zeitpunkt t_1. Derselbe Punkt habe zum Zeitpunkt t_2 die Koordinaten $(x(t_2), y(t_2))$ Dann läßt sich durch einfache Umformung der Gleichung 4.7 und Gleichung 4.8 der Abstand zu diesem Punkt berechnen.

$$\bar{Z}(t_1) = \frac{f\varrho_x - x(t_2)\varrho_z}{\frac{x(t_2)}{f}(r_{31}x(t_1) + r_{32}y(t_1) + fr_{33}) - (r_{11}x(t_1) + r_{12}y(t_1) + fr_{13})}$$

$$\bar{Z}(t_1) = \frac{f\varrho_y - y(t_2)\varrho_z}{\frac{y(t_2)}{f}(r_{31}x(t_1) + r_{32}y(t_1) + fr_{33}) - (r_{21}x(t_1) + r_{22}y(t_1) + fr_{23})}$$

Diese Methode der Abstandsbestimmung wird für die vorliegende Arbeit verwendet. Sie kann nur für solche Bildpunkte angewandt werden, die sich aufgrund der Bewegung des Fahrzeuges bewegt haben. Da das Bild digitalisiert wurde, muß eine Bewegung des Fahrzeuges nicht unbedingt eine Bewegung der Bildpunkte verursachen. Daher muß nach einem Kriterium gesucht werden, das angibt, wann die Z-Koordinate berechnet wird. Dieses Kriterium wird in Kapitel 5 beschrieben.

Im Bild können sich aber auch bewegte Objekte befinden. Daher kann es vorkommen, daß zwei korrespondierende Punkte eines bewegten Objektes verwendet werden, um die Z-Koordinate zu berechnen. Diese Z-Koordinate kann natürlich nicht zur Berechnung der durchschnittlichen Entfernung zum Hintergrund verwendet werden. Deshalb wird folgendes Verfahren angewandt, um solche Werte zu eliminieren. Die durchschnittliche Entfernung zum Hintergrund ist durch den Median aller Z-Koordinaten gegeben. Durch die Medianbildung werden Ausreißer eliminiert [Jain et al. 95].

4.1.4 Korrelation markanter Punkte

Zwischen einzelnen Punkten aus dem Bild $B(t_1)$ und dem Bild $B(t_2)$ muß nun eine Korrespondenz hergestellt werden. Die korrespondierenden Punktpaare werden von der in Abschnitt 4.1.2 beschriebenen Fehlerkorrektur benötigt und zur Berechnung der geschätzten Entfernung zum Hintergrund verwendet.

Die Korrespondenz kann auf verschiedene Arten hergestellt werden. Jain et al. [Jain et al. 95] beschreiben ein iteratives, auf Relaxation und Wahrscheinlichkeiten basierendes Verfahren, um Korrespondenzen zwischen einzelnen Punkten herzustellen. Neben diesem iterativen Verfahren beschreiben Jain et al. aber auch noch ein Verfahren, um die Korrespondenz interessanter Punkte zweier Stereo-Bilder herzustellen. Das Verfahren basiert auf einfacher Korrelation von Regionen um interessante Punkte. Dieses Verfahren wird hier angewandt.

Dazu wird zunächst eine Menge von interessanten Punkten für die Bilder $B(t_1)$ und $B(t_2)$ mit Hilfe des Moravec-Operators [Moravec 77] bestimmt. Der Moravec-Operator wird wie folgt berechnet [Jain et al. 95]:

- Berechnung der Grauwertvarianzen in den vier Richtungen horizontal, vertikal und diagonal einer Region R.

$$\begin{aligned} I_1 &= \sum_{(x,y)\in R} [B(x,y) - B(x,y+1)]^2 \\ I_2 &= \sum_{(x,y)\in R} [B(x,y) - B(x+1,y)]^2 \\ I_3 &= \sum_{(x,y)\in R} [B(x,y) - B(x+1,y+1)]^2 \\ I_4 &= \sum_{(x,y)\in R} [B(x+1,y) - B(x,y+1)]^2 \end{aligned}$$

- Bestimmung der kleinsten Varianz

$$I(x,y) = \min(I_1, I_2, I_3, I_4).$$

- Unterdrückung nicht-lokaler Maxima.

- Alle Punkte, die über einem bestimmten Schwellwert liegen, werden vom Moravec-Operator als interessant definiert.

Die interessanten Punkte von Bild $B(t_1)$ seien $\check{B}(t_1)$, die von Bild $B(t_2)$ seien $\check{B}(t_2)$. Zwischen den interessanten Punkten $\check{B}(t_1)$ und $\check{B}(t_2)$ wird nun eine Korrespondenz hergestellt. Aufgrund der Eigenbewegung der Kamera können sich die Regionen um die markanten Punkte verändert haben [Zheng et al. 95]. Zheng et al. [Zheng et al. 95] haben ein Verfahren entwickelt, markante Punkte einer Bildsequenz für beliebige Kamerabewegungen zu verfolgen. Dazu kompensieren sie zunächst die Eigenbewegung der Kamera aufgrund visueller Informationen und stellen dann die Korrespondenz her. Der Ansatz, zuerst die Eigenbewegung zu kompensieren, und dann die Korrelation durchzuführen, wird auch hier verfolgt.

Die Bildpunkte von Bild $B(t_1)$ werden gemäß der folgenden Gleichungen transformiert. Dabei gibt $\tilde{Z}$ die geschätzte Entfernung zum Hintergrund an. Das resultierende Bild wird mit

$B_T(t_1)$ bezeichnet. Die markanten Punkte von Bild $B_T(t_1)$ seien $\check{B}_T(t_1)$.

$$x_T(t_1) = f\frac{r_{11}x(t_1) + r_{12}y(t_1) + fr_{13} + f\frac{\varrho_x}{\tilde{Z}}}{r_{31}x(t_1) + r_{32}y(t_1) + fr_{33} + f\frac{\varrho_z}{\tilde{Z}}} \tag{4.9}$$

$$y_T(t_1) = f\frac{r_{21}x(t_1) + r_{22}y(t_1) + fr_{23} + f\frac{\varrho_y}{\tilde{Z}}}{r_{31}x(t_1) + r_{32}y(t_1) + fr_{33} + f\frac{\varrho_z}{\tilde{Z}}} \tag{4.10}$$

Für jeden interessanten Punkt $(x_1, y_1) \in \check{B}_T(t_1)$ wird eine Region R_{x_1,y_1} im Bild $B_T(t_1)$ definiert. Diese Region wird mit allen Regionen R_{x_2,y_2} im Bild $B(t_2)$ für alle $(x_2, y_2) \in \check{B}(t_2)$ korrelliert. Es wird also darauf verzichtet, einen Bereich um den interessanten Punkt abzusuchen. Statt dessen werden direkt die Regionen miteinander verglichen. Da markante Punkte für die Korrellation verwendet werden, ist es einfacher geworden, eine Korrespondenz zu finden. Die Regionen besitzen alle eine hohe Varianz und sind somit leichter zu unterscheiden.

Für die vorliegende Arbeit werden korrespondierende Punkte des Hintergrundes benötigt. Daher sollte keine Korrespondenz zwischen den markanten Punkten eines bewegten Objektes hergestellt werden. Die Bewegung des bewegten Objektes ist aber unabhängig von der Kamerabewegung. Daher ist es schwieriger eine Korrespondenz zwischen markanten Punkten eines bewegten Objektes statt zwischen markanten Punkten des Hintergrundes herzustellen. Dies ist genau das gewünschte Verhalten. Eine Korrespondenz markanter Punkte, die sich auf der Kontur des bewegten Objektes befindet, wird ebenfalls erschwert, falls sich das Objekt vor einem nicht-homogenen Hintergrund bewegt. Um zu erreichen, daß nur geeignete Korrespondenzen hergestellt werden, wird ein Schwellwert eingeführt, der alle Korrespondenzen ausschließt, die schlechter als der Schwellwert sind.

4.2 Detektion von Veränderungen

Nachdem die Eigenbewegung der Kamera kompensiert wurde, kann wie auf einer stationären Bildsequenz weitergearbeitet werden. Um Regionen zu finden, an denen eine Bewegung stattgefunden hat, wird zunächst jeder Bildpunkt von Bild $B(t_1)$ gemäß der folgenden Gleichungen transformiert. Die Parameter r_{ij} und ϱ_j $(i, j \in \{1, 2, 3\})$ sind Elemente der Tansformationsmatrix T, die die Eigenbewegung der Kamera beschreibt. Die Brennweite der Kamera sei f und die geschätzte Entfernung zum Hintergrund sei $\tilde{Z}$. Die Verschiebung, die von der Fehlerkorrektur ermittelt wurde, sei $(\Delta x, \Delta y)$.

$$x_{T_\Delta}(t_1) = f\frac{r_{11}x(t_1) + r_{12}y(t_1) + fr_{13} + f\frac{\varrho_x}{\tilde{Z}}}{r_{31}x(t_1) + r_{32}y(t_1) + fr_{33} + f\frac{\varrho_z}{\tilde{Z}}} + \Delta x \tag{4.11}$$

$$y_{T_\Delta}(t) = f\frac{r_{21}x(t_1) + r_{22}y(t_1) + fr_{23} + f\frac{\varrho_y}{\tilde{Z}}}{r_{31}x(t_1) + r_{32}y(t_1) + fr_{33} + f\frac{\varrho_z}{\tilde{Z}}} + \Delta y \tag{4.12}$$

Das so transformierte und korrigierte Bild wird mit $B_{T_\Delta}(t_1)$ bezeichnet. Jetzt wird das Differenzbild zwischen dem transformierten Bild $B_{T_\Delta}(t_1)$ und dem tatsächlichen Bild $B(t_2)$ gebildet. Es wird also die zeitliche Ableitung durch eine einfache Subtraktion approximiert.

$$\frac{\partial B(x, y, t)}{\partial t} \approx B(x, y, t + \delta t) - B_{T_\Delta}(x, y, t)$$

Danach wird das Absolutbild des Differenzbildes berechnet.

4.3 Unterdrückung von verbleibenden Ungenauigkeiten in der Prädiktion

Das Absolutbild wird durch die Anwendung morphologischer Operatoren [Murray et al. 94a] weiterbearbeitet, um verbleibende Störungen zu unterdrücken und Regionen an denen eine Veränderung stattgefunden hat, hervorzuheben. Für die morphologischen Operationen wird ein $w \times h$ großes Strukturelement M verwendet. Die Operationen sind wie folgt definiert:

- Erosion $(B \ominus M)(x,y) = \min\{B(x+x', y+y') | \lfloor -\frac{w}{2} \rfloor \leq x' \leq \lfloor \frac{w}{2} \rfloor, \lfloor -\frac{h}{2} \rfloor \leq y' \leq \lfloor \frac{h}{2} \rfloor\}$
- Dilation $(B \oplus M)(x,y) = \max\{B(x+x', y+y') | \lfloor -\frac{w}{2} \rfloor \leq x' \leq \lfloor \frac{w}{2} \rfloor, \lfloor -\frac{h}{2} \rfloor \leq y' \leq \lfloor \frac{h}{2} \rfloor\}$
- Opening $B \circ M = (B \ominus M) \oplus M$
- Closing $B \bullet M = (B \oplus M) \ominus M$

Die morphologischen Operationen werden auf den Grauwerten des Absolutbildes angewandt. Kleine Bereiche des Differenzbildes, die aufgrund einer verbleibenden Ungenauigkeit in der Prädiktion des aktuellen Bildes entstanden sind, werden unterdrückt. Es werden die folgenden morphologischen Operatoren angewandt.

- Schließung mit einem kleinen quadratischen Strukturelement, um Bereiche mit viel Bewegung miteinander zu verbinden.
- Öffnung mit einem etwas größeren Strukturelement, um Störungen zu unterdrücken.

Nachdem die morphologischen Operationen angewandt wurden, wird das Ergebnisbild binarisiert.

Jetzt werden die äußeren Konturen der Regionen, die nach der Anwendung morphologischer Operatoren übriggeblieben sind, bestimmt. Danach wird nur noch auf den Konturen bzw. Polygonen weitergearbeitet. Die Konturen werden als Punktmenge gespeichert. Dadurch ist eine schnelle Extraktion der bewegten Objekte möglich. Denn von nun an wird nur auf einer relativ kleinen Punktmenge weitergearbeitet und nicht mehr auf ganzen Bildern.

4.4 Zusammenfassen naher Regionen

Wegen der Verwendung von morphologischen Operatoren zur Unterdrückung von kleinen Störungen, kann es vorkommen, daß auch die Kontur eines sich bewegenden Objektes auseinandergerissen wird. Dünne Verbindungen im Differenzbild werden durch die morphologischen Operationen aus dem Differenzbild gelöscht. Einmal gelöscht lassen sich diese Regionen nicht mehr mit morphologischen Operationen verbinden, wenn eine feste Größe des Strukturelements angenommen wird.

In Abbildung 4.5 ist dies für ein Objekt gezeigt, das sich nicht in Richtung einer Kante bewegt. Die durch Kreise markierten Punkte sind nicht im Differenzbild enthalten, da sie zum Objekt im vorhergehenden und im aktuellen Bild gehören. Diese Punkte werden von Yalamanchili et al. [Yalamanchili et al. 82] wieder in das Differenzbild eingefügt. Da in der vorliegenden Arbeit jedoch noch morphologische Operationen auf das Differenzbild angewandt werden, fehlen nicht nur einzelne Punkte, sondern ganze Bereiche im Differenzbild. In Abbildung 4.5 ist dies für eine Open-Operation mit einem kreisförmigen Strukturelement gezeigt.

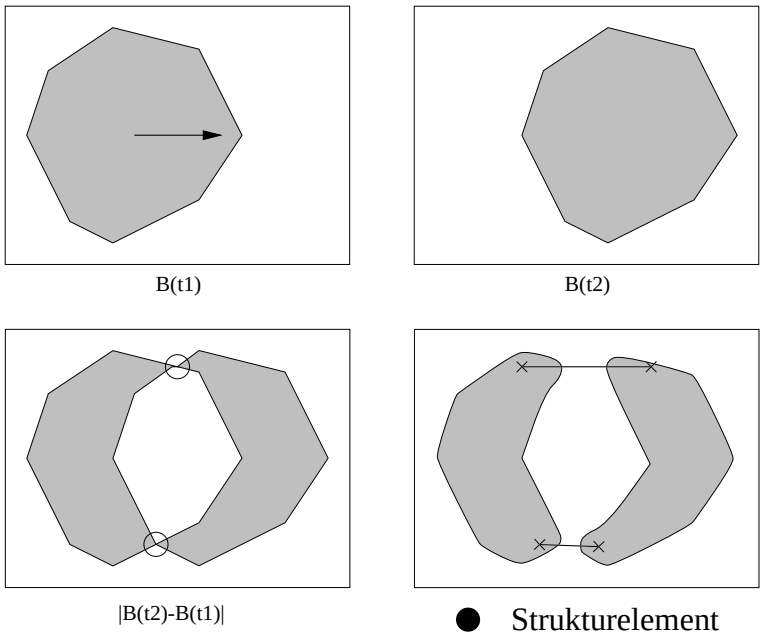

Abbildung 4.5: Zusammenfassen zweier Regionen des Differenzbildes, wenn diese durch morphologische Operationen bearbeitet wurden.

Das Strukturelement trennt die Bereiche des Differenzbildes voneinander. Zwar könnten die Bereiche durch eine Closing-Operation wieder vereint werden. Es stellt sich die Frage, wie groß das Strukturelement sein muß. Um dieses Problem zu lösen, wird eine Heuristik verwandt, mit der die getrennten Bereiche des Differenzbildes wieder zusammengefaßt werden. Für zwei Bereiche werden die beiden nächsten Punkte der beiden Konturen bestimmt. Zwischen beiden Punkten wird eine Strecke gezogen. Diese Strecke wird in beide Richtungen um die Länge der Strecke verlängert. Liegen die Endpunkte der verlängerten Strecke innerhalb der Bereiche, so werden diese Bereiche zusammengefaßt. Diese Heuristik setzt also implizit die Größe und die Entfernung der Bereiche ein, um zu bestimmten, ob die Bereiche zusammengefaßt werden oder nicht.

Beide Bereiche müssen nicht unbedingt zusammengehören. Es kann auch vorkommen, daß die Bereiche von der Heuristik zusammengefaßt werden, obwohl sie nicht zusammengehören. In der Praxis (siehe Kapitel 6) hat sich diese Heuristik als sinnvoll erwiesen. Es wurde auch mit einer schwächeren Bedingung experimentiert. Bereiche könnten auch dann zusammengefaßt werden, wenn nur ein Punkt der verlängerten Strecke innerhalb eines Bereiches liegt. Doch dann werden in der Regel kleine Bereiche zu einem großen Bereich hinzugefügt. Ein sehr großer Bereich würde alle kleineren Bereiche “verschlucken”.

In Abbildung 4.6 ist dies für einen großen Bereich dargestellt. Befindet sich der kleine Bereich nahe beim großen, so wird er durch die Erfüllung des beidseitigen Tests zusammengefaßt. Abbildung 4.6 zeigt die beiden Bereiche auch noch in relativ großer Entfernung voneinander. Bei einem einseitigen Test würden die Bereiche zusammengefaßt werden, bei einem zweiseitigen Test nicht. Dies wirft die Frage auf, warum der entfernte Bereich zum großen Bereich hinzugefügt werden sollte. Bei einem zweiseitigen Test kann argumentiert werden, daß die

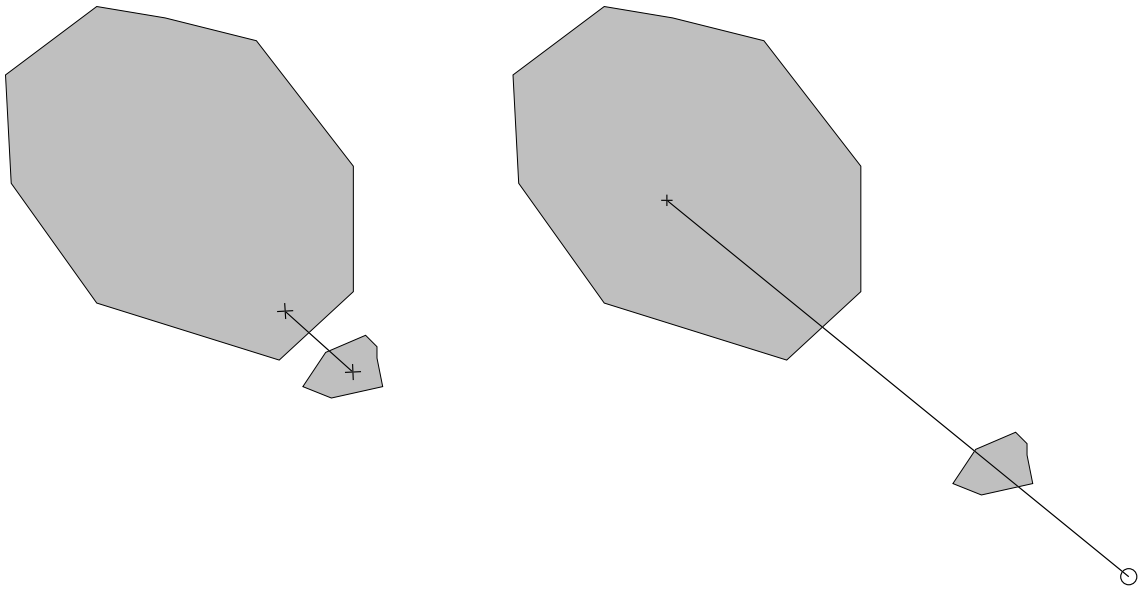

Abbildung 4.6: Die Heuristik wirft die Frage auf, ob ein kleiner Bereich zu einem großen Bereich hinzugefügt werden soll oder nicht.

Bereiche aufgrund ihrer Proportionen zusammengehören. Doch diese Argumentation gilt nicht für den einseitigen Test. Da keine zusätzlichen Informationen über die Bereiche vorhanden sind, werden die Bereiche nur dann zusammengefaßt, wenn sie beide Tests erfüllen.

Somit werden eher zu wenig Bereiche zusammengefaßt als zu viele. Die Verwendung eines einseitigen Tests hätte noch zur Folge, daß sehr schnell, sehr große Bereiche entstehen würden. Diese Bereiche sollten natürlich die Konturen eines bewegten Objektes möglichst genau beschreiben. Doch durch einen einseitigen Test wären auch kleine entfernte Störungen zu einem großen Bereich hinzugefügt worden. Dies hätte zur Folge, daß ein möglicherweise relativ großes bewegtes Objekt extrahiert werden würde, obwohl in Wirklichkeit ein kleines bewegtes Objekt vorhanden ist. Die Grenzbereiche der Heuristik sind in Abbildung 4.7 für zwei quadratische Bereiche dargestellt. Für eine gegebene Strecke lassen sich beliebig viele Objekte finden, bei denen die Heuristik die Objekte nicht zusammenfaßt, die aber durch Verwendung einer anderen Heuristik zusammengefaßt werden könnten.

4.5 Zusammenfassen von Bereichen zu Objekten

Einige Bereiche des Differenzbildes können durch die in Abschnitt 4.4 definierte Heuristik zusammengefaßt werden. Bewegt sich ein Objekt jedoch in Richtung einer Kante, so entstehen zwei getrennte Bereiche im Differenzbild. Diese Bereiche wurden von Yalamanchili et al. [Yalamanchili et al. 82] aufgrund geometrischer Überlegungen zusammengefaßt. Dazu werden colineare, gleichlange Kanten im Differenzbild miteinander verbunden. Das Verfahren wurde bereits in Abschnitt 3.6 beschrieben.

Für die vorliegende Arbeit werden morphologische Operationen zur Unterdrückung von Störungen eingesetzt. Daher entstehen auch getrennte Regionen, wenn die Bewegung nur

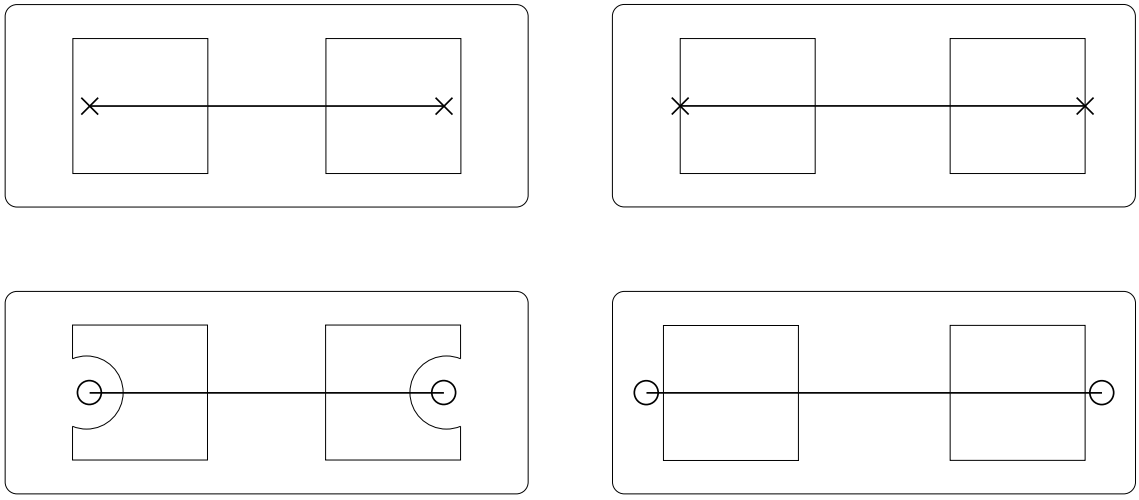

Abbildung 4.7: Grenzbereiche der Heuristik zum Zusammenfassen am Beispiel zweier quadratischer Bereiche

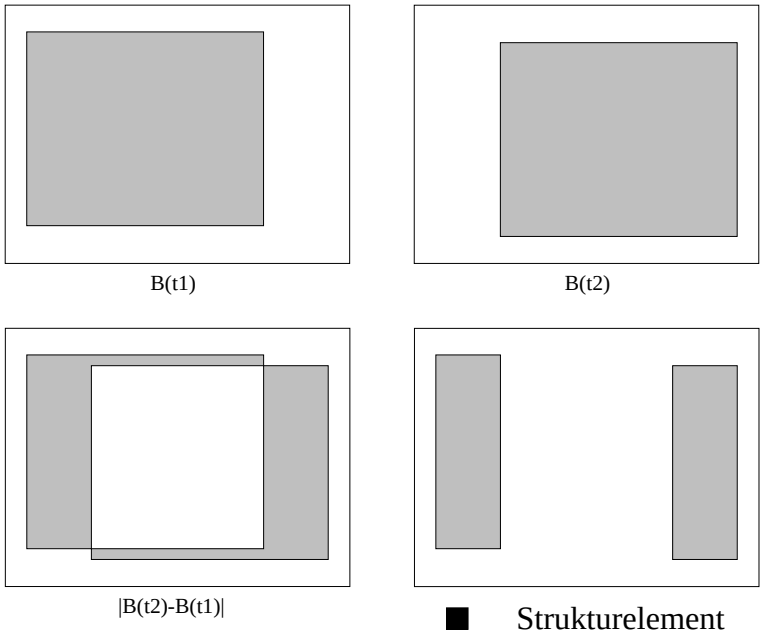

Abbildung 4.8: Zwei getrennte Regionen entstehen auch durch die Anwendung morphologischer Operationen.

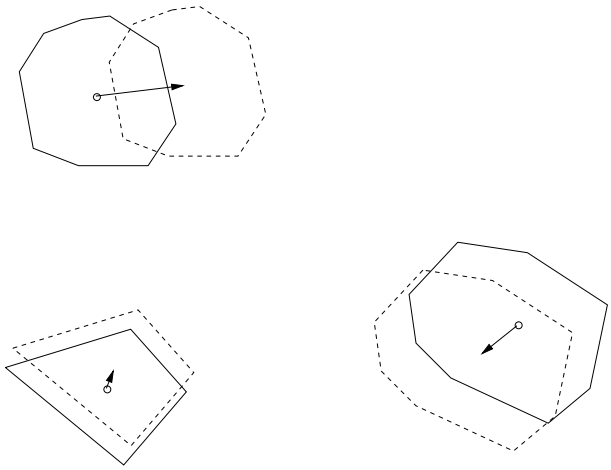

Abbildung 4.9: Prädiktion der Objektbewegung

ungefähr in Richtung einer Kante des Objektes erfolgt. In Abbildung 4.8 ist dies für ein rechteckiges Objekt dargestellt. Das Objekt bewegt sich in Richtung der horizontalen Kante und es bewegt sich etwas nach unten. Dabei entstehen zwei kleine Verbindungsstreifen. Durch die Anwendung einer Opening-Operation werden diese Verbindungsstreifen gelöscht. In Abbildung 4.8 ist dies für ein quadratisches Strukturelement dargestellt. Die beiden Regionen könnten mit dem Verfahren von Yalamanchili et al. [Yalamanchili et al. 82] zusammengefaßt werden.

Yalamanchili et al. [Yalamanchili et al. 82] setzen voraus, daß die Objekte, die sich im Bild der Kamera bewegen, Polygone sind. Für die vorliegende Arbeit wird dies jedoch nicht angenommen. Sind die zu extrahierenden Objekte keine Polygone, so kann der Kolinearitätstest nicht angewandt werden. Daher muß ein anderes Verfahren eingesetzt werden, um entfernte Bereiche des Differenzbildes zusammenzufassen, die zum selben Objekt gehören.

Ist das bewegte Objekt, zu dem die Bereiche gehören, aus früheren Bildern der Sequenz bereits erfolgreich extrahiert, kann diese Information eingesetzt werden, um die Bereiche des aktuellen Differenzbildes zusammenzufassen. Dazu wird die Position des Objektes im vorhergehenden Bild und der Geschwindigkeitsvektor des Objektes verwendet, um die Position im aktuellen Bild zu berechnen, an der es sich voraussichtlich befinden wird. Als Geschwindigkeitsvektor des Objektes wird in der vorliegenden Arbeit die Geschwindigkeit des bewegten Objektes in der Bildebene bezeichnet. Die dreidimensionale Bewegung des Objektes wird auf einen zweidimensionalen Bewegungsvektor reduziert. Der Geschwindigkeitsvektor in der Bildebene wird aus der Geschwindigkeit der Schwerpunkte der Regionen berechnet. In Abbildung 4.9 ist die Bewegung einiger Objekte entsprechend ihres Bewegungsvektors dargestellt.

Mit der vorhergesagten Position der bewegten Objekte können die Bereiche des Differenzbildes zusammengefaßt werden. Sugimoto et al. [Sugimoto et al. 86] verwenden die vorhergesagte Position der Objekte um bewegte Objekte zu extrahieren. Dabei darf ein bewegtes Objekt auch ein anderes bewegtes Objekt verdecken. Sugimoto et al. [Sugimoto et al. 86] vergrößern die vorhergesagte Fläche eines bewegten Objektes entsprechend ihres Bewegungsvektors. Je schneller ein Objekt sich bewegt, umso stärker wird dessen Fläche vergrößert. Verdeckt ein Objekt ein anderes, so werden die Bereiche zu beiden Objekten hinzugefügt.

Dieser Ansatz soll auch hier angewandt werden. Allerdings werden die vorhergesagten

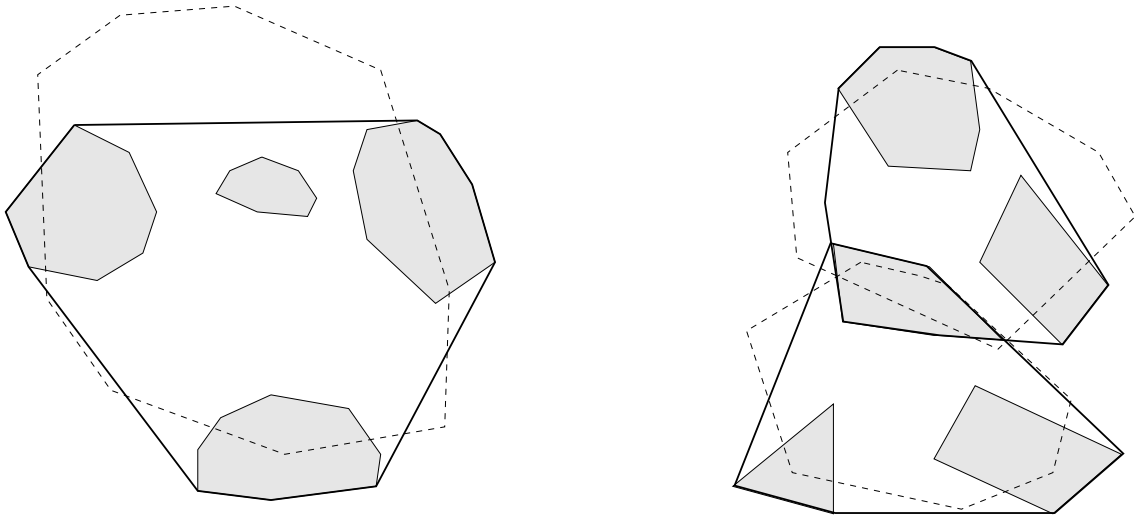

Abbildung 4.10: Erzeugung neuer bewegter Objekte mit Hilfe der vermuteten Position der Objekte. Die Bereiche aus dem Differenzbild sind mit dünner Linie gezeichnet. Die vermutete Positionen der Objekte sind mit gestrichelter Line gezeichnet. Die konvexen Hüllen der resultierenden Objekte sind mit dicker Line gezeichnet.

Flächen der bewegten Objekte nicht vergrößert. Da auf Konturen des Differenzbildes gearbeitet wird, werden auch Bereiche, die aus der Prädiktion herausragen, zum Objekt hinzugefügt. Für jedes vorhergesagte bewegte Objekt werden die Konturen des Differenzbildes zu einem Objekt zusammengefaßt, die die vorhergesagte Fläche des Objektes berühren. Für einige vorhergesagte Objekte und einige Bereiche des Differenzbildes ist dies in Abbildung 4.10 dargestellt. Die Bereiche des Differenzbildes werden ausgehend von der Prädiktion der Objektposition zusammengefaßt. Daher können einzelne Bereiche zu mehreren Objekten hinzugefügt werden.

4.6 Erzeugung eines bewegten Objektes mittels Hypothesen

Wurde bereits ein bewegtes Objekt aus der Bildsequenz extrahiert, so kann diese Information eingesetzt werden, um die Bereiche des Differenzbildes zu einem Objekt zusammenzufügen. Doch es stellt sich die Frage, wie denn das bewegte Objekt extrahiert wird, wenn diese Information noch nicht vorliegt. Es muß also nach einer Möglichkeit gesucht werden, diesen ersten Schritt zu machen.

Die Bereiche des Differenzbildes wurden bereits durch eine Heuristik zusammengefaßt. In der Regel sollten die Bereiche schon fast den bewegten Objekten entsprechen. Trifft dies zu, so können diese Bereiche dazu eingesetzt werden, eine erste Bewegungshypothese zu generieren. Zunächst muß jedoch eine Korrespondenz zwischen den einzelnen Bereichen zweier Differenzbilder hergestellt werden. Die Korrespondenz kann aufgrund eines Kompatibilitätstests bestimmt werden. Für den Test können zum Beispiel die folgenden Kriterien herangezogen werden.

- Größe der Bereiche

- Form der Bereiche
- Farbe bzw. Grauwerte der Objekte
- Position der Bereiche
- Bahn der Bereiche

Für die vorliegende Arbeit stellte sich die Frage, welches Kriterium am geeignetsten für eine Implementierung ist. Um die Frage zu klären, ob alle Kriterien gleich gut geeignet sind, werden die einzelnen Kriterien nun näher betrachtet.

Um die Bahn der Bereiche zu bestimmen, müssen mehrere Differenzbilder verwendet werden. Da ein bewegtes Objekt jedoch so schnell wie möglich extrahiert werden sollte, sollte die Anzahl der Bilder, die für den Kompatibilitätstest herangezogen werden, möglichst klein sein. Die Erzeugung der Bahn ist jedoch relativ aufwendig. Zudem muß für die Generierung einer Bahn für die Bereiche zuerst das Korrespondenzproblem gelöst werden. Doch genau dieses Problem sollte durch die Verwendung der Bahn der Bereiche gelöst werden. Dieser Kreis muß durchbrochen werden. Die Position der Bereiche kann ebenfalls zur Lösung des Korrepondenzproblems herangezogen werden. Geht man davon aus, daß sich die Objekte nicht zu nahe kommen und sich nicht zu schnell bewegen, so könnte aufgrund der Entfernung der Bereiche eine Korrespondenz hergestellt werden. Ein Bereich korrespondiert dabei mit dem nächstbesten Bereich.

Haben die zu extrahierenden Objekte unterschiedliche Farben, so kann diese Information eingesetzt werden, um eine Korrespondenz zwischen den Bereichen des Differenzbildes herzustellen. Da die Bereiche des Differenzbildes aber noch nicht direkt bewegten Objekten entsprechen, ist auch hier noch eine Zuordnung der Farbe zu den Bereichen notwendig. Da in der vorliegenden Arbeit nur auf Grauwertbildern gearbeitet wird, kann die Farbe nicht zur Lösung des Korrespondenzproblems eingesetzt werden. Verändert sich die Form der Bereiche kaum, so kann zwischen den Bereichen eine Korrespondenz hergestellt werden, die aufgrund ihrer Form am ähnlichsten sind. Von Meyer et. al [Meyer et al. 94] [Meyer et al. 92] werden die summierten quadratischen Entfernungen der Punkte zweier Polygone als Maß für die Ähnlichkeit der Polygone herangezogen.

Die Größe der Bereiche ist ein relativ einfaches Maß, um die Korrespondenz herzustellen. Es wird angenommen, daß sich die Größe der Bereiche nicht zu stark unterscheiden. Dies ist in der Regel der Fall, wenn sich die tatsächlichen bewegten Objekte nicht zu schnell auf die Kamera zu oder von ihr weg bewegen und die Bereiche schon recht genau die Form der tatsächlich bewegten Objekte wiedergeben.

Ist die Korrespondenz hergestellt, so kann für den Schwerpunkt der Bereiche ein Bewegungsvektor berechnet werden. Jede Korrespondenz führt also auf ein vermutlich bewegtes Objekt, im folgenden kurz Hypothese genannt. Die Hypothese kann vielleicht später validiert werden. Ob es validiert werden kann, hängt von der Qualität der Hypothese ab. Je besser der Mechanismus ist, der die Korrespondenz herstellt, desto wahrscheinlicher ist es, daß die Hypothese auch tatsächlich validiert wird.

Möchte man sicher sein, daß auch wirklich ein bewegtes Objekt extrahiert wird, so kann die Hypothese zum Beispiel erst dann validiert werden, wenn die Hypothese mit der Realität schon einige Zeit übereingestimmt hat. Als Maß der Übereinstimmung könnte geprüft werden, wie genau die Bereiche des Differenzbildes zur vermuteten Position des Objektes passen. Liegt kein entsprechender Bereich im Differenzbild vor, so kann die Hypothese nicht validiert werden.

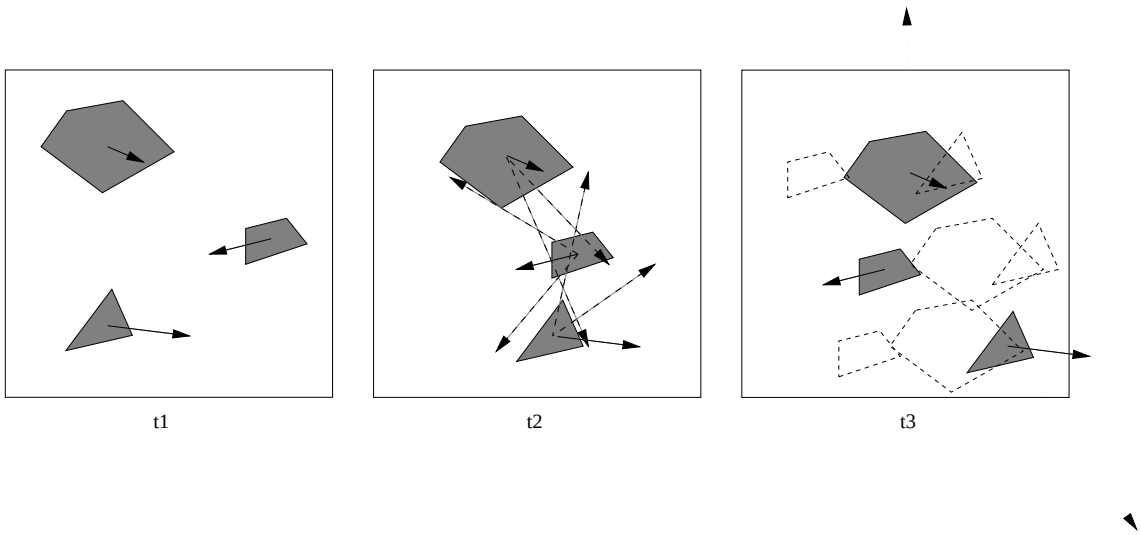

Abbildung 4.11: Drei Bilder mit extrahierten Objekten. Die ersten beiden Bilder können zur Generierung der Bewegungsvektoren eingesetzt werden. Die tatsächlichen Bewegungsvektoren sind mit dicken Pfeilen eingezeichnet. Die gestrichelten Bewegungsvektoren ergeben sich, falls zwischen allen Bereichen der beiden ersten Bildern eine Korrespondenz hergestellt wird. Im dritten Bild können die tatächlichen Bewegungsvektoren bestimmt werden. Bei einem wenig restriktiven Vergleich der Prädiktion mit den Bereichen des dritten Bildes könnten noch die gepunktet eingezeichneten Bewegungsvektoren generiert werden. Doch bereits im folgenden Bild liegen die Prädiktion der Bereiche außerhalb des Bildes.

In Abbildung 4.11 sind einige bewegte Objekte dargestellt. Ihr Bewegungsvektor ist als dick gezeichneter Pfeil dargestellt. Neben dem tatsächlichen Bewegungsvektor sind aber auch noch einige Bewegungsvektoren gestrichelt gezeichnet. Diese Bewegungsvektoren entstehen, falls eine Korrespondenz eines Objektes des ersten Bildes mit allen anderen Objekten des zweiten Bildes angenommen wird. Zur Generierung der Hypothesen wurde das erste und das zweite Bild herangezogen. Im dritten Bild könnten die Hypothesen validiert werden. Die Konturen der Hypothesen sind im dritten Bild gestrichelt eingezeichnet. Die Konturen der Hypothesen, die mit der tatsächlichen Bewegung der Objekte zusammenfällt, sind nicht extra eingezeichnet. Zieht man die Größe und Form der Bereiche zur Validierung heran, so werden die korrekten Bereiche im dritten Bild eindeutig wiedergefunden.

In Abbildung 4.11 sind im dritten Bild zwei Hypothesen zu sehen, deren Konturen mit den tatsächlichen Objekten zusammenfallen. Bei einem wenig restriktiven Vergleich der Form und Größe könnte eine Korrespondenz zwischen den angrenzenden Bereichen des dritten Bildes und den Hypothesen hergestellt werden. Dies hätte zur Folge, daß die Bewegungsvektoren der Hypothesen wie durch die gepunkteten Pfeile angegeben, gesetzt werden würden. Der Bewegungsvektor wurde dabei erneut aus dem Schwerpunkt der korrespondierenden Objekte des zweiten und dritten Bildes berechnet. Die Hypothesen könnten also bereits im folgenden Bild nicht mehr validiert werden.

Aufgrund dieser Überlegungen wird in der vorliegenden Arbeit für jeden Bereich des Differenzbildes mit jedem Bereich des Differenzbildes eine Korrespondenz hergestellt, wenn die Flächen der Bereiche nicht zu stark voneinander abweichen. Dabei entstehen, wie an obigem Beispiel verdeutlicht wurde, viele falsche Hypothesen. Doch die falschen Hypothesen werden relativ schnell in folgenden Differenzbildern eliminiert. Nur wenn eine Hypothese validiert werden kann, extrahiert der Algorithmus ein bewegtes Objekt. Eine Hypothese wird in der vorliegenden Arbeit validiert, wenn die Prädiktion eines bewegten Objektes mit einer Veränderung im Differenzbild zusammenfällt.

Kapitel 5

Das Programm EMO - Extraction of Moving Objects

5.1 Eingesetzte Hardware und Software

Für das COMROS-Projekt [Levi et al. 95] stehen im Robotik-Labor der Universität Stuttgart drei Fahrzeuge zur Verfügung, die unterschiedlich ausgestattet sind.

- Athos ist mit einer Stereo-Schwarz-Weiß-Kamera auf einer Schwenk/Nick/Schiel-Einheit ausgestattet.
- Porthos ist mit einer Farb-Kamera auf einer Schwenk/Nick-Einheit ausgestattet.
- Aramis ist mit einem Greifarm ausgestattet.

Für die vorliegende Arbeit wurde das Fahrzeug Athos eingesetzt. Da nur eine monokulare Bildsequenz verarbeitet wird, wird lediglich eine der beiden Kameras eingesetzt.

Die Kamera liefert Schwarz-Weiß-Bilder. Die Bilder werden per Funkethernet an eine Sun-Workstation übertragen und dort digitalisiert. Die Verarbeitung der Bilder erfolgt ebenfalls auf der Sun-Workstation. Um die Bilder zu digitalisieren wurden die von Vogt und Mamier [Levi et al. 95] entwickelten SUN-Framegrabber-Routinen verwendet. Diese Routinen sind der Projektverwaltung [Vogt et al. 96] des COMROS-Projektes unterstellt und tragen das Kürzel `SFG`. Mit den SUN Framegrabber Routinen können Bilder mit einer Auflösung von 768×576, 384×288, 256×192, 192×144 und 153×115 digitalisiert werden.

Motor	1	2	3	4
Schwenkbereich	$\pm 170°$	$\pm 100°$	$\pm 45°$	$\pm 45°$
Beschleunigung	$1500 \frac{°}{s^2}$	$1500 \frac{°}{s^2}$	$6124 \frac{°}{s^2}$	$6124 \frac{°}{s^2}$
Geschwindigkeit	$300 \frac{°}{s}$	$300 \frac{°}{s}$	$350 \frac{°}{s}$	$350 \frac{°}{s}$
Auflösung	$\frac{0.9 \text{Bogenminuten}}{\text{Encoderpuls}}$	$\frac{0.9 \text{Bogenminuten}}{\text{Encoderpuls}}$	$\frac{0.9 \text{Bogenminuten}}{\text{Encoderpuls}}$	$\frac{0.9 \text{Bogenminuten}}{\text{Encoderpuls}}$
Optischer Encoder	$\frac{500 \text{Pulse}}{\text{Umdrehung}}$	$\frac{500 \text{Pulse}}{\text{Umdrehung}}$	$\frac{500 \text{Pulse}}{\text{Umdrehung}}$	$\frac{500 \text{Pulse}}{\text{Umdrehung}}$

Tabelle 5.1: Spezifikation der Gelenkmotoren der Stereo-Kamera (nach [Robosoft 94b])

Die Motoren des Fahrzeuges und die Motoren der Manipulatoren werden durch einen 16MHz 68020 Computer mit VME-Bus gesteuert. Dieser Computer befindet sich direkt auf dem Fahrzeug. Die Steuerung des Fahrzeugs und der Kamera wird durch das Echtzeitbetriebssystem ALBATROS [Robosoft 94a] vorgenommen. Die Kommandos, mit denen das Fahrzeug bewegt wird, werden über Funk auf das Fahrzeug übertragen. Das Betriebssystem ALBATROS hält eine Liste mit den vom Anwender gewünschten Bewegungsabläufen und arbeitet diese der Reihe nach ab. Der Anwender kann aber auch Befehle direkt abarbeiten lassen. Sie werden dann sofort ausgeführt und nicht in die Liste eingetragen. Die Verbindung zum Fahrzeug wird durch das von Rausch [Levi et al. 95] entwickelte Robot-Interface hergestellt. Das Robot-Interface unterliegt ebenfalls der Projektverwaltung des COMROS-Projektes und trägt das Kürzel `RoI`.

Um die digitalisierten Bilder zu verarbeiten, wird das von Pope et al. [Pope et al. 94a] entwickelte Bildverarbeitungsprogramm Vista eingesetzt. Vista ist eigentlich nicht für die Verarbeitung von Bildern in Echtzeit entwickelt worden. Es wurde dennoch für die vorliegende Arbeit eingesetzt, da Vista dem Anwender direkten Zugriff auf alle Datenstrukturen gibt. So können eigenständig ergänzende Operatoren für die Bildverarbeitung implementiert werden. Zudem besitzt Vista eine sehr komfortable Bibliothek zur Verwaltung von Polygonen. Vista stellt eine Reihe von grundlegenden Operatoren, wie zum Beispiel Kantenextraktionsoperator oder Faltungsoperatoren bereit. Im Rahmen dieser Arbeit mußten jedoch noch eine Reihe von speziellen Operatoren entwickelt werden. Die Programmierung mit Vista ist in [Pope et al. 94b] beschrieben. Bei der Entwicklung neuer Operatoren wurden die Vista-Konventionen beachtet [Pope 94].

5.2 Modellierung der Kamera

Die Stereo-Schwarz-Weiß-Kamera mit ihren Gelenken des Fahrzeuges Athos läßt sich nach dem ISO-Standard [Bräunl 95], wie in Abbildung 5.1 darstellen. In Tabelle 5.1 sind die von Robosoft [Robosoft 94b] angegebenen Daten für die Kamera zusammengefaßt. Die Kinematik der Kamera wird durch die Denavit-Hartenberg-Notation [Craig 89] beschrieben. An jedem Gelenk der Kamera wird ein Koordinatensystem angebracht. Die Gelenke werden von Gelenk 1 bis Gelenk 4 durchnumeriert. Dies entspricht der von Robosoft [Robosoft 94b] angegebenen Numerierung.

An der Kamera werden eine Reihe von Koordinatensystemen angebracht. Da zur Extraktion der bewegten Objekte nur eine Kamera eingesetzt wird, erfolgt die Beschreibung der Koordinatensysteme für nur eine Kamera. In Abbildung 5.2 ist die Positionierung der Koordinatensysteme an der Kamera visualisiert. Für die Kamera werden 5 Koordinatensysteme definiert. In Tabelle 5.2 sind die Koordinatensysteme der Kamera, des Fahrzeuges und das Weltkoordinatensystem zusammengefaßt.

Die Denavit-Hartenberg-Notation beschreibt einen Manipulator durch die Angabe von 4 Parametern für jede Verbindung [Craig 89].

- a_i Kürzeste Entfernung zwischen $\hat{\mathbf{Z}}_i$ und $\hat{\mathbf{Z}}_{i+1}$ gemessen in Richtung von $\hat{\mathbf{X}}_i$
- α_i Winkel zwischen $\hat{\mathbf{Z}}_i$ und $\hat{\mathbf{Z}}_{i+1}$ gemessen um die Drehachse $\hat{\mathbf{X}}_i$
- d_i Entfernung von $\hat{\mathbf{X}}_i$ und $\hat{\mathbf{X}}_{i+1}$ gemessen in Richtung von $\hat{\mathbf{Z}}_i$
- θ_i Winkel zwischen $\hat{\mathbf{X}}_{i-1}$ und $\hat{\mathbf{X}}_i$ gemessen um die Drehachse $\hat{\mathbf{Z}}_i$

Die Denavit-Hartenberg Parameter der Kamera sind in Tabelle 5.3 zusammengefaßt. Die

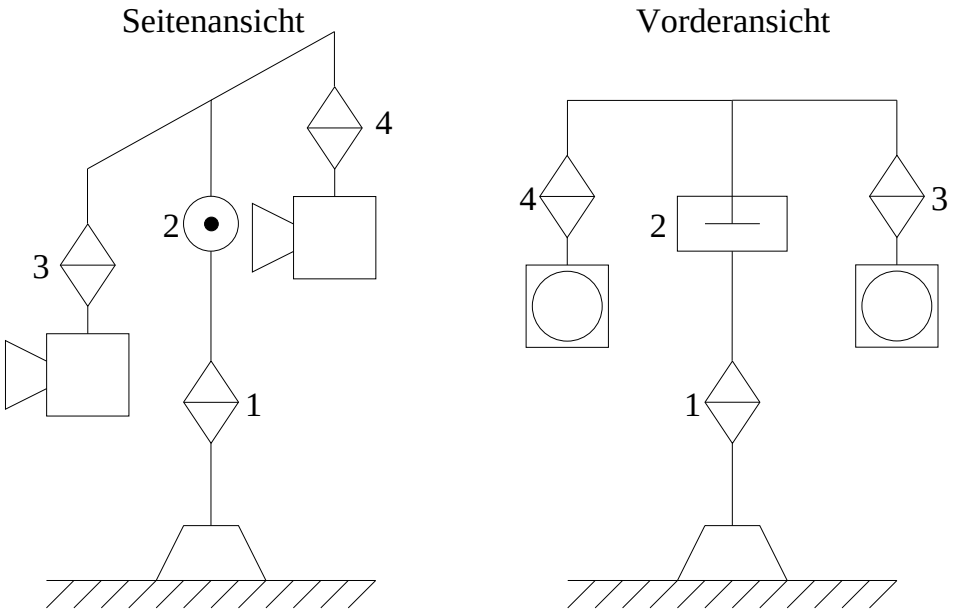

Abbildung 5.1: Darstellung der Stereo-Kamera nach ISO

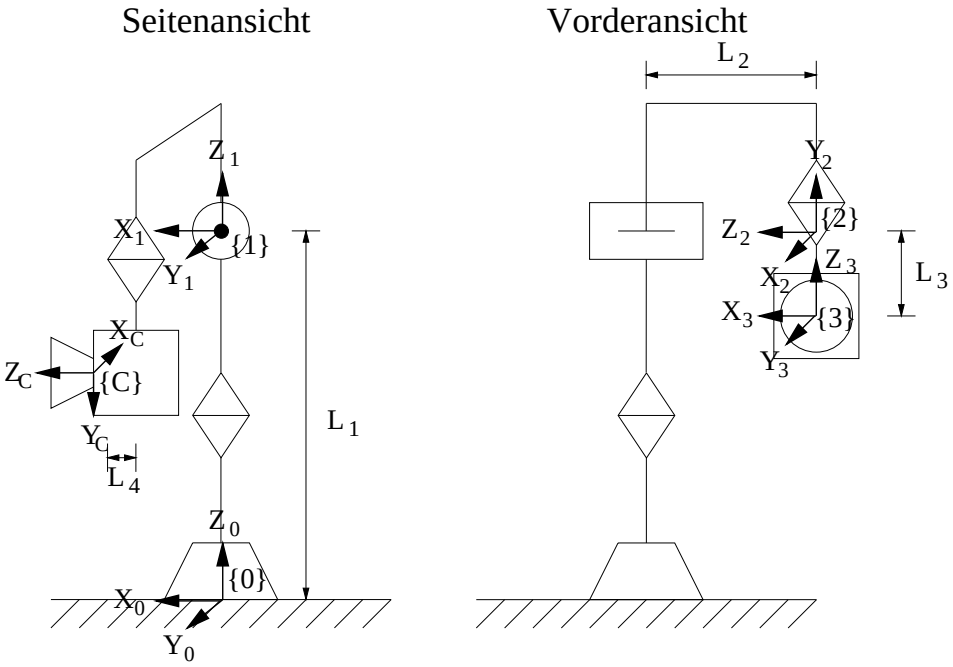

Abbildung 5.2: Position der Koordinatensysteme der Kamera

$\{R\}$	Koordinatensystem des Fahrzeuges
$\{0\}$	Koordinatensystem der Kamerabasis
$\{1\}$	Koordinatensystem der Kameraachse 1
$\{2\}$	Koordinatensystem der Kameraachse 2
$\{3\}$	Koordinatensystem der Kameraachse 3
$\{4\} = \{C\}$	Koordinatensystem der Kamera
$\{W\}$	Koordinatensystem der Welt

Tabelle 5.2: Liste der Koordinatensysteme

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	$0°$	0	L_1	θ_1
2	$90°$	0	$-L_2$	θ_2
3	$-90°$	0	$-L_3$	$\theta_3 - 90°$
4	$-90°$	0	L_4	0

Tabelle 5.3: Denavit-Hartenberg Parameter der Kamera

tatsächlichen Abmessungen der Parameter der Kamera sind in Tabelle 5.4 angegeben[1].

5.3 Modellierung des Roboters

Das Koordinatensystem $\{R\}$ des Fahrzeuges befindet sich, wie in Abbildung 5.3 dargestellt, zwischen den beiden Hinterrädern. Zur Zeichnung des Fahrzeugs wurde die Norm von Bräunl [Bräunl 96] verwendet. Die Z-Achse zeigt nach oben, die Y-Achse nach links und die X-Achse in Fahrtrichtung. Dies entspricht der von Robosoft in [Robosoft 94a] angegebenen Position und Orientierung. Die Bewegung des Fahrzeuges wird bezogen auf dieses Koordinatensystem angegeben. Das Fahrzeug kann über die Angabe der linearen Geschwindigkeit v und die Drehgeschwindigkeit w gesteuert werden. Die Kamerabasis befindet sich im Koordinatensystem des Fahrzeugs an der Position $[b_x, b_y, b_z]$. Die Abmessungen sind in Tabelle 5.5 aufgeführt.

[1]Die Blende der Kamera befindet sich direkt unterhalb der Schiel-Achse. Die Kamera kann aber in Z-Richtung verschoben werden. In diesem Fall muß ein Wert ungleich Null für L_4 verwendet werden.

L_1	L_2	L_3	L_4
$37,5cm$	$\pm 17,5cm$	$3,0cm$	$0cm$

Tabelle 5.4: Konstante Parameter der Kamera.

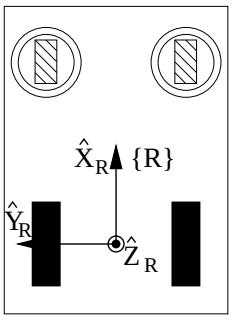

Abbildung 5.3: Koordinatensystem des Fahrzeuges

b_x	b_y	b_z
$37,5cm$	$0cm$	$53,5cm$

Tabelle 5.5: Position der Kamerabasis im Koordinatensystem des Fahrzeuges.

5.4 Bewegung der Kamera

Die Bewegung der Kamera kann durch eine allgemeine Transformationsmatrix beschrieben werden. ${}^{C(t_2)}_{C(t_1)}T$ sei diese Transformationsmatrix, die das Koordinatensystem der Kamera $\{C\}$ zum Zeitpunkt t_1 auf das Koordinatensystem der Kamera zum Zeitpunkt t_2 transformiert. Diese Transformationsmatrix soll nun berechnet werden. Dazu wird angenommen, daß zum Zeitpunkt t_1 die Position der Kamera relativ zum Fahrzeug durch die drei Variablen θ_1, θ_2 und θ_3 gegeben sind. Zum Zeitpunkt t_2 befinden sich die Gelenke auf den Positionen θ_1', θ_2' und θ_3'. Das Fahrzeug bewege sich mit der linearen Geschwindigkeit v und der Winkelgeschwindigkeit w.

Nun kann die Transformationsmatrix ${}^{C(t_2)}_{C(t_1)}T$ berechnet werden. Sie setzt sich aus drei Transformationsmatrizen zusammen. Die erste beschreibt die Transformation des Koordinatensystems der Kamera zum Zeitpunkt t_1 zum Koordinatensystem des Fahrzeuges. Die zweite beschreibt die Transformation des Koordinatensystem des Fahrzeuges während der Zeit $\delta t = t_2 - t_1$. Die dritte schließlich beschreibt die Transformation vom Koordinatensystem des Fahrzeuges zum Koordinatensystem der Kamera zum Zeitpunkt t_2. Also setzt sich ${}^{C(t_2)}_{C(t_1)}T$ wie folgt zusammen.

$$\begin{aligned} {}^{C(t_2)}_{C(t_1)}T &= {}^{C(t_2)}_{R(t_2)}T \cdot {}^{R(t_2)}_{R(t_1)}T \cdot {}^{R(t_1)}_{C(t_1)}T \\ &= {}^{R(t_2)}_{C(t_2)}T^{-1} \cdot {}^{R(t_2)}_{R(t_1)}T \cdot {}^{R(t_1)}_{C(t_1)}T \end{aligned}$$

Zunächst soll die Transformationsmatrix ${}^{R}_{C}T$ berechnet werden, die die Transformation des Koordinatensystems des Fahrzeuges auf das Koordinatensystem der Kamera beschreibt. In der folgenden Herleitung wird bei der Bezeichnung der Matrizen die Notation von Craig [Craig 89]

eingehalten. Durch $R_X(\theta)$, $R_Y(\theta)$ und $R_Z(\theta)$ werden Rotationsmatrizen beschrieben. Sie nehmen eine Drehung des Koordinatensystems um den Winkel θ um die X-, Y- bzw. Z-Achse vor. Mit $D_X(x)$, $D_Y(y)$ und $D_Z(z)$ wird eine Translation des Koordinatensystems in Richtung der X-, Y- bzw. Z-Achse um x, y bzw. z vorgenommen. Eine allgemeine Transformation von einem Koordinatensystem $\{i-1\}$ auf das Koordinatensystem $\{i\}$ ist nach [Craig 89] durch

$$
\begin{aligned}
{}^{i-1}_{i}T &= R_X(\alpha_{i-1})D_X(a_{i-1})R_Z(\theta_i)D_Z(d_i) \\
&= \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

gegeben.

Somit ergeben sich die einzelnen Transformationsmatrizen der Kamera wie folgt. Für $\sin\theta_i$ wird die übliche Abkürzung s_i verwendet und für $\cos\theta_i$ wird c_i geschrieben.

$$
{}^{0}_{1}T = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
{}^{1}_{2}T = \begin{pmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & L_2 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
{}^{2}_{3}T = \begin{pmatrix} s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & -L_3 \\ c_3 & -s_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
{}^{3}_{C}T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & L_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

Da das Koordinatensystem des Roboters die gleiche Orientierung wie das Koordinatensystem der Kamerabasis hat, ist die Transformationsmatrix ${}^{R}_{0}T$ durch eine einfache Translation gegeben:

$$
{}^{R}_{0}T = \begin{pmatrix} 1 & 0 & 0 & b_x \\ 0 & 1 & 0 & b_y \\ 0 & 0 & 1 & b_z \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

Nun soll die Transformationsmatrix ${}^{R}_{C}T$ berechnet werden, die das Koordinatensystem $\{R\}$ auf das Koordinatensystem $\{C\}$ transformiert.

$$
\begin{aligned}
{}^{R}_{C}T &= {}^{R}_{0}T \cdot {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T \cdot {}^{3}_{C}T \\
&= {}^{R}_{0}T \cdot {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot \begin{pmatrix} s_3 & 0 & c_3 & L_4 c_3 \\ 0 & -1 & 0 & -L_3 \\ c_3 & 0 & -s_3 & -L_4 s_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= {}^{R}_{0}T \cdot {}^{0}_{1}T \cdot \begin{pmatrix} c_2 s_3 & s_2 & c_2 c_3 & L_4 c_2 c_3 + L_3 s_2 \\ -c_3 & 0 & s_3 & L_4 s_3 + L_2 \\ s_2 s_3 & -c_2 & s_2 c_3 & L_4 s_2 c_3 - L_3 c_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= {}^{R}_{0}T \cdot \begin{pmatrix} c_1 c_2 s_3 + s_1 c_3 & c_1 s_2 & c_1 c_2 c_3 - s_1 s_3 & L_4(c_1 c_2 c_3 - s_1 s_3) + L_3 c_1 s_2 - L_2 s_1 \\ s_1 c_2 s_3 - c_1 c_3 & s_1 s_2 & s_1 c_2 c_3 + c_1 s_3 & L_4(s_1 c_2 c_3 + c_1 s_3) + L_3 s_1 s_2 + L_2 c_1 \\ s_2 s_3 & -c_2 & s_2 c_3 & L_4 s_2 c_3 - L_3 c_2 + L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} c_1 c_2 s_3 + s_1 c_3 & c_1 s_2 & c_1 c_2 c_3 - s_1 s_3 & L_4(c_1 c_2 c_3 - s_1 s_3) + L_3 c_1 s_2 - L_2 s_1 + b_x \\ s_1 c_2 s_3 - c_1 c_3 & s_1 s_2 & s_1 c_2 c_3 + c_1 s_3 & L_4(s_1 c_2 c_3 + c_1 s_3) + L_3 s_1 s_2 + L_2 c_1 + b_y \\ s_2 s_3 & -c_2 & s_2 c_3 & L_4 s_2 c_3 - L_3 c_2 + L_1 + b_z \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

Um die Transformation des Koordinatensystems der Kamera auf das Koordinatensystem des Fahrzeuges zu berechnen, wird noch die inverse Matrix zu ${}^{R}_{C}T$ benötigt.

$$
\begin{aligned}
{}^{C}_{R}T &= {}^{R}_{C}T^{-1} \\
&= \begin{pmatrix} c_1 c_2 s_3 + s_1 c_3 & s_1 c_2 s_3 - c_1 c_3 & s_2 s_3 & -L_1 s_2 s_3 + L_2 c_3 - b_x(c_1 c_2 s_3 + s_1 c_3) \\ & & & -b_y(s_1 c_2 s_3 - c_1 c_3) - b_z(s_2 s_3) \\ c_1 s_2 & s_1 s_2 & -c_2 & L_1 c_2 - L_3 - b_x c_1 s_2 - b_y s_1 s_2 + b_z c_2 \\ c_1 c_2 c_3 - s_1 s_3 & s_1 c_2 c_3 + c_1 s_3 & s_2 c_3 & -L_1 s_2 c_3 - L_2 s_3 - L_4 - b_x(c_1 c_2 c_3 - s_1 s_3) \\ & & & -b_y(s_1 c_2 c_3 + c_1 s_3) - b_z(s_2 c_3) \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

Nun soll die Transformationsmatrix ${}^{R(t_2)}_{R(t_1)}T$ berechnet werden. Die Bewegung des Fahrzeuges kann wie folgt annähernd beschrieben werden (Abbildung 5.4). Das Fahrzeug bewegt sich in der Zeit δt um die Strecke $d_x = v\delta t$ geradeaus und dreht sich um den Winkel $\beta = w\delta t$. Insgesamt ergibt sich also folgende Transformationsmatrix für die Bewegung des Fahrzeuges:

$$
\begin{aligned}
{}^{R(t_1)}_{R(t_2)}T &= D_X(v(t_2 - t_1)) \cdot R_Z(w(t_2 - t_1)) \\
&= D_X(d_x) \cdot R_Z(\beta)
\end{aligned}
$$

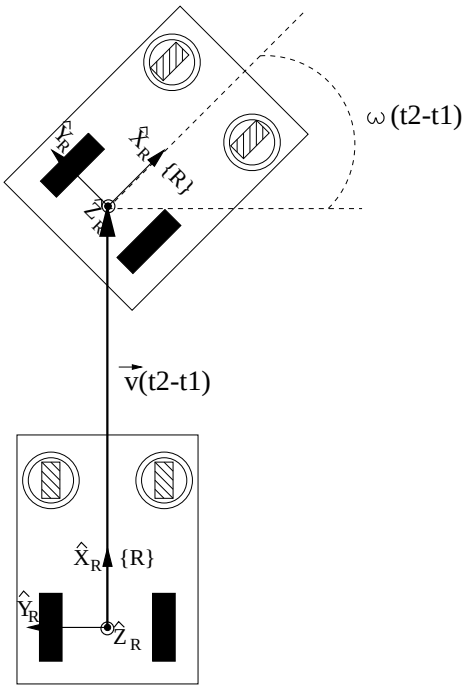

Abbildung 5.4: Bewegung des Fahrzeuges

Folglich ergibt sich für ${}^{R(t_2)}_{R(t_1)}T$:

$$
\begin{aligned}
{}^{R(t_2)}_{R(t_1)}T &= {}^{R(t_1)}_{R(t_2)}T^{-1} \\
&= (D_X(d_x) \cdot R_Z(\beta))^{-1} \\
&= R_Z^{-1}(\beta) \cdot D_X^{-1}(d_x) \\
&= R_Z(-\beta) \cdot D_X(-d_x) \\
&= \begin{pmatrix} c_\beta & s_\beta & 0 & 0 \\ -s_\beta & c_\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -d_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} c_\beta & s_\beta & 0 & d_x \\ -s_\beta & c_\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

Jetzt kann die Transformationsmatrix ${}^{C(t_2)}_{C(t_1)}T$ berechnet werden.

$$\begin{matrix}C(t_2)\\C(t_1)\end{matrix}T = \begin{matrix}R(t_2)\\C(t_2)\end{matrix}T^{-1} \cdot \begin{matrix}R(t_2)\\R(t_1)\end{matrix}T \cdot \begin{matrix}R(t_1)\\C(t_1)\end{matrix}T$$

Die Elemente der Transformationsmatrix $\begin{matrix}C(t_2)\\C(t_1)\end{matrix}T$ werden wie folgt bezeichnet:

$$\begin{matrix}C(t_2)\\C(t_1)\end{matrix}T = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \varrho_x \\ r_{21} & r_{22} & r_{23} & \varrho_y \\ r_{31} & r_{32} & r_{33} & \varrho_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

mit (wobei s_i' für $\sin\theta_i'$ und c_i' für $\cos\theta_i'$ geschrieben wird $i \in \{1,2,3\}$)

$$\begin{aligned}
r_{11} &= (c_1'c_2's_3' + s_1'c_3')(c_\beta(c_1c_2s_3 + s_1c_3) - s_\beta(s_1c_2s_3 - c_1c_3)) \\
&\quad +(s_1'c_2's_3' - c_1'c_3')(s_\beta(c_1c_2s_3 + s_1c_3) + c_\beta(s_1c_2s_3 - c_1c_3)) \\
&\quad +s_2's_3's_2s_3 \\
r_{12} &= (c_1'c_2's_3' + s_1'c_3')(c_\beta c_1s_2 - s_\beta s_1s_2) \\
&\quad +(s_1'c_2's_3' - c_1'c_3')(s_\beta c_1s_2 + c_\beta s_1s_2) \\
&\quad -s_2's_3'c_2 \\
r_{13} &= (c_1'c_2's_3' + s_1'c_3')(c_\beta(c_1c_2c_3 - s_1s_3) - s\beta(s_1c_2c_3 + c_1s_3)) \\
&\quad +(s_1'c_2's_3' - c_1'c_3')(s_\beta(c_1c_2c_3 - s_1s_3) + c_\beta(s_1c_2c_3 + c_1s_3)) \\
&\quad +s_2's_3's_2c_3 \\
r_{21} &= c_1's_2'(c_\beta(c_1c_2s_3 + s_1c_3) - s_\beta(s_1c_2s_3 - c_1c_3)) \\
&\quad +s_1's_2'(s_\beta(c_1c_2s_3 + s_1c_3) + c_\beta(s_1c_2s_3 - c_1c_3)) \\
&\quad -c_2's_2s_3 \\
r_{22} &= c_1's_2'(c_\beta c_1s_2 - s_\beta s_1s_2) \\
&\quad +s_1's_2'(s_\beta c_1s_2 + c_\beta s_1s_2) \\
&\quad +c_2'c_2 \\
r_{23} &= c_1's_2'(c_\beta(c_1c_2c_3 - s_1s_3) - s\beta(s_1c_2c_3 + c_1s_3)) \\
&\quad +s_1's_2'(s_\beta(c_1c_2c_3 - s_1s_3) + c_\beta(s_1c_2c_3 + c_1s_3)) \\
&\quad -c_2's_2c_3 \\
r_{31} &= (c_1'c_2'c_3' - s_1's_3')(c_\beta(c_1c_2s_3 + s_1c_3) - s_\beta(s_1c_2s_3 - c_1c_3)) \\
&\quad +(s_1'c_2'c_3' + c_1's_3')(s_\beta(c_1c_2s_3 + s_1c_3) + c_\beta(s_1c_2s_3 - c_1c_3)) \\
&\quad +s_2'c_3's_2s_3 \\
r_{32} &= (c_1'c_2'c_3' - s_1's_3')(c_\beta c_1s_2 - s_\beta s_1s_2) \\
&\quad +(s_1'c_2'c_3' + c_1's_3')(s_\beta c_1s_2 + c_\beta s_1s_2) \\
&\quad -s_2'c_3'c_2' \\
r_{33} &= (c_1'c_2'c_3' - s_1's_3')(c_\beta(c_1c_2c_3 - s_1s_3) - s\beta(s_1c_2c_3 + c_1s_3)) \\
&\quad +(s_1'c_2'c_3' + c_1's_3')(s_\beta(c_1c_2c_3 - s_1s_3) + c_\beta(s_1c_2c_3 + c_1s_3)) \\
&\quad +s_2'c_3's_2c_3
\end{aligned}$$

$$
\begin{aligned}
\varrho_x &= (c_1'c_2's_3' + s_1'c_3')(c_\beta(L_4(c_1c_2c_3 - s_1s_3) + L_3c_1s_2 - L_2s_1 + b_x + d_x)) \\
&\quad +(c_1'c_2's_3' + s_1'c_3')(-s_\beta(L_4(s_1c_2c_3 + c_1s_3) + L_3s_1s_2 + L_2c_1 + b_y)) \\
&\quad +(s_1'c_2's_3' - c_1'c_3')(s_\beta(L_4(c_1c_2c_3 - s_1s_3) + L_3c_1s_2 - L_2s_1 + b_x + d_x)) \\
&\quad +(s_1'c_2's_3' - c_1'c_3')(c_\beta(L_4(s_1c_2c_3 + c_1s_3) + L_3s_1s_2 + L_2c_1 + b_y)) \\
&\quad +s_2's_3'(L_4s_2c_3 - L_3c_2 + L_1 + b_z) \\
&\quad -L_1s_2's_3' + L_2c_3' - b_x(c_1'c_2's_3' + s_1'c_3') - b_y(s_1'c_2's_3' - c_1'c_3') - b_z(s_2's_3') \\
\varrho_y &= c_1's_2'(c_\beta(L_4(c_1c_2c_3 - s_1s_3) + L_3c_1s_2 - L_2s_1 + b_x + d_x)) \\
&\quad +c_1's_2'(-s_\beta(L_4(s_1c_2c_3 + c_1s_3) + L_3s_1s_2 + L_2c_1 + b_y)) \\
&\quad +s_1's_2'(s_\beta(L_4(c_1c_2c_3 - s_1s_3) + L_3c_1s_2 - L_2s_1 + b_x + d_x)) \\
&\quad +s_1's_2'(c_\beta(L_4(s_1c_2c_3 + c_1s_3) + L_3s_1s_2 + L_2c_1 + b_y)) \\
&\quad -c_2'(L_4s_2c_3 - L_3c_2 + L_1 + b_z) \\
&\quad +L_1c_2' - L_3 - b_xc_1's_2' - b_ys_1's_2' + b_zc_2' \\
\varrho_z &= (c_1'c_2'c_3' - s_1's_3')(c_\beta(L_4(c_1c_2c_3 - s_1s_3) + L_3c_1s_2 - L_2s_1 + b_x + d_x)) \\
&\quad +(c_1'c_2'c_3' - s_1's_3')(-s_\beta(L_4(s_1c_2c_3 + c_1s_3) + L_3s_1s_2 + L_2c_1 + b_y)) \\
&\quad +(s_1'c_2'c_3' + c_1's_3')(s_\beta(L_4(c_1c_2c_3 - s_1s_3) + L_3c_1s_2 - L_2s_1 + b_x + d_x)) \\
&\quad +(s_1'c_2'c_3' + c_1's_3')(c_\beta(L_4(s_1c_2c_3 + c_1s_3) + L_3s_1s_2 + L_2c_1 + b_y)) \\
&\quad +s_2'c_3'(L_4s_2c_3 - L_3c_2 + L_1 + b_z) \\
&\quad -L_1s_2'c_3' - L_2s_3' - L_4 - b_x(c_1'c_2'c_3' - s_1's_3') - b_y(s_1'c_2'c_3' + c_1's_3') - b_z(s_2'c_3')
\end{aligned}
$$

5.5 Kalibrierung der Kamera

Um die Tranformation des Bildes nach Gleichung 4.9 und Gleichung 4.10 bzw. nach Gleichung 4.11 und Gleichung 4.12 vornehmen zu können, muß die Brennweite der Kamera bekannt sein. Ferner setzt die Transformation voraus, daß die Pixel des Bildes quadratisch sind. Ist dies nicht der Fall, dann müssen die aufgenommenen Bilder entsprechend skaliert werden.

Mit der bei der vorliegenden Arbeit eingesetzten Kamera wurde ein Kalibrierungsbild aufgenommen. Zur Kalibrierung wurde ein weißes Blatt, auf dem sich 6 $5 \times 5cm^2$ große Quadrate im Abstand von 5cm befunden haben, verwendet. Eine Skalierung der Bilder wird in der vorliegenden Arbeit nicht durchgeführt, da keine Veränderung der Form des Kalibrierungsblattes im digitalisierten Bild festgestellt wurde. Aus der Größe der Quadrate im Bild der Kamera wurde die Brennweite in Pixeln berechnet. Für die vorliegende Arbeit wurde f auf 792 gesetzt. Eine exakte Bestimmung der internen Parameter der Kamera [Jain et al. 95] wird hier nicht vorgenommen.

5.6 Extraktion bewegter Objekte

Um Objekte zu extrahieren, die sich selbst bewegen, wird in der vorliegenden Arbeit zunächst die Eigenbewegung kompensiert. Danach wird auf approximiert stationären Bildern weitergearbeitet. Dieser Ansatz wurde gewählt, weil der Ansatz auch für große Zeitdifferenzen zwischen den einzelnen Bildern geeignet ist. Bei großen Zeitdifferenzen ist die resultierende Bewegung

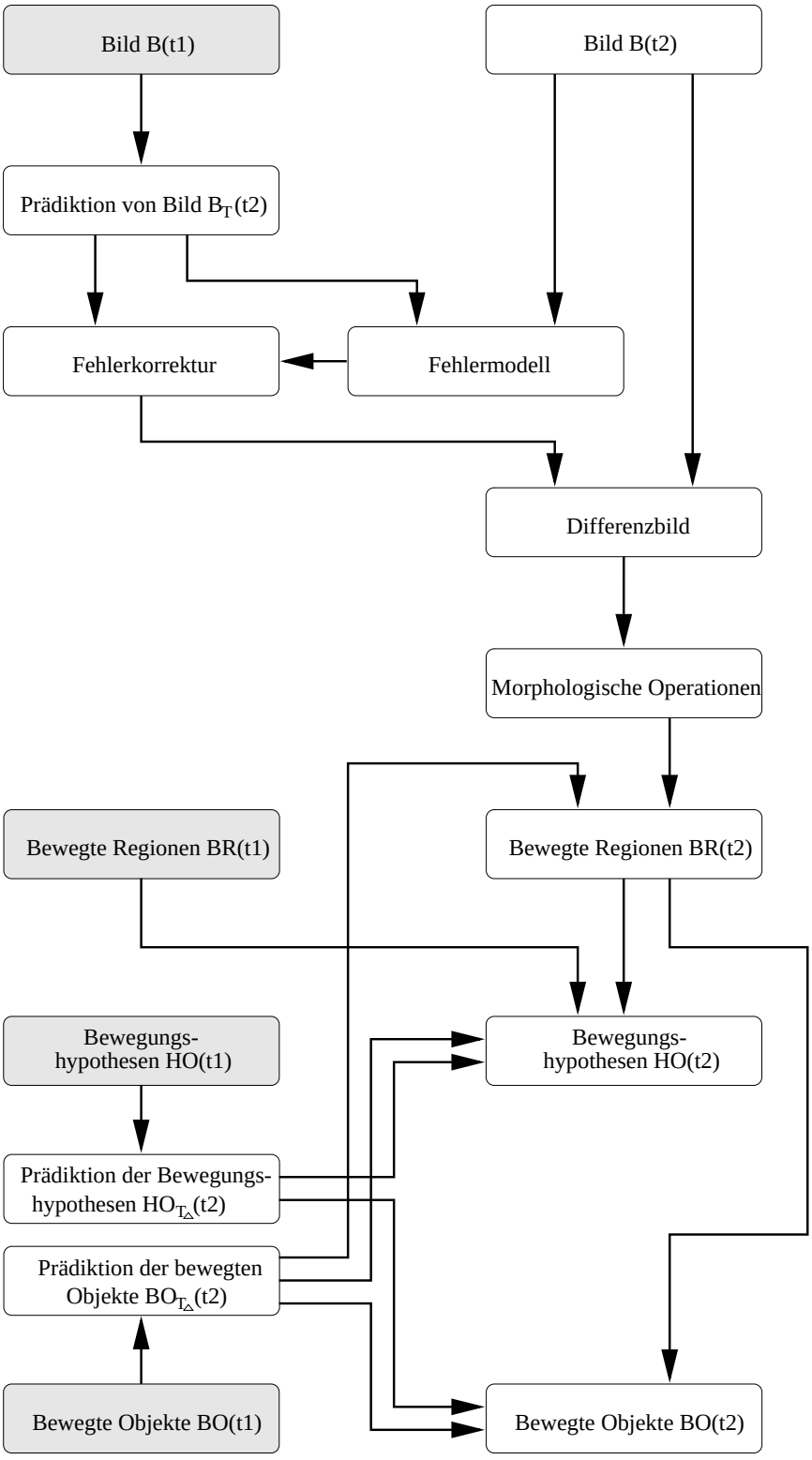

Abbildung 5.5: Der Algorithmus zur Extraktion bewegter Objekte

der einzelnen Bildpunkte besonders stark. Zudem kann sich der Abstand, mit dem die Bilder einer Bildsequenz aufgenommen werden, sich von einem Bild zum nächsten verändern. Der Datenfluß des vollständigen Algorithmuses ist in Abbildung 5.5 graphisch dargestellt. Die Komplexität der entwickelten Algorithmen wird in der Θ-Notation [Cormen et al. 90] angegeben.

5.6.1 Daten, die aus einer vorhergehenden Iteration existieren

Der Algorithmus bekommt in der aktuellen Iteration ein Bild $B(t_2)$ zur Verfügung gestellt, das zum Zeitpunkt t_2 aufgenommen sei. Aus der vorhergehenden Iteration steht das Bild $B(t_1)$ zur Verfügung, das zum Zeitpunkt t_1 aufgenommen sei. Außerdem existieren die folgenden Mengen ebenfalls aus der vorhergehenden Iteration des Algorithmuses. Die Mengen seien bei der ersten Iteration des Algorithmuses leer.

- $BR(t_1)$

  Die Menge $BR(t_1)$ enthält die bewegten Regionen, die für das Bild zum Zeitpunkt t_1 bestimmt wurden. Eine bewegte Region entsteht durch das Zusammenfassen von Regionen, die durch Veränderungen im Differenzbild entstanden sind. Wie die Veränderungen des Differenzbildes zu bewegten Regionen $BR(t_1)$ zusammengefaßt werden, wird weiter unten beschrieben.

  Jedes Element $R \in BR(t_1)$ ist ein konvexes Polygon. Die Fläche von R wird durch A_R bezeichnet, der Schwerpunkt durch $(s_x(R), s_y(R))$. Als Schwerpunkt des Polygons wurde der Schwerpunkt der einzelnen Punkte des Polygons herangezogen.

- $BH(t_1)$

  Die Menge $BH(t_1)$ enthält die Bewegungshypothesen zum Zeitpunkt t_1. Jedes Element $H \in BH(t_1)$ ist ein Polygon, das vermutlich ein bewegtes Objekt beschreibt. Wenn die Hypothese H in späteren Bildern validiert werden kann, dann wird H zu einem bewegten Objekt.

- $BO(t_1)$

  Die Menge $BO(t_1)$ enthält die bewegten Objekte zum Zeitpunkt t_1. Jedes Element $O \in BO(t_1)$ ist ein Polygon, das ein bewegtes Objekt beschreibt. Das bewegte Objekt O kann später wieder zu einer Hypothese werden, wenn es in folgenden Bildern nicht mehr wiedergefunden wird.

Die Elemente aus BH und BO besitzen die folgenden Eigenschaften und Werte.

- Schwerpunkt des Polygons (s_x, s_y)
- Geschwindigkeitsvektor des Polygons (v_x, v_y)
- Ein Wert, der angibt, wie oft die Hypothese mit der Wirklichkeit übereinstimmen muß, bevor sie validiert wird. In der vorliegenden Arbeit wird dieser Wert auf 1 gesetzt.
- Die Lebensdauer des Objektes. Die Lebensdauer wird in Iterationen des Algorithmus angegeben. In der vorliegenden Arbeit wird eine Lebensdauer von 4 Iterationen verwendet.
- Eine maximale Anzahl an Versuchen, die eine Hypothese hat, bevor sie gelöscht wird.
- Fläche des Polygons

5.6.2 Berechnung der Transformationsmatrix

Nun wird die Transformationsmatrix T berechnet. Die Transformationsmatrix T hängt von den Parametern $\theta_1(t_1)$, $\theta_2(t_1)$, $\theta_3(t_1)$, $\theta_1(t_2)$, $\theta_2(t_2)$, $\theta_3(t_2)$, v, w und $\delta t = t_2 - t_1$ ab. Mit dieser Transformation wird das Bild $B(t_1)$ auf das Bild $B_T(t_1)$ transformiert. Dabei sollte das Bild $B_T(t_1)$ möglichst genau auf das echte Bild $B(t_2)$ transformiert werden. Für die Berechnung der Transformationsmatrix T ist es also wichtig, die Parameter möglichst exakt zu bestimmen.

Um eine exakte Transformation T zu berechnen, wäre es am geeignetsten, die Positionen und Geschwindigkeiten direkt aus dem Kontrollalgorithmus [An et al. 88] des Fahrzeuges und der Kamera zu verwenden. Auf diese Daten kann jedoch nicht direkt zugegriffen werden. Die Position der Kamera wird über den Befehl `STEP POS` [Robosoft 94b] abgefragt. Als Ergebnis werden die Winkel der einzelnen Gelenke zurückgeliefert. Die Geschwindigkeit des Fahrzeuges wird mit dem Befehl `MOTV ST` [Robosoft 94a] abgefragt. Als Ergebnis werden die lineare Geschwindigkeit v und die Winkelgeschwindigkeit w des Fahrzeuges zurückgeliefert.

Falls der Algorithmus zur Aufnahme der Bilder schnell genug ist, kann es vorkommen, daß die Aufnahme der Bilder schneller ist, als die bereitgestellten Statusinformationen. Aufgrund der durch die Funkverbindung gegebenen Verzögerung können die Statusinformationen des Fahrzeugs und der Kamera, die von EMO benötigt werden, nur in Abständen von etwa $0,357s$ bestimmt werden. Daher kann es passieren, daß die von der Statusabfrage zurückgelieferten Positionen sich von einem Bild zum nächsten nicht verändert haben. In Wirklichkeit kann sich die Kamera aber bewegt haben. Da die Abfrage der Positionen nicht zum exakt gleichen Zeitpunkt wie das Grabben des Bildes erfolgt, besteht zudem noch das Problem, daß die Positionsangaben nicht völlig exakt zum Bild passen. Um beide Probleme zu lösen, wird zur Berechnung der Transformationsmatrix ${}^{C(t_2)}_{C(t_1)}T$ die Position der Kameragelenke zum Zeitpunkt t_1 und deren Geschwindigkeit zum Zeitpunkt t_1 eingesetzt. Die Geschwindigkeit der Gelenke wird aus zwei aufeinander folgenden Statusabfragen der Position der Gelenke berechnet.

Diese Geschwindigkeiten werden nun benutzt, um die gelieferten Positionsdaten zu verbessern. Die Gelenkpositionen $\theta_1(t_2)$, $\theta_2(t_2)$ und $\theta_3(t_2)$ werden einfach aus der Position und Geschwindigkeit, die sie zum Zeitpunkt t_1 besaßen, neu berechnet.

$$\begin{aligned}
\theta_1(t_2) &= \theta_1(t_1) + \dot{\theta}_1(t_1)(t_2 - t_1) \\
\theta_2(t_2) &= \theta_2(t_1) + \dot{\theta}_2(t_1)(t_2 - t_1) \\
\theta_3(t_2) &= \theta_3(t_1) + \dot{\theta}_3(t_1)(t_2 - t_1)
\end{aligned}$$

Aus den Parametern $\theta_1(t_1)$, $\theta_2(t_1)$, $\theta_3(t_1)$, $\theta_1(t_2)$, $\theta_2(t_2)$, $\theta_3(t_2)$, v, w und $\delta t = t_2 - t_1$ wird nun die Transformationsmatrix T berechnet.

5.6.3 Prädiktion der Objektbewegung

Die Objekte in $BO(t_1)$ und $HO(t_1)$ werden entsprechend ihres Bewegungsvektors auf die Position gebracht, an der sie zum Zeitpunkt t_2 vermutet werden. Die einzelnen Punkte der Objekte werden also wie folgt bewegt.

$$\begin{aligned}
x(t_2) &= x(t_1) + v_x(t_1)(t_2 - t_1) \\
y(t_2) &= y(t_1) + v_y(t_1)(t_2 - t_1)
\end{aligned}$$

Schließlich wird die Position der Objekte noch aufgrund der Eigenbewegung der Kamera angepaßt. Die einzelnen Punkte des Objektes werden also noch entsprechend der folgenden Gleichungen transformiert.

$$
\begin{aligned}
x_{T_\Delta}(t_2) &= f\frac{r_{11}x(t_2) + r_{12}y(t_2) + fr_{13} + f\frac{\varrho_x}{Z}}{r_{31}x(t_2) + r_{32}y(t_2) + fr_{33} + f\frac{\varrho_z}{Z}} + \Delta x \\
y_{T_\Delta}(t_2) &= f\frac{r_{21}x(t_2) + r_{22}y(t_2) + fr_{23} + f\frac{\varrho_y}{Z}}{r_{31}x(t_2) + r_{32}y(t_2) + fr_{33} + f\frac{\varrho_z}{Z}} + \Delta y
\end{aligned}
$$

Die so bewegten und transformierten Objekte werden mit $BO_{T_\Delta}(t_2)$ und $HO_{T_\Delta}(t_2)$ bezeichnet.

5.6.4 Korrelation markanter Punkte

Mit einem 5×5 Moravec-Operator [Moravec 77] werden markante Punkte $\check{B}(t_2)$ im Bild $B(t_2)$ bestimmt. Die Koordinaten der markanten Punkte $\check{B}(t_1)$ von Bild $B(t_1)$ werden mit der Transformationsmatrix T entsprechend Gleichung 4.9 und Gleichung 4.10 transformiert. Um den Einfluß von Deformationen zu verringern wird die Korrelation auf dem transformierten Bild durchgeführt [Zheng et al. 95] Als Entfernung der Punkte wird eine geschätzte Entfernung zum Hintergrund verwendet. Die so transformierten Punkte werden mit $\check{B}_T(t_1)$ bezeichnet. Zu jedem Punkt aus $\check{B}_T(t_1)$ wird ein korrespondierender Punkt aus $\check{B}(t_2)$ durch Korrelation [Jain et al. 95] bestimmt. Für jeden markanten Punkt $(x_1, y_1) \in \check{B}_T(t_1)$ wird der Korrelationswert c mit jedem markanten Punkt $(x_2, y_2) \in \check{B}(t_2)$ berechnet. Dazu wird um jeden markanten Punkt eine $w \times h$ große Region definiert. Der Korrelationswert c berechnet sich dann wie folgt:

$$
c = \lambda \sum_{-\frac{w}{2} \le j \le \frac{w}{2}} \sum_{-\frac{h}{2} \le k \le \frac{h}{2}} \left(B(x_1 + j, y_1 + k, t_1) - B(x_2 + j, y_2 + k, t_2)\right)^2
$$

mit

$$
\lambda = \left(1 + \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\right).
$$

Es wird also geprüft, wie ähnlich die beiden Regionen um die markanten Punkte sind. Der Gewichtungsfaktor λ ist dazu da, nahe beieinanderliegende Punkte zu bevorzugen. Denn durch die Prädiktion der Bildbewegung sind die Punkte schon ungefähr an der Position, an der sie sich tatsächlich befinden. Also muß für jeden Punkt $(x_1, y_1) \in \check{B}_T(t_1)$ ein markanter Punkt $(x_2, y_2) \in \check{B}_(t_2)$ in der Nähe von (x_1, y_1) gesucht werden. Der Gewichtungsfaktor λ ist dazu da, entfernten Punkten einen größeren Korrelationswert zuzuweisen. Zheng et al. [Zheng et al. 95] setzen ebenfalls eine gewichtete Korrelation ein. Allerdings gewichten sie einzelne Pixel bei der Berechnung des Korrelationswertes um den Einfluß von Deformationen zu verringern. Hier wird ein Gewichtungsfaktor eingesetzt, um zu erreichen, daß eine Korrespondenz naher Regionen hergestellt wird.

Der Punkt (x_2, y_2) wird als korrespondierender Punkt zu (x_1, y_1) ausgewählt, wenn er den minimalen Korrelationswert c hat, und dieser Wert kleiner als ein Schwellwert ist. Die Größe der Region ist 10×10 für ein 384×288 Bild. Besitzen die Bilder eine andere Größe, so wird

der Bereich entsprechend angepaßt. Für Bilder der Größe $w_B \times h_B$ berechnet sich der Bereich nach:

$$\begin{aligned} w &= 10\frac{w_B}{384} \\ h &= 10\frac{h_B}{288} \end{aligned}$$

Der Schwellwert wird eingesetzt, um zu gewährleisten, daß nicht irgendeine, sondern nur eine passende Region gewählt wird. Ein von der Größe der Region abhängiger Schwellwert von $3000wh$ wurde für die unten beschriebenen Daten eingesetzt.

Der Moravec-Operator wurde mit der Komplexität $\Theta(whw_Bh_B)$ implementiert. Dabei gibt $w \times h$ die Größe der Region an und $w_B \times h_B$ die Größe des Bildes. Der Aufwand der Korrelation ist $\Theta(n_1n_2wh)$. Dabei gibt $w \times h$ wieder die Größe der Region an. Die Zahl der markanten Punkte in $\check{B}(t_1)$ sei n_1, die in $\check{B}(t_2)$ sei n_2.

5.6.5 Berechnung der geschätzten Entfernung zum Hintergrund

Für je zwei markante Punkte $(x_1, y_1) \in \check{B}(t_1)$ und $(x_2, y_2) \in \check{B}(t_2)$ zwischen denen eine Korrespondenz hergestellt wurde, wird jetzt die Entfernung des Punktes berechnet. Die Entfernung kann auf zwei Arten berechnet werden. Dazu wurden Gleichung 4.7 und Gleichung 4.8 umgeformt. Beide Gleichungen liefern dieselbe Z-Koordinate. Da jedoch auf einem digitalisierten Bild gearbeitet wird, und zudem keine völlig exakten Daten aus der Transformationsmatrix zur Verfügung stehen, unterscheiden sich die tatsächlich berechneten Werte. Daher werden zunächst beide Werte berechnet. Der Wert, der durch die Bewegung des Punktes in x-Richtung entstanden ist, sei Z_x und der Wert, der durch die Bewegung des Punktes in y-Richtung entstanden ist, sei Z_y. Die Berechnung der Z-Koordinaten ist von je einer Bedingung abhängig.

$$\begin{aligned} Z_x &= \frac{f\varrho_x - x_2\varrho_z}{\frac{x_2}{f}(r_{31}x_1 + r_{32}y_1 + fr_{33}) - (r_{11}x_1 + r_{12}y_1 + fr_{13})} \quad \text{falls} \quad |\hat{x}_1 - \tilde{x}_1| > 0 \\ Z_y &= \frac{f\varrho_y - y_2\varrho_z}{\frac{y_2}{f}(r_{31}x_1 + r_{32}y_1 + fr_{33}) - (r_{21}x_1 + r_{22}y_1 + fr_{23})} \quad \text{falls} \quad |\hat{y}_1 - \tilde{y}_1| > 0 \end{aligned}$$

Werden beide Z-Koordinaten Z_x und Z_y berechnet, so wird deren Mittelwert gebildet. Falls nur eine der Koordinaten berechnet werden kann, so wird diese weiterverwendet. Es kann auch vorkommen, daß keine der beiden Werte berechnet werden konnte.

Das Kriterium für die Berechnung der Z-Koordinaten soll nun näher betrachtet werden. Es gilt:

$$\begin{aligned} \hat{x}_1 &= f\frac{r_{11}x_1 + r_{12}y_1 + fr_{13} + f\varrho_x}{r_{31}x_1 + r_{32}y_1 + fr_{33} + f\varrho_z} \\ \tilde{x}_1 &= f\frac{r_{11}x_1 + r_{12}y_1 + fr_{13}}{r_{31}x_1 + r_{32}y_1 + fr_{33}} \\ \hat{y}_1 &= f\frac{r_{21}x_1 + r_{22}y_1 + fr_{23} + f\varrho_y}{r_{31}x_1 + r_{32}y_1 + fr_{33} + f\varrho_z} \\ \tilde{y}_1 &= f\frac{r_{21}x_1 + r_{22}y_1 + fr_{23}}{r_{31}x_1 + r_{32}y_1 + fr_{33}} \end{aligned}$$

Dabei gibt $(\hat{x}_1, \hat{y}_1)$ die Koordinaten des transformierten Bildpunktes zum Zeitpunkt t_2 an, wenn sich dieser Punkt zum Zeitpunkt t_1 an der Position (x_1, y_1) befunden hat und durch

einen Punkt im Abstand 1m von der Kamera erzeugt wurde. $(\tilde{x}_1, \tilde{y}_1)$ gibt die Koordinaten des transformierten Bildpunktes desselben Punktes an, ohne die translatorische Bewegung der Kamera zu berücksichtigen.

Demnach wird nur dann eine Z-Koordinate berechnet, wenn die Bewegung des Fahrzeuges groß genug war, um eine Bewegung von mindestens einem Pixel des Bildpunktes zu verursachen, wenn der Hintergrund sich in einer Entfernung von 1m befunden hat. Diese Referenzfläche im Abstand von einem Meter wurde gewählt, damit eine schnelle Anpassung der Z-Koordinaten gewährleistet ist. Als Kriterium hätte auch $|\hat{x}_1 - \tilde{x}_1| > 0$ und $|\hat{y}_1 - \tilde{y}_1| > 0$ mit

$$\begin{aligned} \hat{x}_1 &= f\frac{r_{11}x_1 + r_{12}y_1 + fr_{13} + f\frac{\varrho_x}{\tilde{Z}}}{r_{31}x_1 + r_{32}y_1 + fr_{33} + f\frac{\varrho_z}{\tilde{Z}}} \\ \hat{y}_1 &= f\frac{r_{21}x_1 + r_{22}y_1 + fr_{23} + f\frac{\varrho_y}{\tilde{Z}}}{r_{31}x_1 + r_{32}y_1 + fr_{33} + f\frac{\varrho_z}{\tilde{Z}}} \end{aligned}$$

verwendet werden können. Dann wäre statt einer Referenzfläche im Abstand von einem Meter die Ebene mit der geschätzten Entfernung zum Hintergrund verwandt worden. Also wäre der Abstand nur dann berechnet worden, wenn die Bewegung der Kamera für die geschätzte Entfernung zum Hintergrund groß genug gewesen wäre, um eine sinnvolle Berechnung des Abstands zu ermöglichen. Dies hätte allerdings zur Folge, daß der Algorithmus keine neuen Abstandswerte berechnet hätte, wenn ein Wechsel von einem weit entfernten auf einen nahen Hintergrund stattgefunden hätte. Denn für große Entfernungen gilt $\lim_{\tilde{Z}\to\infty} \hat{x} = \tilde{x}$ und $\lim_{\tilde{Z}\to\infty} \hat{y} = \tilde{y}$. Also wären die Tests $|\hat{x}_1 - \tilde{x}_1| > 0$ und $|\hat{y}_1 - \tilde{y}_1| > 0$ für große Entfernungen $\tilde{Z}$ immer falsch. Der Algorithmus hätte daher keine Möglichkeit bei einem Wechsel von einem weit entfernten Hintergrund auf einen nahen, die geschätzte Entfernung zum Hintergrund anzupassen.

Für jedes korrespondierende Punktpaar wird nach Möglichkeit die zugehörige Z-Koordinate, wie eben beschrieben, berechnet. Die so berechneten Entfernungen werden in einer Liste L_B gesammelt. Negative Abstandswerte werden nicht in die Liste L_B eingetragen. Negative Werte können durch Wackler der Kamera verursacht werden. Es kann auch vorkommen, daß eine Korrespondenz zwischen zwei Punkten hergestellt wurde, die sich auf einem Objekt befinden, das sich bewegt, also nicht Teil des Hintergrundes ist. Für korrespondierende Punkte, die sich auf einem bewegten Objekt befinden, kann die obige Berechnung natürlich nicht angewandt werden. Daher dürfen diese Punkte auch nicht zur Bestimmung des geschätzten Abstands zum Hintergrund herangezogen werden. Solange jedoch noch kein bewegtes Objekt im Bild extrahiert worden ist, kann nicht zwischen interessanten Punkten auf einem bewegten Objekt und interessanten Punkten des Hintergrundes unterschieden werden. Daher wird der Median aus der Liste L_B bestimmt. Dieser Wert ist dann die geschätzte Entfernung zum Hintergrund zum Zeitpunkt $\tilde{Z}(t_1)$.

Die eben berechnete Entfernung setzt voraus, daß die Transformationsmatrix T relativ exakt berechnet wurde. Da jedoch die Parameter, die zur Berechnung der Transformationsmatrix T herangezogen werden, nicht völlig genau sind, und zudem Wackler die Bildbewegung beeinflussen können, kann es immer noch zu Ausreißern kommen. Daher wird eine weitere Liste L_z eingesetzt. Ist die Entfernung $\tilde{Z}(t_1)$ größer als 50cm, so wird sie in der Liste L_z gespeichert. Der Schwellwert von 50cm wurde gewählt, um zu kleine Entfernungen auszuschließen. Wird eine neue Entfernung $\tilde{Z}(t_1)$ in die Liste L_z eingetragen, so wird dabei der älteste Wert in der Liste L_z überschrieben. Der Median der Liste L_z wird zur aktuellen Entfernung $\tilde{Z}$ zum Hintergrund. Da der Median der Liste zur geschätzten Entfernung zum Hintergrund wird, darf

diese Liste nicht zu groß sein. Die Liste L_z enthalte l Werte. Dann dauert es mindestens die halbe Anzahl an Bildern $\frac{l}{2}$ bis sich die geschätzte Entfernung zum Hintergrund an eine neue Umgebung anpaßt. Die Liste L_z muß aber auch groß genug sein, um Ausreißer zu eliminieren. Daher muß für die Größe der Liste ein geeigneter Wert gefunden werden. Für die vorliegende Arbeit wurde eine Liste L_z mit 5 Werten eingesetzt.

Die Liste L_z enthält Entfernungen, die zu unterschiedlichen Zeitpunkten bestimmt wurden. Für eine möglichst genaue Schätzung der Entfernung zum Hintergrund ist es wichtig, die alten Werte in der Liste entsprechend der Bewegung der Kamera anzupassen. Daher werden die in der Liste L_z gespeicherten Z-Koordinaten vor dem Eintrag eines neuen Wertes noch um die zurückgelegte Strecke in Blickrichtung der Kamera ϱ_z angepaßt.

5.6.6 Fehlerkorrektur

Um Schwingungen, die durch den fahrenden Roboter verursacht werden und ungenaue Daten bei der Positionsmessung der Gelenke und bei der Messung der Geschwindigkeit des Fahrzeuges zu korrigieren, wird eine Fehlerkorrektur durchgeführt. Dazu wird für jedes korrespondierende Punktpaar $(x_1, y_1) \in \check{B}_T(t_1)$ und $(x_2, y_2) \in \check{B}(t_2)$ die Differenz $x_2 - x_1$ in der Liste L_x und die Differenz $y_2 - y_1$ in der Liste L_y gespeichert. Der Median der Liste L_x sei Δx und der Median der Liste L_y sei Δy. Das Bild ist also zusätzlich zur Transformation durch die Matrix T noch um Δx Pixel in X-Richtung und um Δy Pixel in Y-Richtung zu verschieben. Dieses relativ einfache Fehlermodell ist vor allem zur Elimination von Schwingungen geeignet.

5.6.7 Detektion von Veränderungen

Das Bild $B(t_1)$ wird anhand von Gleichung 4.11 und Gleichung 4.12 transformiert. Das durch die homogene Transformation T vorhergesagte und anschließend um $(\Delta x, \Delta y)$ verschobene Bild werde mit $B_{T_\Delta}(t_1)$ bezeichnet. Das Differenzbild

$$\frac{\partial B(x, y, t)}{\partial t} \approx B(x, y, t + \delta t) - B_{T_\Delta}(x, y, t)$$

wird nun berechnet.

Dann wird das Absolutbild des Differenzbildes gebildet. Das Absolutbild wird durch die Anwendung einiger morphologischer Operatoren [Murray et al. 94a] wie in Abschnitt 4.3 beschrieben weiterbearbeitet. Wie bei der Korrelation markanter Punkte wird auch hier das Strukturelement an die Größe des Bildes angepaßt. Die angegebenen Größen der Strukturelemente werden bei einem Bild der Größe 384×288 eingesetzt. Wenn ein kleineres Bild verarbeitet wird, wird die Größe des Strukturelements proportional zur Bildgröße angepaßt.

- Schließung mit einem kleinen quadratischen 5×5-Strukturelement , um Bereiche mit viel Bewegung miteinander zu verbinden.
- Öffnung mit einem 9×9-Strukturelement, um Störungen zu unterdrücken.

Anschließend werden Pixelwerte, die unterhalb 15% des maximal möglichen Pixelwertebereiches liegen, unterdrückt. Somit ist ein Binärbild entstanden. Jede Region dieses Binärbildes ist entweder durch ein bewegtes Objekt oder durch Fehler in der Prädiktion entstanden. In folgenden Schritten wird versucht, Regionen, die durch Fehler in der Prädiktion entstanden sind, zu eliminieren.

Die Transformation wurde mit der Komplexität $\Theta(w_B h_B)$ implementiert. Die morphologischen Operatoren mit der Komplexität $\Theta(whw_B h_B)$. Dabei gibt $w \times h$ die Größe der Region an und $w_B \times h_B$ die Größe des Bildes.

5.6.8 Konturbildung

Die Kontur der Regionen, die nach der Anwendung morphologischer Operatoren übriggeblieben sind, werden ebenfalls durch morphologische Operatoren bestimmt [Gonzales et al. 92].

$$\beta(B) = B - (B \ominus M)$$

Das Konturbild $\beta(B)$ von Bild B ergibt sich durch Erosion von B durch M und anschließender Subtraktion des Ergebnisses von B. Für M wird ein 3×3 Strukturelement verwendet. Übrig bleibt die Kontur der Fläche. Die Regionen werden als Punktmengen $\{(x,y)|(x,y) \in \beta(B)\}$ gespeichert. Da bei dem Verfahren eventuell vorhandene Löcher in den Regionen auch als Kontur auftreten, werden die Punktmengen der Löcher nachträglich gelöscht. Es bleiben nur äußere Konturen, also Konturen, die von keiner anderen Kontur umschlossen werden, übrig. Die Menge der äußeren Konturen wird mit K bezeichnet.

5.6.9 Zusammenfassen nahe beieinanderliegender Konturen

Nahe beieinanderliegende Konturen werden wie in Abschnitt 4.4 beschrieben zusammengefaßt. Die Heuristik soll nun formalisiert werden. Es wird folgende Relation definiert. K seien die Konturen der Bereiche des Differenzbildes.

$$\sim' \quad = \{(A,B) \quad | \quad A \in K, \quad B \in K, \quad \text{nahe}(A,B)\}$$

Um zu bestimmen, ob zwei Bereiche des Differenzbildes nahe beieinander sind, wird das Prädikat nahe wie folgt definiert:

Für zwei Konturen A und B seien $(x_1, y_1) \in A$ und $(x_2, y_2) \in B$ die beiden Punkte mit dem geringsten Abstand. Das Prädikat nahe(A,B) ist genau dann wahr, wenn folgendes gilt:

$$(2x_2 - x_1, 2y_2 - y_1) \in R_B \quad \wedge \quad (2x_1 - x_2, 2y_1 - y_2) \in R_A$$

Dabei sind R_A die Punktmenge des durch die Kontur A umschlossenen Bereiches. R_B ist die Punktmenge des durch die Kontur B umschlossenen Bereiches.

In Worten ausgedrückt besagt obige Definition folgendes. Es werden die beiden nächsten Punkte der beiden Konturen gesucht. Zwischen ihnen wird eine Strecke gelegt, und die Strecke wird in beiden Richtungen um die Länge der Strecke verlängert. Das Prädikat nahe(A,B) ist genau dann wahr, wenn die Endpunkte der Strecke innerhalb der zugehörigen Kontur liegen.

Die Relation $\sim$ sei die reflexive und transitive Hülle der Relation $\sim'$. Dann ist $\sim$ eine Äquivalenzrelation, da sie reflexiv, symmetrisch und transitiv ist [Griffiths et al. 76]. Die Relation $\sim$ teilt die Menge K in die Äquivalenzklassen $K/\sim$ ein. Für jede Äquivalenzklasse ist eine Kontur zu generieren. Mit einem von Wall et al. [Wall et al. 84] beschriebenen Algorithmus kann eine Kontur durch ein Polygon approximiert werden und somit die Anzahl der Punkte reduziert werden. Da hier jedoch mehrere Regionen zusammengefaßt werden und daher keine geschlossene Kontur vorhanden ist, wird die konvexe Hülle aller Punkte berechnet, die sich in einer Äquivalenzklasse befinden. Die Berechnung der konvexen Hülle eines Polygons mit n Punkten wurde mit der Komplexität $\Theta(n^2)$ (im ungünstigsten Fall) nach einem von

Sedgewick [Sedgewick 92] beschriebenen Verfahren implementiert. Von nun an wird auf den konvexen Hüllen der Äquivalenzklassen $K/\sim$ weitergearbeitet.

Um die Äquivalenzklassen $K/\sim$ zu bestimmen, wird eine Liste mit Klassen gebildet. Jede Klasse wird mit einer Kontur initialisiert. Für je zwei Konturen werden die beiden Punkte mit der kleinsten Entfernung bestimmt. Die Komplexität des implementierten Algorithmus ist $\Theta(n_1 n_2)$ wobei n_1 die Anzahl der Punkte der einen Kontur und n_2 die Anzahl der Punkte der anderen Kontur ist. Wie in Abschnitt 4.4 beschrieben wird nun geprüft, ob der Punkt $(2x_2 - x_1, 2y_2 - y_1)$ innerhalb von Polygon B und ob der Punkt $(2x_1 - x_2, 2y_1 - y_2)$ innerhalb von Polygon A liegt. Wenn dies zutrifft, so werden die beiden Klassen von A und B vereinigt. Der Test, ob ein Punkt innerhalb eines Polygons liegt, wurde nach einem von Sedgewick [Sedgewick 92] beschriebenen Verfahren mit der Komplexität $\Theta(n)$, wenn das Polygon n Punkte besitzt, implementiert.

5.6.10 Bestimmung bewegter Regionen

Für jedes Objekt $O \in BO_{T_\Delta}(t_2)$ werden die Konturen aus $K/\sim$ zusammengefaßt, die ganz in dem Objekt O enthalten sind oder das Objekt O schneiden. Schließlich wird die konvexe Hülle der so zusammengefaßten Punkte gebildet. Die entstandenen Regionen werden mit $BR(t_2)$ bezeichnet.

5.6.11 Erzeugung einer Bewegungshypothese

Die bewegten Regionen $BR(t_1)$, die bereits früher im Bild $B(t_1)$ gefunden wurden, werden auf die Position gebracht, an der sie in Bild $B(t_2)$ vermutet werden. Dabei wird sowohl die Kamerabewegung als auch die Fehlerkorrektur berücksichtigt. Die so transformierten und korrigierten Regionen werden mit $BR_{T_\Delta}(t_1)$ bezeichnet.

Zur Generierung neuer Bewegungshypothesen $HO(t_2)$ werden die bewegten Regionen $BR_{T_\Delta}(t_1)$ und $BR(t_2)$ herangezogen. Zwischen den Regionen $R_1 \in BR_{T_\Delta}(t_1)$ und $R_2 \in BR(t_2)$ wird eine Hypothese H aufgestellt, wenn es sich um Regionen handelt, die vermutlich das gleiche Objekt beschreiben. Als Kompatibilitätstest der Regionen können z.B. die Flächen beider Regionen herangezogen werden. Dieser relativ einfache Kompatibilitätstest wurde aus den in Abschnitt 4.6 beschriebenen Gründen eingesetzt. Die Flächen zweier Regionen seien A_1 und A_2. Die kleinere der beiden Flächen sei A_1. Von [Radig 81] wurde folgendes Kompatibilitätsmaß eingesetzt.

$$1 - \frac{|A_1 - A_2|}{|A_1 + A_2|}$$

Dabei gibt der Wert 0 an, daß die Regionen völlig unterschiedlich sind und der Wert 1 gibt an, daß die Flächen identisch sind. Für die vorliegende Arbeit werden die Regionen als kompatibel betrachtet, falls

$$\frac{A_1}{A_2} \geq \tau.$$

Demnach gelten zwei Regionen als kompatibel, wenn sich deren Flächen um höchstens $100\tau\%$ voneinander unterscheiden. Der Schwellwert wurde für die vorliegende Arbeit auf 50% gesetzt.

Der Bewegungsvektor (v_x, v_y) der Hypothese H wird wie folgt berechnet.

$$v_x(H) \quad = \quad \frac{s_x(R_2) - s_x(R_1)}{\delta t}$$

$$v_y(H) = \frac{s_x(R_2) - s_x(R_1)}{\delta t}$$

Dabei gibt $(s_x(P), s_y(P))$ den Schwerpunkt des Polygons P an.

$$s_x(P) = \frac{1}{||P||} \sum_{(x,y)\in P} x$$
$$s_y(P) = \frac{1}{||P||} \sum_{(x,y)\in P} y$$

Demnach wird jeder der Punkte des Polygons mit einer Einheitsmasse versehen und so der Schwerpunkt berechnet. Daher stimmt der so berechnete Schwerpunkt nicht mit dem Schwerpunkt der Fläche des Polygons überein.

Die Hypothese H wird zu $HO(t_2)$ hinzugefügt, falls der Schwerpunkt von H weder innerhalb einer bereits existierenden Hypothese $O_h \in HO(t_2)$ noch innerhalb eines bewegten Objektes $O_b \in BO(t_2)$ liegt.

5.6.12 Aktualisierung der Bewegungshypothesen und der bewegten Objekte

Um die Bewegungshypothesen $HO(t_2)$ und die bewegten Objekte $BO(t_2)$ zu bestimmen, werden die Objekte $BO_{T_\Delta}(t_2)$, $HO_{T_\Delta}(t_2)$ und die Regionen $BR(t_2)$ verwendet. $BO(t_2)$ und $HO(t_2)$ werden mit der leeren Menge initialisiert.

Für jedes Objekt $H \in HO_{T_\Delta}(t_2)$, das noch nicht seine Lebensdauer überschritten hat, werden die Konturen aus $BR(t_2)$ zusammengefaßt, die ganz in dem Objekt H enthalten sind oder das Objekt H schneiden. Ist die konvexe Hülle der zusammengefaßten Punkte R leer, so wird H zu den Bewegungshypothesen $HO(t_2)$ hinzugefügt. Ist R nicht leer, so wird R zur Menge der Bewegungshypothesen $HO(t_2)$ hinzugefügt, falls sie noch nicht validiert ist. Sie wird zu den bewegten Objekten $BO(t_2)$ hinzugefügt, wenn die Hypothese validiert ist. Für die vorliegende Implementierung wird die Hypothese sofort validiert, falls die Punktmenge nicht leer ist. Der Bewegungsvektor wird wie folgt aktualisiert, falls R nicht leer ist. Der Faktor $\frac{1}{2}$ dient der Stabilisierung des Algorithmuses. Die tatsächliche Position des Objektes wird zwischen der Position der Prädiktion $(s_x(H), s_y(H))$ und der neuen Position $(s_x(R), s_y(R))$ angenommen.

$$v_x(R) = v_x(H) + \frac{s_x(R) - s_x(H)}{2\delta t}$$
$$v_y(R) = v_y(H) + \frac{s_y(R) - s_y(H)}{2\delta t}$$

Wenn zwei Hypothesen in derselben Iteration validiert werden und diese dasselbe Objekt beschreiben, dann wird der Mittelwert des Bewegungsvektors für das bewegte Objekt berechnet und nur ein bewegtes Objekt erzeugt.

Die Aktualisierung der bewegten Objekte $BO(t_2)$ geschieht analog zur Aktualisierung der Bewegungshypothesen $HO(t_2)$. Für jedes Objekt $O \in BO_{T_\Delta}(t_2)$ werden die Konturen aus $BR(t_2)$ zusammengefaßt, die ganz in dem Objekt O enthalten sind oder das Objekt O schneiden. Trifft dies für keine Kontur aus $BR(t_2)$ zu, so wird mit dem nächsten Objekt O fortgefahren. Das Objekt, das sich aus der konvexen Hülle R der zusammengefaßten Regionen ergibt,

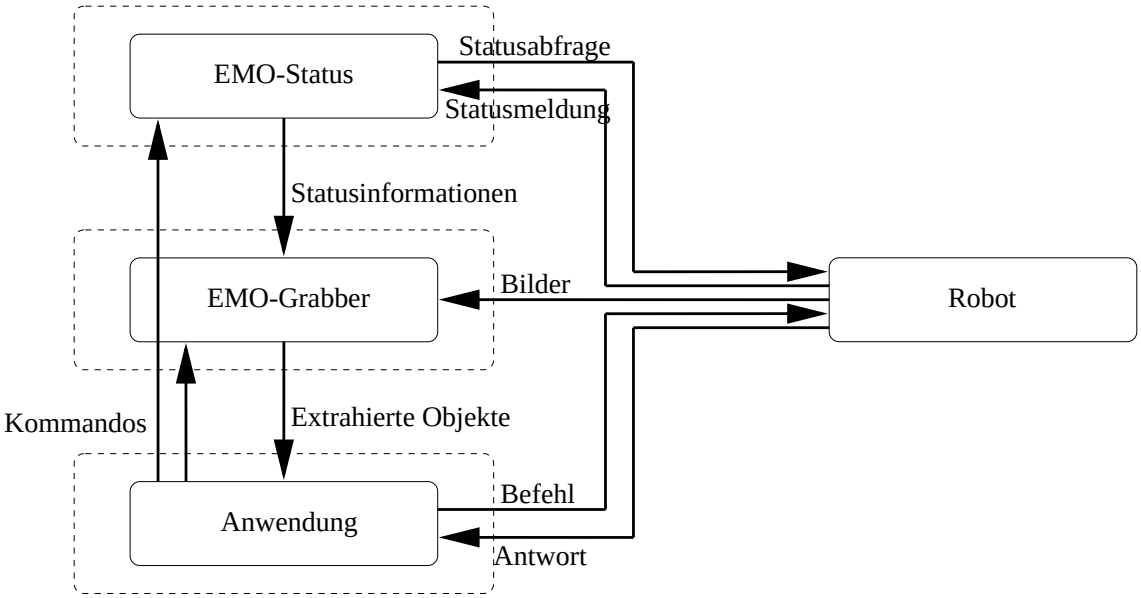

Abbildung 5.6: Kommunikationskanäle der einzelnen Prozesse

wird, falls R nicht leer ist, zur Menge der bewegten Objekte $BO(t_2)$ hinzugefügt. Existiert das Objekt bereits aufgrund der Validierung einer Hypothese, so wird das bereits existierende Objekt überschrieben. Der Bewegungsvektor wird wie folgt aktualisiert:

$$\begin{aligned} v_x(R) &= v_x(O) + \frac{s_x(R) - s_x(O)}{2\delta t} \\ v_y(R) &= v_y(O) + \frac{s_y(R) - s_y(O)}{2\delta t} \end{aligned}$$

Ist R leer, so wird O zu den Bewegungshypothesen $HO(t_2)$ hinzugefügt.

So wurden neue Bewegungshypothesen $BH(t_2)$ und bewegte Objekte $BO(t_2)$ generiert, die für die nächste Iteration des Algorithmuses zur Verfügung stehen.

5.7 Benutzung der Bibliothek

Das Programm EMO wurde zur Benutzung gemeinsam mit anderen Programmen entworfen. Dem Anwender stehen die folgenden Prozeduren zur Verfügung, um mit dem Modul EMO zu arbeiten.

```
emoType *emo_init(RoI_Handle robot,int width, int height, double scale, double
time, char *model, double z,int show,int viewW,int viewH);
```

Der Anwender muß als erstes die Routine `emo_init` aufrufen, damit das EMO-Modul initialisiert wird. Für jedes Fahrzeug ist ein Aufruf von `emo_init` erforderlich. Der Parameter `robot` gibt dabei das Fahrzeug an. Es handelt sich um das RoI-Handle [Levi et al. 95]. Vor der Benutzung von EMO muß eine Verbindung zu einem Fahrzeug durch `RoI_open` hergestellt worden

sein. Der Framegrabber muß ebenfalls vor dem Aufruf von `emo_init` initialisiert worden sein. Die Parameter `width` und `height` geben die Größe der vom Framegrabber gelieferten Bilder an.

Der Parameter `scale` gibt einen Skalierungsfaktor an. Denn die Größe der Bilder, die EMO intern verarbeitet, ist unabhängig von der Größe der durch den Framegrabber gelieferten Bilder. Die durch den Grabber gelieferten Bilder werden um den Faktor `scale` verkleinert. Der Parameter `scale` kann die Werte 0.25 bis 1 annehmen. Das heißt, die Bilder können maximal auf ein Viertel der gelieferten Größe verkleinert werden.

Der Parameter `time` gibt eine Verzögerungszeit an. Der Algorithmus wartet nach jeder Iteration für `time` Nanosekunden, bevor die nächste Iteration ausgeführt wird.

Der Parameter `model` gibt das Kameramodell an, das zur Berechnung der Transformationsmatrix verwendet wird. Als Kameramodel stehen derzeit `stereo1` und `stereo2` für die linke und rechte Stereokamera zur Verfügung. Das Modell `stat` kann für eine stationäre Kamera verwendet werden. Das Kameramodell wird spezifiziert, indem ein Zeiger auf einen der angegebenen Strings übergeben wird.

Der Parameter `show` gibt an, wie viele Fenster geöffnet werden sollen, um die Arbeitsweise von EMO zu verfolgen. `EMO_VIEWNONE` gibt an, daß keine Fenster geöffnet werden sollen. `EMO_VIEWONE` gibt an, daß lediglich das aktuelle Bild mit den extrahierten bewegten Objekten geöffnet werden soll. `EMO_VIEWALL` gibt an, daß alle Fenster von EMO geöffnet werden sollen. In diesen Fenstern können die einzelnen Schritte von EMO kontrolliert werden. Die Parameter `viewW` und `viewH` geben die Größe der Fenster in Pixeln an.

Nachdem `emo_init` ausgerufen wurde, werden zwei weitere Prozesse erzeugt. Ein Prozeß ist für die Datenabfrage zuständig. Er liest in einer Schleife die Statusinformationen des Fahrzeuges und der Kamera aus. Der zweite Prozeß ist für die Extraktion der bewegten Objekte zuständig. Da kontinuierlich Bilder verarbeitet werden müssen, arbeitet dieser Prozeß völlig getrennt vom aufrufenden Programm. Der Austausch der Daten erfolgt über Shared-Memory-Segmente.

Die Kontrolle wird schließlich an den Aufrufer zurückgegeben. Er erhält ein Handle, über das die gewünschten Informationen aus dem Modul EMO abgefragt werden können. Der Aufrufer kann nun den Roboter und die Kamera völlig beliebig steuern. EMO extrahiert dann im Hintergrund die bewegten Objekte. Wenn der Aufrufer wissen möchte, welche bewegten Objekte vorhanden sind, so kann er dies, wie weiter unten beschrieben, abfragen. Tritt ein Fehler auf, so wird `NULL` zurückgeliefert.

```
int emo_takeMovie(emoType *emoHandle,int len,int loopFlag);
```

Mit der Routine `emo_takeMovie` kann ein Film vom Modul EMO aufgenommen werden. Als Parameter werden das Handle `emoHandle` übergeben. Der Parameter `len` gibt an, wie viele Bilder der Reihe nach abgespeichert werden sollen. Der Parameter `loopFlag` gibt an, ob die Bilder kontinuierlich gespeichert werden sollen. Wenn die Bilder nicht kontinuierlich gespeichert werden, dann werden, nachdem `len` Bilder gespeichert wurden, keine weiteren Bilder mehr abgespeichert. Sonst werden die Bilder immer wieder überschrieben. Wenn keine Fehler aufgetreten sind, wird der Wert `EMO_NOERR` zurückgeliefert.

```
int emo_saveMovie(emoType *emoHandle,char *name);
```

Mit der Routine `emo_saveMovie` kann die zuvor mit der Routine `emo_takeMovie` aufgenommene Videosequenz abgespeichert werden. Als Parameter wird wieder das Handle `emoHandle` übergeben. Der Parameter `name` ist ein Zeiger auf einen String, der den Namen enthält, unter dem die Videosequenz abgespeichert wird.

Die Bilder der zuvor aufgenommenen Sequenz werden im PGM-Format [Murray et al. 94b] gespeichert. Dem Parameter `name` wird für jedes Bild die Erweiterung `x.pgm` angehängt. Dabei steht `x` für die Nummer des Bildes. Das erste Bild trägt die Nummer 0. Wenn keine Fehler aufgetreten sind, wird der Wert `EMO_NOERR` zurückgeliefert.

```
void emo_freeMovie(emoType *emoHandle);
```

EMO stellt einen Speicher für die Bilder bereit, wenn die `emo_takeMovie` aufgerufen wird. Dieser Speicher sollte durch einen Aufruf von `emo_freeMovie` wieder freigegeben werden. Als Parameter wird lediglich das Handle `emoHandle` übergeben.

```
double *emo_getMoving(emoType *emoHandle, picEmoType *pic, double *z, int *dx,
int *dy, double *r11, double *r12, double *r13, double *r21, double *r22, double
*r23, double *r31, double *r32, double *r33, double *tx, double *ty, double *tz,
int *num);
```

Der Anwender kann über die Routine `emo_getMoving` abfragen, ob bewegte Objekte extrahiert wurden. Als Parameter wird das Handle `emoHandle` übergeben. Zudem muß der Anwender noch die Adressen einiger Variablen angeben. Folgende Variablen werden außer dem Handle benötigt:

`picEmoType *pic;`	Zeiger für das aktuelle Bild
`double *z;`	Zeiger für die geschätzte Entfernung
`int *dx, *dy;`	Zeiger für die Verschiebung des Bildes
`double *r11, *r12, *r13, *tx;`	Zeiger für die Transformationsmatrix
`double *r21, *r22, *r23, *ty;`	Dabei bezeichnen die r_{ij} die Rotationsmatrix
`double *r31, *r32, *r33, *tz;`	und t_j bezeichnet den Translationsvektor
`int *num;`	Zeiger für die Anzahl der bewegten Objekte

Die Routine schreibt die Werte in die vom Anwender angegebenen Adressen. Dem Anwender wird zudem noch ein Zeiger auf einen Integer-Wert zurückgeliefert. Ab diesem Wert sind die bewegten Objekte nacheinander abgelegt. Werden die bewegten Objekte nicht mehr benötigt, so muß der Anwender den allokatierten Speicher wieder zurückgeben.

Die bewegten Objekte werden von EMO intern als `VEdges` gespeichert. Da die Objekte in den Speicherbereich des aufrufenden Programms kopiert werden müssen, wird die Information aus `VEdges` geglättet und linear abgespeichert. Da die hier verwendete Struktur für allegemeine `VEdges` geeignet ist, werden auch Informationen gespeichert, die auf den ersten Blick nicht notwendig sind. Weil intern Hypothesen und bewegte Objekte mit der gleichen Struktur gespeichert werden, sind in den Objektinformationen auch Daten enthalten, die zur Validierung der Hypothesen herangezogen werden.

Zuerst folgt ein Kopf, im dem die folgenden Informationen gespeichert sind.

- `int number;` Anzahl der bewegten Objekte
- `int width;` Breite des Bildes, aus dem die Objekte extrahiert wurden.

- `int height`; Höhe des Bildes, aus dem die Objekte extrahiert wurden.
- `int field_num`; Anzahl der Integer-Werte der Objektinformationen.
- `int point_num`; Anzahl der Integer-Werte, die pro Punkt eines Polygons gespeichert werden.

Nach dem Header folgen der Reihe nach die `number` bewegten Objekte. Ein bewegtes Objekt besitzt ebenfalls einen Header. Dieser Header ist wie folgt abgelegt.

- `int n`; Anzahl der Punkte der konvexen Hülle des bewegten Objektes.
- `int closed`; Wert, der angibt, ob es sich bei den Punkten um eine geschlossene Kontur handelt. Dieser Wert ist immer auf `TRUE` gesetzt.

Nach diesen beiden Werten folgen die Integer-Werte der Objektinformation. Da ein bewegtes Objekt zunächst eine Hypothese gewesen ist, sind in den Objektinformationen auch Daten aus der Zeit als Hypothese gespeichert.

- `int sx`; X-Koordinate des Schwerpunktes
- `int sy`; Y-Koordinate des Schwerpunktes
- `int vx`; X-Komponente des Bewegungsvektors
- `int vy`; Y-Komponente des Bewegungsvektors
- `int timeToValidate`; Zähler, der angibt, für wieviel Iterationen des Algorithmuses ein bewegtes Objekt innerhalb der Fläche einer Bewegungshypothese liegen muß, bevor die Hypothese validiert wird.
- `int life`; Lebensdauer des Objektes in Iterationen.
- `int maxTrials`; Maximale Lebensdauer der Hypothese in Iterationen.
- `int area`; Fläche des bewegten Objektes
- `int motVecNum`; Anzahl der Bewegungsvektoren, die bei der Validierung der Hypothesen verwendet wurden, um den Bewegungsvektor des bewegten Objektes zu berechnen. Da verschiedene Hypothesen in einer Iteration gleichzeitig validiert werden können, kann es passieren, daß aus mehreren Hypothesen ein bewegtes Objekt entsteht.
- `int id`; Identitätsnummer des bewegten Objektes. Die Nummern werden für die Hypothesen vergeben. Wird eine Hypothese zu einem bewegten Objekt, so wird diese Nummer beibehalten.

Dies schließt den Kopf der Informationen des bewegten Objektes ab. Es folgen die Punkte der konvexen Hülle des bewegten Objektes.

- 2n Integerwerte, die die Koordinaten des Polygons angeben. Zuerst folgt die X-Koordinate, dann die Y-Koordinate. Nach den ersten beiden Koordinaten folgen die Koordinaten der restlichen Punkte.

Falls weitere bewegte Objekte in der Liste enthalten sind, folgt ein bewegtes Objekt nach dem anderen.

Das Bild, aus dem die bewegten Objekte extrahiert wurden, wird in die vom Anwender durch den Parameter `pic` angegebene Struktur geschrieben. Die Struktur hat das folgende Format.

```
struct picEmoStruct {
       imageEmoType  *image;
       int           allocated;
       int           width;
       int           height;
       int           maxGray;
       double        time;
       int           frame;
       char          model[EMO_MODEL_LEN];
       double        f;
       camAngEmoType camPos;
       camAngEmoType camVel;
       robVelEmoType robVel;
     };

typedef picEmoStruct picEmoType;
```

Die Pixelwerte des Bildes sind zeilenweise ab der Adresse auf die `image` zeigt, abgespeichert. Ist `allocated` auf `TRUE` gesetzt, so wurde der Speicher für die Pixelwerte des Bildes allokatiert. Die Elemente `width` und `height` geben die Größe des Bildes an. Das Element `maxGray` gibt den maximalen Grauwert an. Die Zeit, zu der das Bild gegrabbt wurde, ist in `time` (in Sekunden) gespeichert. Die Nummer des Bildes ist in `frame` enthalten. Das verwendete Kameramodell ist in dem String `model` angegeben. Die Brennweite der Kamera ist in `f` (in Pixeln) gespeichert. Die Position der Kameragelenke ist in `camPos`, die Geschwindigkeit der Kameragelenke ist in `camVel` und die Geschwindigkeit des Fahrzeugs ist in `robVel` gespeichert.

Für die Position und Geschwindigkeit der Kameragelenke wird die folgende Struktur verwendet.

```
struct camAngEmoStruct {
       double     theta1;
       double     theta2;
       double     theta3;
       double     theta4;
     };
typedef struct camAngEmoStruct camAngEmoType;
```

Die Position der Kameragelenke wird in Milligrad, die Geschwindigkeit in Milligrad pro Sekunde angegeben.

Die Geschwindigkeit des Fahrzeugs wird mit folgender Struktur angegeben.

```
struct robVelEmoStruct {
       double     v;
       double     w;
```

```
    };
typedef struct robVelEmoStruct robVelEmoType;
```

Die lineare Geschwindigkeit `v` wird in Millimeter, die Winkelgeschwindigkeit `w` in Millirad pro Sekunde angegeben.

```
void emo_exit(emoType *emoHandle);
```

Mit der Routine `emo_exit` beendet der Anwender das Modul EMO. Als Parameter wird lediglich das Handle `emoHandle` übergeben. EMO beendet dann die zuvor gestarteten Prozesse und gibt allokatierten Speicher wieder frei.

5.8 Benutzung des Programms

Neben der Bibliothek zur Extraktion bewegter Objekte wurde noch ein Programm entwickelt, mit dem die bewegten Objekte einer bereits aufgenommene Bildsequenz betrachtet werden kann. Die vom Programm EMO erzeugten Fenster können als Postscript-Dokument [Adobe 90] abgespeichert werden. Mit diesem Programm wurden die in der vorliegenden Arbeit enthaltenen Bilder erzeugt. Die Daten, zu den in der vorliegenden Arbeit enthaltenen Diagrammen, wurden ebenfalls mit diesem Programm abgespeichert. Das Programm hat den Namen `emo`. Dem Programm können eine Reihe von Parameter beim Aufruf des Programms übergeben werden. Die Parameter werden durch `-Xparameter` in beliebiger Reihenfolge spezifiziert. Dabei steht `X` für den Namen des Parameters und `parameter` gibt den String an, der als Parameter `X` übergeben wird. Die folgenden Parameter können spezifiziert werden.

- `F` Spezifiziert den Dateinamen des Bildes. An den Dateinamen fügt das Programm `show` die Erweiterung `N.pgm` an. Dabei steht `N` für die Nummer des Bildes. Die Bilder werden beginnend mit der Nummer 0 der Reihe nach eingelesen.
- `N` Gibt die Anzahl der Bilder an, die maximal eingelesen werden. Ist dieser Parameter nicht spezifiziert, so werden alle Bilder beginnend bei 0 eingelesen. Höchstens jedoch 30 Bilder.
- `E` Gibt die Anzahl der Iterationen an, nach denen die Bearbeitung der Bildsequenz abgebrochen wird. Werden mehr Iterationen spezifiziert als Bilder eingelesen wurden, so wird nach dem letzten Bild der Sequenz wieder mit dem ersten Bild der Sequenz begonnen.
- `Z` Gibt die geschätzte Entfernung zum Hintergrund an, die für das erste Bild der Sequenz verwendet wird. Die Entfernung wird in Metern angegeben.
- `S` Gibt den Skalierungsfaktor an, mit dem die Bilder der Sequenz verkleinert werden können. Der Skalierungsfaktor wird im Bereich von 0 bis 1 spezifiziert und gibt die Bildgröße in Prozent an, auf denen gearbeitet wird.
- `V` Gibt die Art der Ausgabe des Algorithmuses an. Als Parameter können hier die Werte `0`, `1` und `2` übergeben werden. Dabei bedeutet der Wert `0`, daß keine Ausgabefenster geöffnet werden. Der Wert `1` gibt an, daß lediglich ein Fenster mit den bewegten Objekten der Sequenz angezeigt wird. Der Wert `2` gibt an, daß alle Fenster geöffnet werden.

- `W` Gibt die Größe des Fensters an, in dem die Bildsequenz angezeigt wird. Die Größe wird in der Form `w,h` angegeben. Dabei steht `w` für die Breite des Bildes und `h` steht für die Höhe des Bildes.

- `G` Gibt den sogenannten Grab-String an. Durch die Angabe dieses Parameters kann der Inhalt einzelner Fenster als Postscript-Dokument abgespeichert werden. Doch dazu müssen die entspechenden Fenster geöffnet sein. Der String hat das folgende Format:

  `N-M,filename,C,1,2,...,11`

  Der Parameter `N` gibt die Iteration an, ab der die Inhalte der Fenster gespeichert werden sollen. Der Parameter `N` gibt die Iteration an, bis zu der die Inhalte der Fenster gespeichert werden sollen. Falls der Parameter `N` Null ist, kann statt `0-M` auch nur `M` angegeben werden. Der Parameter `filename` gibt den Dateinamen an, unter dem die Postscript-Dokumente gespeichert werden. An den Dateinamen wird noch die Erweiterung `WXtY.ps` angehängt. Dabei steht `X` für die Nummer des Fensters und `Y` steht für die Nummer der Iteration. Nach dem Dateinamen `filename` wird durch den Parameter `C` angegeben, ob eine Bildunterschrift mit den Bild-, Kamera- und Fahrzeugparametern gewünscht wird. Dabei gibt `Y` an, daß eine Bildunterschrift generiert wird und `N` gibt an, daß keine Bildunterschrift generiert wird. Nach dem Parameter `C` können eine beliebige Anzahl an Nummern, durch Komma getrennt, folgen. Die angegebenen Nummern geben die Nummern der Fenster an, deren Inhalt abgespeichert werden soll.

- `D` Gibt den String für die Speicherung der geschätzten Entfernungen zum Hintergrund an. Der String hat das folgende Format:

  `N,filename`

  Dabei gibt der Parameter `N` die Anzahl der Iterationen an, für die die geschätzten Entfernungen zum Hintergrund gespeichert werden. Der Dateiname wird durch den Parameter `filename` spezifiziert. An den Dateinamen wird noch die Erweiterung `z.data` angehängt. Die Entfernungen werden in der Datei zeilenweise im ASCII-Format gespeichert.

- `Y` Gibt den String für die Speicherung der vertikalen Verschiebung des Bildes, mit der die Prädiktion verbessert wird, an. Der String hat das folgende Format:

  `N,filename`

  Dabei gibt der Parameter `N` die Anzahl der Iterationen an, für die die vertikalen Verschiebungen gespeichert werden. Der Dateiname wird durch den Parameter `filename` spezifiziert. An den Dateinamen wird noch die Erweiterung `dy.data` angehängt. Die vertikalen Verschiebungen werden in der Datei zeilenweise im ASCII-Format gespeichert.

- `T` Gibt den String für die Speicherung der Zeitdauer an, die für eine Iteration zur Extraktion bewegter Objekte benötigt wird. Der String hat das folgende Format:

  `N,filename`

  Dabei gibt der Parameter `N` die Anzahl der Iterationen an, für die die Zeitdauer gespeichert wird. Der Dateiname wird durch den Parameter `filename` spezifiziert. An den

Dateinamen wird noch die Erweiterung `t.data` angehängt. Die Zeitdauern werden in der Datei zeilenweise im ASCII-Format gespeichert. Die Zeit einer Iteration enthält nur die Zeit, die zur Verarbeitung eines Bildes von EMO benötigt wird. Die Zeiten, die zum Grabben des Bildes und zur Bestimmung der Statusinformationen benötigt werden, sind in der gespeicherten Zeitdauer nicht enthalten.

Ein Aufruf des Programms könnte zum Beispiel wie folgt aussehen:

```
emo -Fbild -N20 -E20 -Z2.0 -S0.75 -V2 -W192,144 -T20,time
      -G12-17,fenster,Y,1,2,4,5 -D20,entfernung -Y20,verschiebung
```

Dieser Aufruf liest der Reihe nach die Bilder `bild0.pgm` bis maximal Bild `bild20.pgm` ein. Das Programm wird nach 20 Iterationen, also nach dem 20. Bild abgebrochen. Die geschätzte Entfernung wird auf 2.0 gesetzt. Die Bilder werden auf 75% verkleinert, bevor sie bearbeitet werden. Es werden alle Ausgabefenster geöffnet. Die Fenstergröße beträgt 192×144. Die Zeitmessungen werden für 20 Iterationen in der Datei `time` gespeichert. Die Ausgabefenster mit der Nummer 1,2,4 und 5 werden mit Bildunterschrift für die Iterationen 12 bis 17 mit dem Dateinamen `fenster` gespeichert. Für 20 Iterationen werden die geschätzten Entfernungen zum Hintergrund unter dem Dateinamen `entfernung` gespeichert. Die vertikale Verschiebung wird für 20 Iterationen unter dem Dateinamen `verschiebung` gespeichert.

5.9 Graphische Ausgabe des Programms EMO

Das Programm EMO öffnet 11 Fenster um die einzelnen Stufen der Bildverarbeitung zu visualisieren. Die Fenster werden im folgenden näher beschrieben.

- Fenster 1

  In Fenster 1 wird das vorhergehende Bild $B(t_1)$ und die zugehörigen bewegten Objekte $BO(t_1)$ in weiß angezeigt.

- Fenster 2

  In Fenster 2 wird das aktuelle Bild $B(t_1)$ und die markanten Punkte $\check{B}(t_1)$ des aktuellen Bildes angezeigt. Die markanten Punkte sind durch gelbe Quadrate dargestellt.

- Fenster 3

  In Fenster 3 wird die Prädiktion $B_T(t_1)$ des aktuellen Bildes angezeigt. Zudem sind die markanten Punkte des Bildes $\check{B}_T(t_1)$ durch gelbe Quadrate eingezeichnet. Die Prädiktion $BO_T(t_2)$ der bewegten Objekte zum Zeitpunkt des aktuellen Bildes sind rot dargestellt. Für markante Punkte, die nicht innerhalb eines rot eingezeichneten bewegten Objektes liegen, wird versucht eine Korrespondenz zu einem markanten Punkt aus $B(t_2)$ herzustellen. Kann eine Korrespondenz zweier markanter Punkte aus Prädiktion und aktuellem Bild hergestellt werden, so wird der Verschiebungsvektor des markanten Punktes ebenfalls in Fenster 3 eingezeichnet. Die resultierende Verschiebung $(\Delta x, \Delta y)$ wird im Zentrum des Fensters in einem grünen Kreis als grüner Vektor visualisiert.

- Fenster 4

  In Fenster 4 wird das Differenzbild aus der Prädiktion $B_T(t_1)$ und des aktuellen Bildes $B(t_2)$ dargestellt.

- Fenster 5

  In Fenster 5 wird das Differenzbild aus der durch den Verschiebungsvektor $(\Delta x, \Delta y)$ korrigierten Prädiktion $B_{T_\Delta}(t_1)$ und des aktuellen Bildes $B(t_2)$ dargestellt.

- Fenster 6

  In Fenster 6 wird das Absolutbild des Differenzbildes von Fenster 5 angezeigt.

- Fenster 7

  Das Absolutbild des Differenzbildes wird durch die Anwendung morphologischer Operatoren bearbeitet und binarisiert. In Fenster 7 wird das binarisierte Bild angezeigt. Die äußeren Konturen K der Regionen sind rot eingezeichnet. Zwischen den beiden nächsten Punkten zweier Konturen wird eine Linie eingezeichnet, die in beide Richtungen um den Abstand der beiden Punkte verlängert wird. Liegen die beiden Endpunkte der Linie innerhalb der korrespondierenden Regionen, so wird die Linie rot eingezeichnet, andernfalls blau. Wird eine Linie rot gezeichnet, so werden die beiden zugehörigen Konturen zusammengefaßt. Schließlich wird die konvexe Hülle der zusammengefaßten Regionen gebildet. Die resultierenden Regionen $K/\sim$, sind gelb visualisiert.

- Fenster 8

  In Fenster 8 ist nochmals das vorhergehende Bild $B(t_1)$ angezeigt. In dem Bild sind die bewegten Regionen $BR(t_1)$ grün eingezeichnet.

- Fenster 9

  In Fenster 9 ist nochmals das aktuelle Bild $B(t_2)$ angezeigt. In dem Bild sind die bewegten Objekte $BO_{T_\Delta}(t_2)$ gelb eingezeichnet. Die neuen bewegten Regionen $BR(t_2)$ sind grün dargestellt. Regionen für die eine Bewegungshypothese aufgestellt werden kann sind blau visualisiert.

- Fenster 10

  In Fenster 10 ist nochmals das aktuelle Bild $B(t_2)$ angezeigt. In dem Bild sind die neuen Bewegungshypothesen $BH_(t_2)$ weiß eingezeichnet.

- Fenster 11

  In Fenster 11 ist nochmals das aktuelle Bild $B(t_2)$ angezeigt. In dem Bild sind die neuen bewegten Objekte $BO(t_2)$ weiß eingezeichnet.

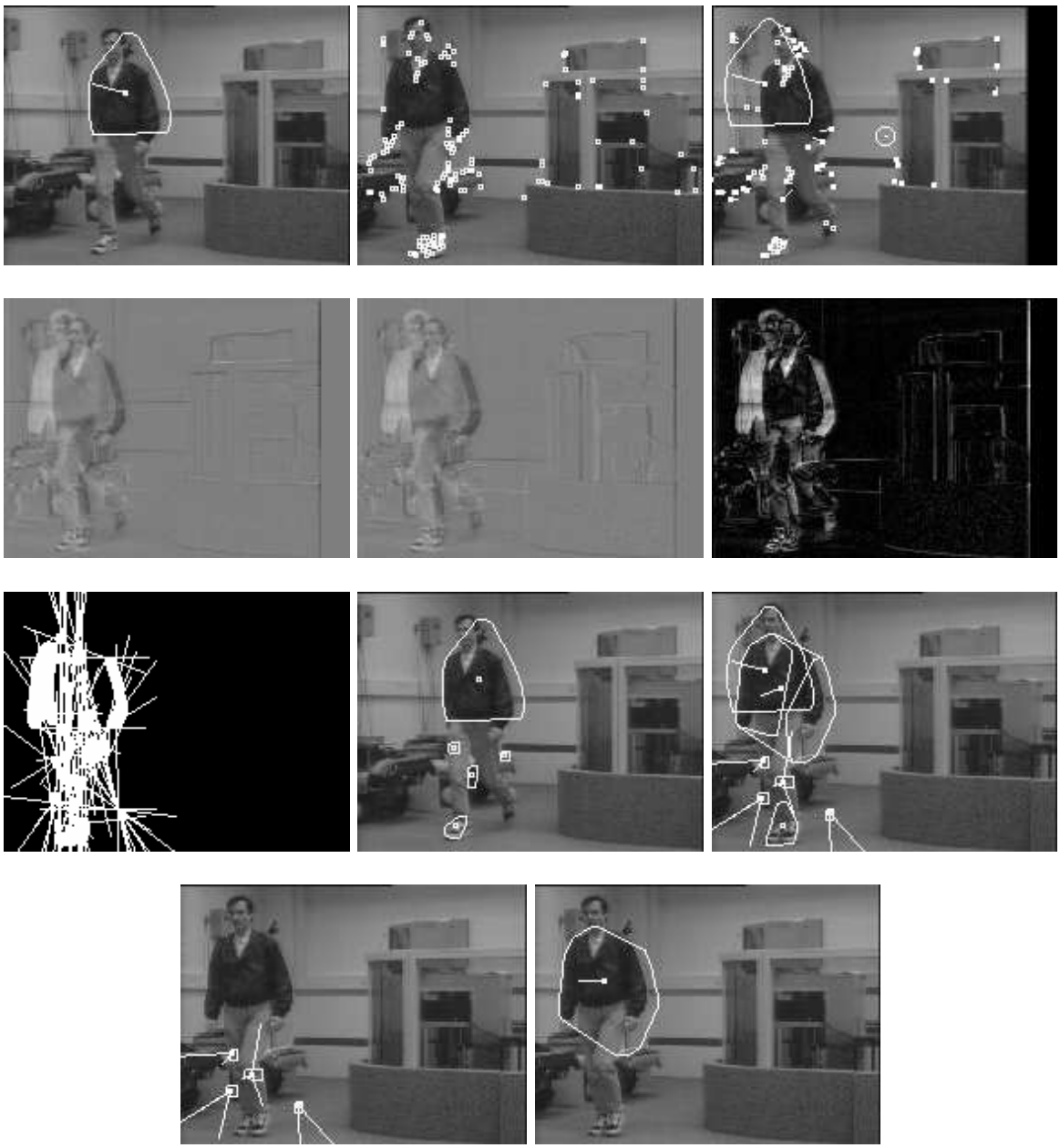

Abbildung 5.7: Von EMO werden 11 Fenster geöffnet. In den Fenstern werden die einzelnen Stufen des Algorithmuses visualisiert.

Kapitel 6

Experimente und Ergebnisse

Im folgenden werden die mit dem Programm EMO durchgeführten Experimente beschrieben. Unter den in diesem Kapitel gezeigten Bildern sind die relevanten Parameter dargestellt.

- `t`: Zeitpunkt t der Aufnahme.
- `z`: Geschätzte Entferung $\tilde{Z}$ des Hintergrundes zur Kamera.
- `v`: Lineare Geschwindigkeit des Fahrzeuges v.
- `w`: Winkelgeschwindigkeit des Fahrzeuges w
- `theta1`: Position von Gelenk 1.
- `theta2`: Position von Gelenk 2.
- `theta3`: Position von Gelenk 3.
- `dottheta1`: Winkelgeschwindigkeit von Gelenk 1.
- `dottheta2`: Winkelgeschwindigkeit von Gelenk 2.
- `dottheta3`: Winkelgeschwindigkeit von Gelenk 3.

6.1 Schwingungen der Kamera

Um zu testen, wie stark die Kamera schwingt, wurde folgendes Experiment durchgeführt. Das Fahrzeug fuhr mit einer linearen Geschwindigkeit von $0.5\frac{m}{s}$ geradeaus, wobei die Kamera auf eine Wand an der linken Seite des Fahrzeuges gerichtet war. Die Wand befand sich im Abstand von 2.45m zur Bildebene. Die Entfernung zur Wand wurde mittels Kalibrierungsblätter, die an der Wand angebracht wurden, berechnet. Auf den Kalibrierungsblättern befanden sich $5 \times 5cm^2$ große schwarze Bereiche auf einem weißen Hintergrund.

Die Kamera bewegte sich parallel zur Wand nach rechts. Daher bewegten sich die Bildpunkte von einem Bild zum nächsten nach links. Die Transformation entsprechend Gleichung 4.9 und Gleichung 4.10 verschiebt das Bild aufgrund der vom Fahrzeug gelieferten Statusinformationen nach links. Aufgrund von Schwingungen der Kamera bewegt sich die Kamera aber auch noch in vertikaler Richtung. Die Schwingungen werden zum Beispiel durch Unebenheiten

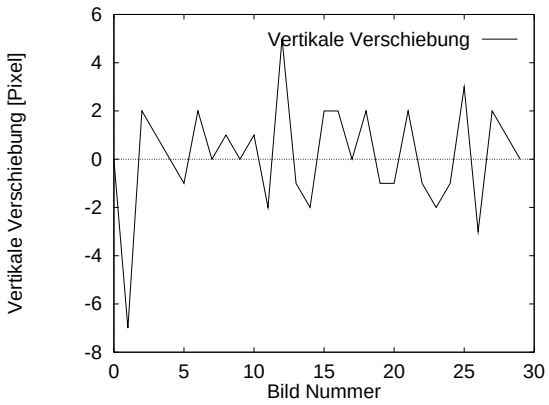

Abbildung 6.1: Vertikale Verschiebung der Prädiktion des Kamerabildes zum Ausgleich vertikaler Schwingungen. Die aus der Statusinformationen des Fahrzeuges und der Kamera berechnete Transformation verschiebt das Bild nur horizontal und nicht vertikal.

im Boden verursacht. Die vom Fahrzeug gelieferten Statusinformationen reichen nicht aus, um die Schwingungen der Kamera zu kompensieren.

Um die Schwingungen zu kompensieren, verschiebt das Programm EMO das Bild in horizontaler und vertikaler Richtung. So wird die Prädiktion der Bildbewegung verbessert. Die vertikale Verschiebung, die das Programm EMO zu Verbesserung der Prädiktion durchgeführt hat, sind in Abbildung 6.1 dargestellt. Die Größte vom Programm EMO durchgeführte Verschiebung des Bildes betrug 7 Pixel. Dies zeigt, daß auf eine Fehlerkorrektur zur Verbesserung der Prädiktion nicht verzichtet werden kann.

6.2 Genauigkeit der Entfernungsberechnung

Um zu überprüfen, wie genau der Algorithmus die Entfernung zum Hintergrund abschätzt, wurde die gleiche Bildsequenz wie die zur Bestimmung der vertikalen Kameraschwingungen eingesetzt. Zudem wurde eine zweite Sequenz aufgenommen, bei der sich ein bewegtes Objekt im Bild der Kamera befand. Diese Sequenz wird hier kurz beschrieben.

Das Fahrzeug fuhr ungefähr in konstanter Entfernung an einer Wand des Labors entlang. Die Kamera war dabei auf die Wand auf der linken Seite des Fahrzeuges gerichtet. Im Blickfeld der Kamera befand sich nur die Wand. Bei einer Vorwärtsbewegung des Fahrzeuges bewegte sich der Hintergrund im Blickfeld der Kamera nach links. Das Fahrzeug fuhr mit einer Geschwindigkeit von $0.5\frac{m}{s}$. An der Wand waren Kalibrierungsblätter angebracht. Eine Person bewegte sich im Blickfeld der Kamera mit annähernd der gleichen Geschwindigkeit wie das Fahrzeug. Das bewegte Objekt, das vom Programm EMO extrahiert wurde, ist in Abbildung 6.2 und Abbildung 6.3 dargestellt.

In Abbildung 6.4 sind die geschätzten Entfernungen für die beiden Videosequenzen dargestellt. Im linken Schaubild sind die geschätzten Entfernungen ohne bewegtes Objekt und im rechten die mit einem bewegten Objekt im Sichtbereich der Kamera dargestellt. Die tatsächliche Entfernung zum Hintergrund ist in beiden Schaubildern aufgetragen. Sie wurde aus der Größe der Kalibrierungsblöcke im Bild berechnet. Die geschätzte Entfernung für

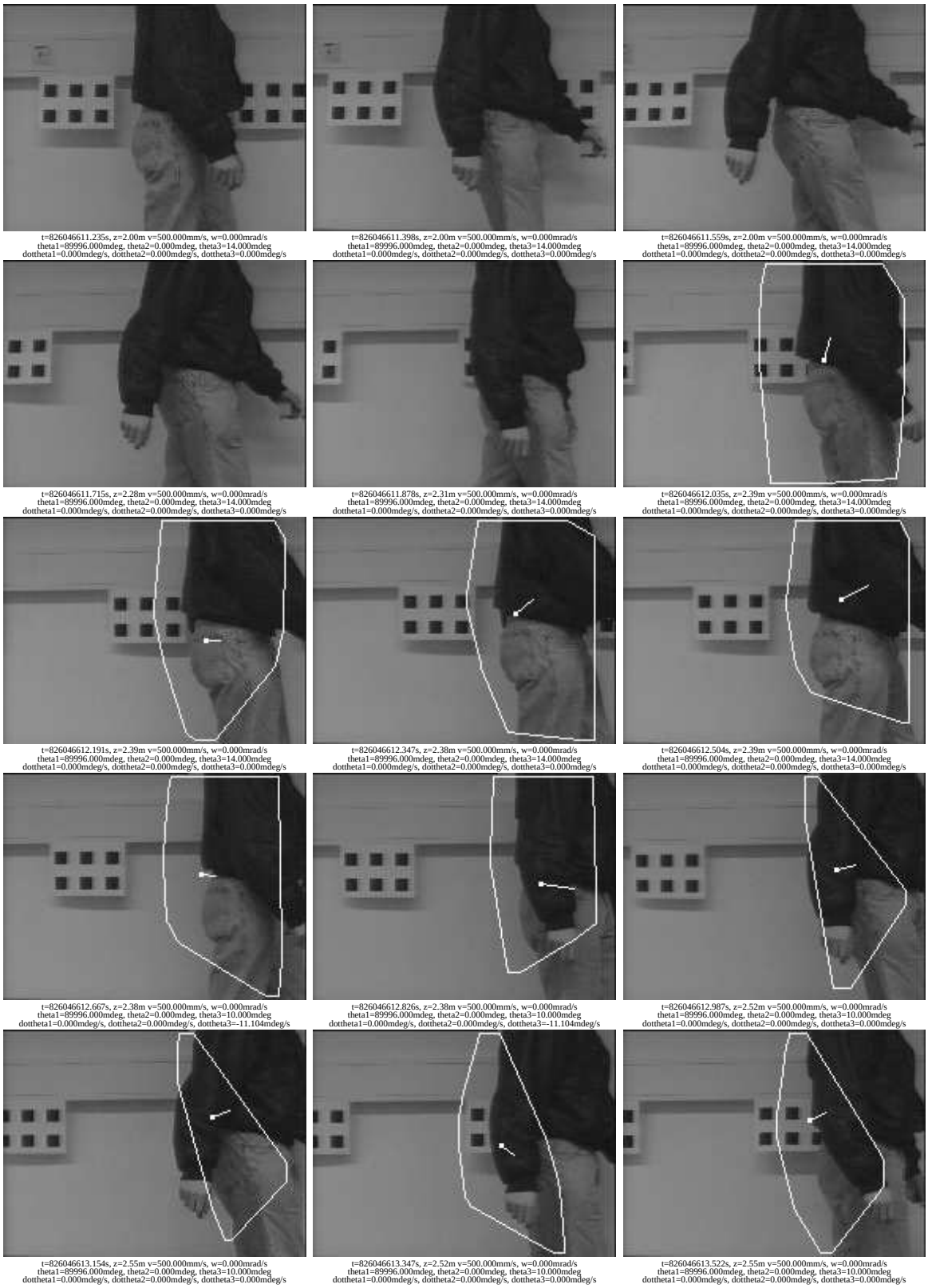

Abbildung 6.2: Bilder 0 bis 14 einer Sequenz (Sequenz 2 der Appendix), bei der sich die Kamera translatorisch entlang einer Wand bewegt.

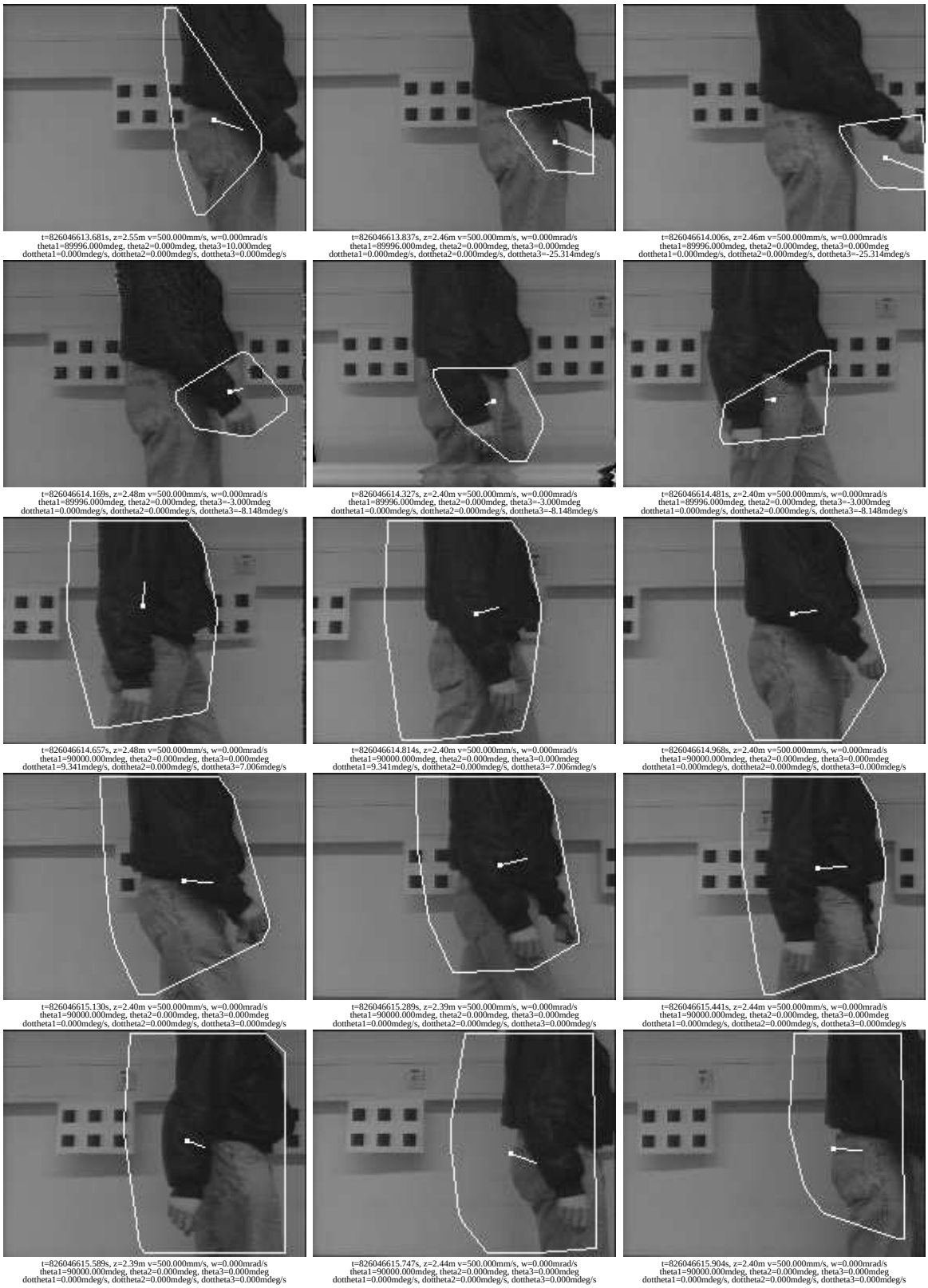

Abbildung 6.3: Bilder 15 bis 29 einer Sequenz (Sequenz 2 der Appendix), bei der sich die Kamera translatorisch entlang einer Wand in konstantem Abstand bewegt.

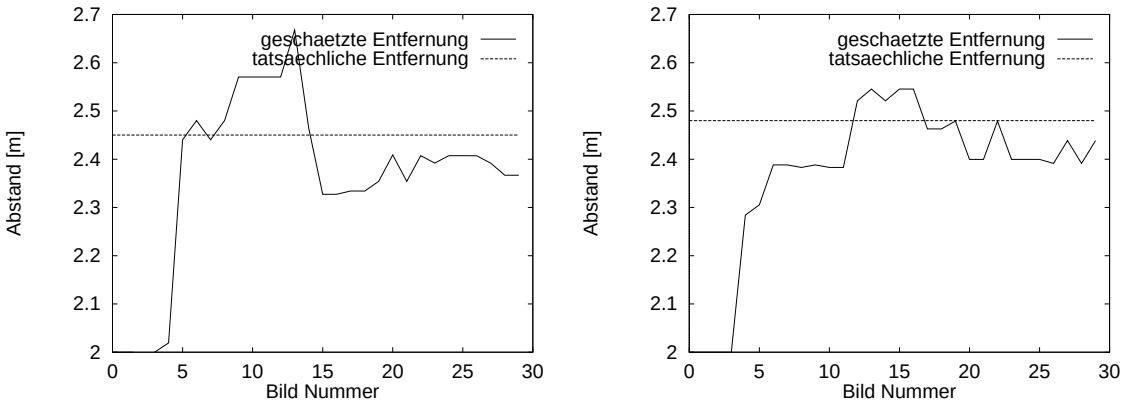

Abbildung 6.4: Beide Schaubilder zeigen die geschätzte und die tatsächliche Entfernung der Kamera zum Hintergrund. Die Kamera war auf eine Wand gerichtet, an der sich Kalibrierungsblätter befanden. Aus der Größe dieser Blätter wurde die tatsächliche Entfernung berechnet. Die Kamera bewegte sich beide Male translatorisch nach rechts mit $0.5\frac{m}{s}$. In der Sequenz des rechten Schaubildes befand sich ein bewegtes Objekt im Sichtbereich der Kamera.

das erste Bild der beiden Sequenzen wurde auf 2m gesetzt. Ohne bewegtes Objekt werden fünf Bilder benötigt, um ab dem sechsten Bild die Entfernung mit einer Genaugikeit von 8,9% abzuschätzen. Bei der Sequenz mit bewegtem Objekt werden fünf Bilder benötigt, um ab dem sechsten Bild die Entfernung mit einer Genauigkeit von 5,9% abzuschätzen.

6.3 Schnelligkeit der Entfernungsanpassung

EMO verwendet die aus dem Bild berechnete Entfernung nicht sofort, sondern trägt sie in eine Liste mit Abständen ein, die in vorhergehenden Iterationen berechnet wurden. Der Median dieser Liste ist dann die geschätzte Entfernung zum Hintergrund. Falsche Entfernungen sollen somit eliminiert werden. Denn bei der Berechnung der Entfernung geht der Algorithmus davon aus, daß die Daten über die Bewegung der Kamera korrekt sind. Dies trifft aber nicht immer zu. EMO setzte für die Medianbildung eine Liste mit 5 Werten ein. Somit sind mindestens 3 Werte nötig, bis ein neuer Wert verwandt wird. Bei einem Schwenk der Kamera von einer entfernten auf eine nahe Szene (oder umgekehrt) sind also mindestens 3 Werte nötig.

Um die Schnelligkeit der Entfernungsanpassung zu prüfen wurden noch zwei weitere Sequenzen aufgenommen, bei denen ein plötzlicher Tiefenunterschied auftrat. Das Fahrzeug fuhr wie in Abschnitt 6.2 an einer Wand entlang. Durch eine künstliche Wand, die sich im Abstand von etwa 1.50m von der echten Wand befand, wurde eine starke Tiefendifferenz erzeugt. Die vom Programm EMO extrahierten bewegten Objekte dieser Sequenz sind in Abbildung 6.5 und Abbildung 6.6 gezeigt.

Abbildung 6.7 zeigt die vom Algorithmus geschätzten Entfernungen. Die geschätzte Entfernung für das erste Bild der beiden Sequenzen wurde auf 2m gesetzt. Das Bild, in dem die Wand in den Sichtbereich der Kamera kommt, ist durch einen senkrechten Balken markiert. Das Bild, in dem die künstliche Wand den linken Bildrand erreicht hat, ist durch den zweiten senkrechten Balken markiert. Die Einschwingphase, bis die Entfernung zur Wand abgeschätzt

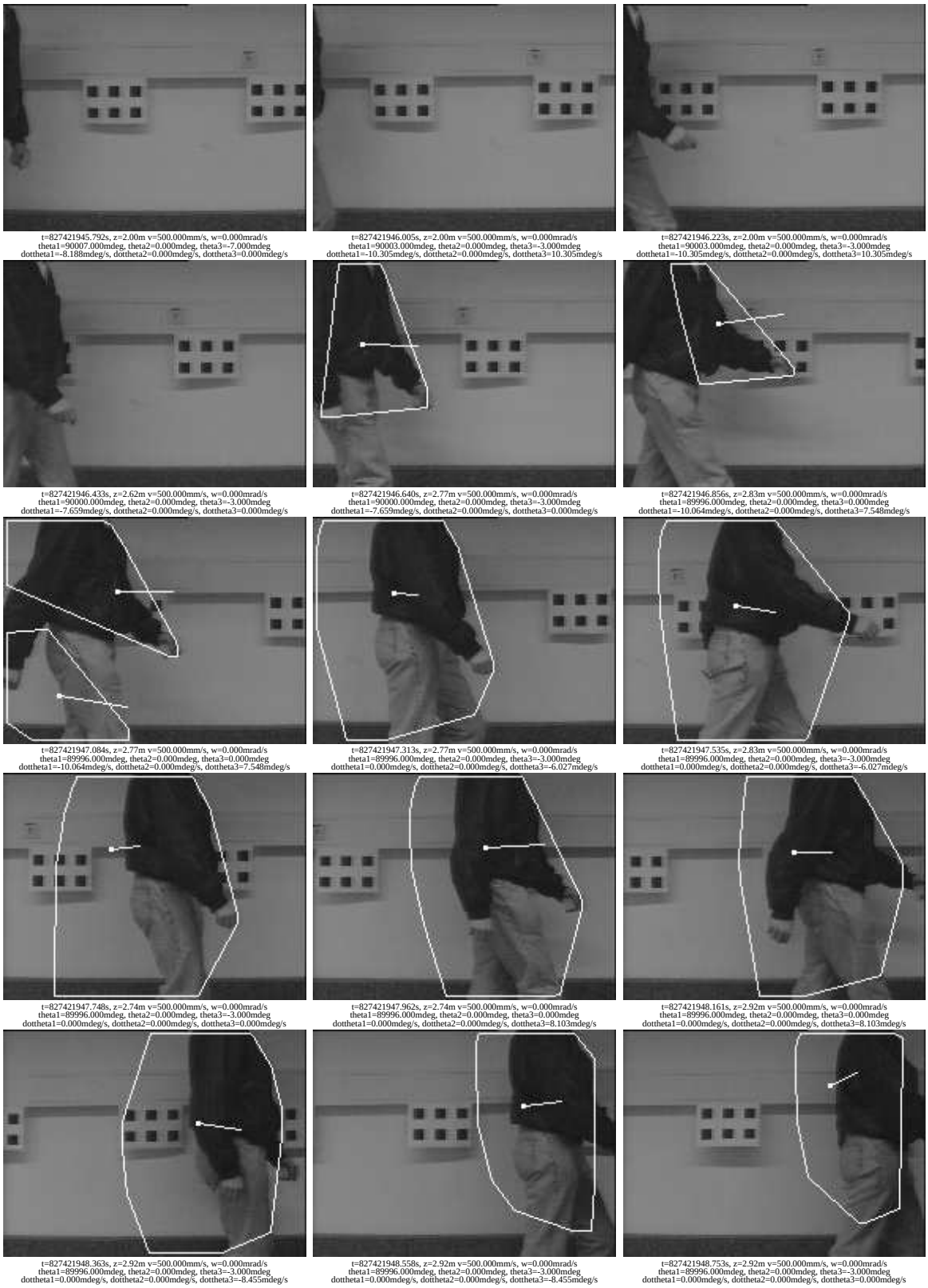

Abbildung 6.5: Bilder 0 bis 14 einer Sequenz (Sequenz 3 der Appendix), bei der sich die Kamera translatorisch entlang einer Wand mit Tiefendifferenz bewegt.

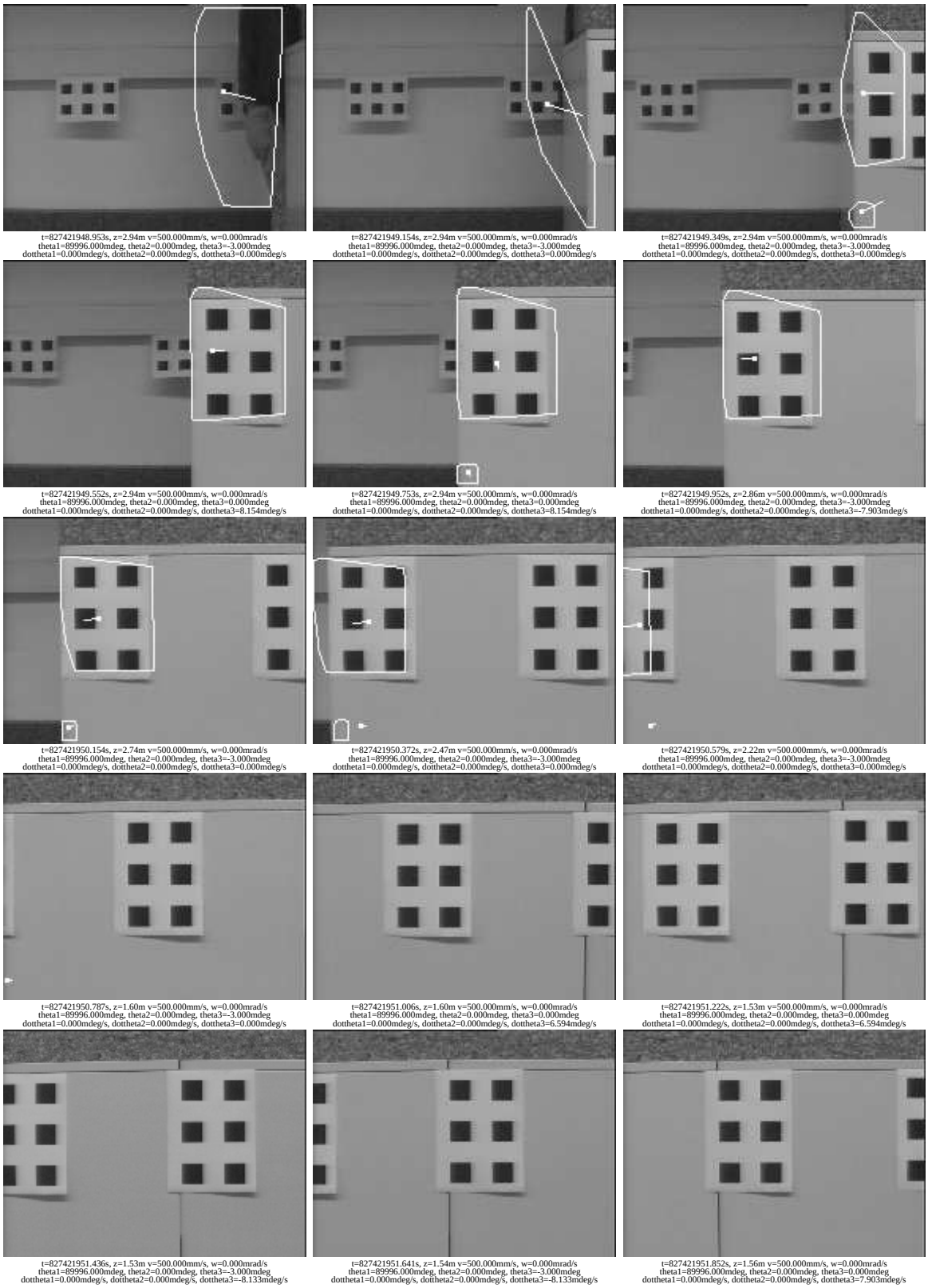

Abbildung 6.6: Bilder 15 bis 29 einer Sequenz (Sequenz 3 der Appendix), bei der sich die Kamera translatorisch entlang einer Wand mit Tiefendifferenz bewegt.

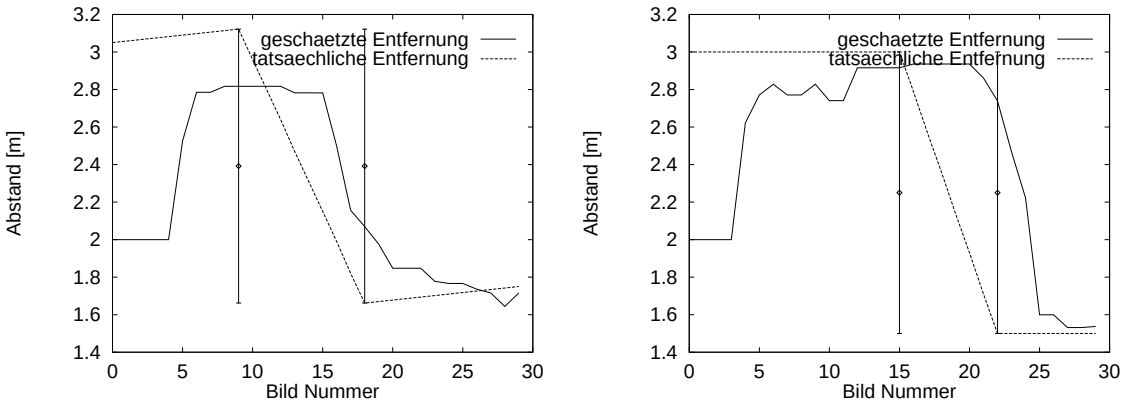

Abbildung 6.7: Beide Schaubilder zeigen die geschätzte und die tatsächliche Entfernung der Kamera zum Hintergrund. Die Kamera war auf eine Wand gerichtet, an der sich Kalibrierungsblätter befanden. Aus der Größe dieser Blätter wurde die tatsächliche Entfernung berechnet. Die Kamera bewegte sich beide Male translatorisch nach rechts mit $0.5\frac{m}{s}$. Der erste senkrechte Balken markiert das Bild, in dem eine künstliche Wand in den Sichtbereich der Kamera kommt. Der zweite Balken markiert das Bild, in dem die künstliche Wand den linken Bildrand erreicht hat. In der Sequenz des rechten Schaubildes befand sich ein bewegtes Objekt im Sichtbereich der Kamera.

wurde, beträgt für die Sequenz ohne bewegtes Objekt sechs Bilder und fünf Bilder bei der Sequenz mit bewegtem Objekt. Nachdem die künstliche Wand in den Sichtbereich der Kamera kommt, ist das Programm EMO bei beiden Videosequenzen in der Lage, die neue Entfernung zum Hintergrund mit abzuschätzen. Wichtig ist hier die Geschwindigkeit (gemessen in Anzahl der Bilder), die der Algorithmus benötigt, um sich an die veränderte Entfernung zum Hintergrund anzupassen. Ohne bewegtes Objekt wird die Entfernung mit einer Zeitverschiebung von etwa vier Bildern angepaßt. In der Sequenz mit bewegtem Objekt beträgt die Zeitverschiebung etwa fünf Bilder.

Problematisch wird es allerdings, wenn der Algorithmus direkt auf eine Wand zufährt. Dann nämlich muß die Entfernung kontinuierlich angepaßt werden. Ist dem Algorithmus die korrekte Entfernung zum Hintergrund bekannt, dann wird die Entfernung aufgrund der Eigenbewegung des Fahrzeuges angepaßt. Um zu prüfen, ob die Entfernung in diesem Fall korrekt abgeschätzt wird, wurde eine weitere Sequenz aufgenommen. In dieser Sequenz fährt das Fahrzeug direkt auf eine Wand zu. Es befinden sich keine bewegten Objekte in der Sequenz. An der Wand sind wieder Kalibrierungsblätter angebracht, mit deren Hilfe die tatsächliche Entfernung von der Kamera zum Hintergrund berechnet werden kann. Die Wand füllt den Sichtbereich der Kamera vollständig aus.

In Abbildung 6.8 ist das letzte Bild der Sequenz gezeigt. Das Programm EMO extrahierte im letzten Bild fälschlicherweise ein Teil des linken Kalibrierungsblattes als bewegtes Objekt. Zu diesem Zeitpunkt betrug die tatsächliche Entfernung zum Hintergrund 1.89m. Das Bild expandierte aufgrund der geringen Entfernung sehr stark. EMO war dann nicht mehr in der Lage, korrekt die Eigenbewegung zu kompensieren.

In Abbildung 6.9 sind die geschätzten und die tatsächlichen Entfernungen zum Hintergrund gezeigt. Das linke Schaubild zeigt die Entferungen für die eben beschriebene Sequenz. In einem

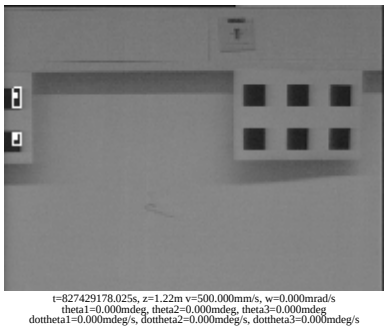

Abbildung 6.8: Letztes Bild einer Sequenz, bei der das Fahrzeug mit $0.5\frac{m}{s}$ auf eine Wand mit Kalibrierungsblättern zugefahren ist. Ein Teil des linken Kalibrierungsblattes wird von EMO fälschlicherweise als bewegtes Objekt extrahiert. Beim letzten Bild der Sequenz war der Algorithmus nicht mehr in der Lage, die Eigenbewegung der Kamera korrekt zu kompensieren.

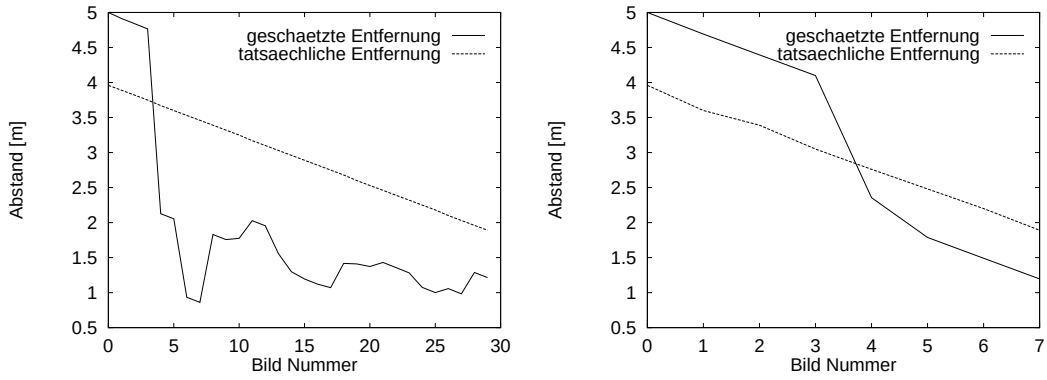

Abbildung 6.9: Beide Schaubilder zeigen die geschätzte und die tatsächliche Entfernung der Kamera zum Hintergrund. Die Kamera war auf eine Wand gerichtet, an der sich Kalibrierungsblätter befanden. Aus der Größe dieser Blätter wurde die tatsächliche Entfernung berechnet. Die Kamera bewegte sich beide Male auf die Wand mit $0.5\frac{m}{s}$ zu. In der Sequenz des rechten Schaubildes wurde dem Algorithmus nur jedes vierte Bild aus der Sequenz des linken Schaubildes gegeben.

weiteren Versuch wurde dem Algorithmus nur jedes vierte Bild aus der Sequenz zur Verfügung gestellt. In diesem Fall ist die Expansion von einem Bild zum nächsten deutlich größer. Die geschätzten Entfernungen sind im rechten Schaubild gezeigt. Aufgrund der größeren Bewegung des Fahrzeuges von einem Bild zum nächsten in der Bildsequenz des Schaubildes auf der rechten Seite von Abbildung 6.9 kann die Entfernung in diesem Fall genauer bestimmt werden, als bei der Bildsequenz des linken Schaubildes. Bei beiden Sequenzen wurde die Entfernung zum Hintergrund für das erste Bild auf 5m gesetzt.

6.4 Einfluß der Perspektive

Bestehen in einer Szene große Tiefenunterschiede der betrachteten Objekte, dann kann die Szene nicht durch einen in konstanter Entfernung von der Kamera befindlichen Hintergrund approximiert werden. Die Prädiktion der einzelnen Punkte des Hintergrundes ist dann umso schlechter, je stärker die tatsächliche Entfernung der Punkte zur Kamera von der geschätzten Entfernung abweicht.

Es wurde eine Bildsequenz aufgenommen, in der große Tiefenunterschiede vorhanden waren. Das Fahrzeug fuhr an einer etwa 8m entfernten Wand mit $0.5\frac{m}{s}$ entlang. Die Kamera wurde dabei auf die Wand gerichtet. Auf dem Boden des Labors, der sich im Sichtbereich der Kamera befand, lagen einige Schaumstoffteile. In der Videosequenz ist ein bewegtes Objekt zu sehen. Dabei handelt es sich um eine Person, die sich an der Wand entlang mit etwa der gleichen Geschwindigkeit wie das Fahrzeug bewegt.

Die von EMO extrahierten bewegten Objekte sind in Abbildung 6.10 visualisiert. EMO extrahiert die Person korrekt als bewegtes Objekt. Allerdings extrahiert der Algorithmus auch ein Schaumstoffteil, das sich parallel zur Blickrichtung auf dem Boden befindet. Das Schaumstoffteil wird extrahiert, weil die Prädiktion eine geschätzte Entfernung zum Hintergrund von 3m im ersten Bild und 5.58m im letzten Bild annimmt. Das Schaumstoffteil befindet sich, im Vergleich zur geschätzten Entfernung zum Hintergrund, nahe an der Kamera. Daher ist das Programm EMO nicht in der Lage, die Bewegung des Schaumstoffteils vorherzusagen. Durch eine andere Modellierung des Hintergrundes könnte der Einfluß der Perspektive verringert werden.

Auch andere Verfahren, bewegte Objekte zu extrahieren, werden durch starke Tiefenunterschiede beeinflußt. Smith et al. [Smith et al. 95] extrahieren bewegte Objekte, indem sie den optischen Fluß segmentieren. In [Smith et al. 95] extrahieren sie einen Teil der Straße nach dem Segmentierungsschritt. Dieser Teil wird aber später herausgefiltert. Um solche Objekte herauszufiltern, verwenden Smith et al. die Form und das Bewegungsmodell des Objektes.

6.5 Einfluß großer Bewegungen der Kamera

Der Einfluß großer Kamerabewegungen wurde ebenfalls anhand einer Videosequenz getestet. Das Fahrzeug bewegte sich mit $0.5\frac{m}{s}$ vorwärts und drehte sich dabei mit $0.25\frac{mrad}{s}$ nach rechts. Es fuhr also im Kreis herum. Die Kamera schwenkte zusätzlich noch von links nach rechts mit einer Winkelgeschwindigkeit von $21\frac{^\circ}{s}$. Der gemeinsam betrachtete Bereich zweier aufeinanderfolgender Bilder machte dabei nur noch etwa die halbe Bildbreite aus.

Die von EMO extrahierten Objekte sind in Abbildung 6.11 zu sehen. Das Differenzbild, das entstanden ist, nachdem die korrektive Verschiebung durchgeführt wurde, ist ebenfalls in Abbildung 6.11 zu sehen. Wie in Abbildung 6.11 zu sehen ist, gelang es dem Algorithmus sogar

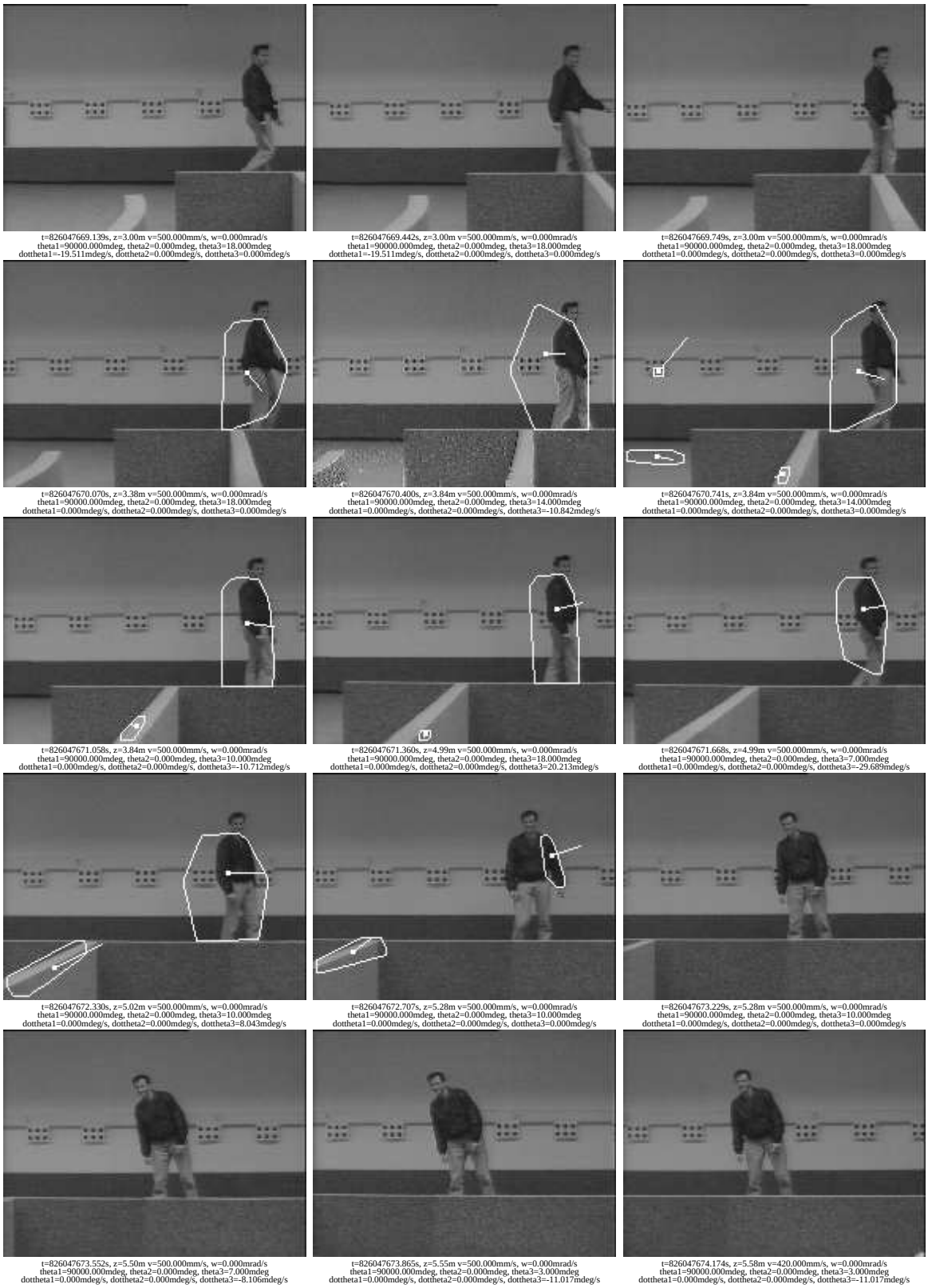

Abbildung 6.10: Extrahierte bewegte Objekte aus einer Videosequenz mit großen Tiefenunterschieden. Die Kamera bewegt sich translatorisch mit einer Geschwindigkeit von $0.5\frac{m}{s}$ nach rechts. Es wird die Person korrekt als bewegtes Objekt extrahiert. Allerdings wird auch ein Schaumstoffteil, das sich auf dem Boden befindet, fälschlicherweise als bewegtes Objekt extrahiert.

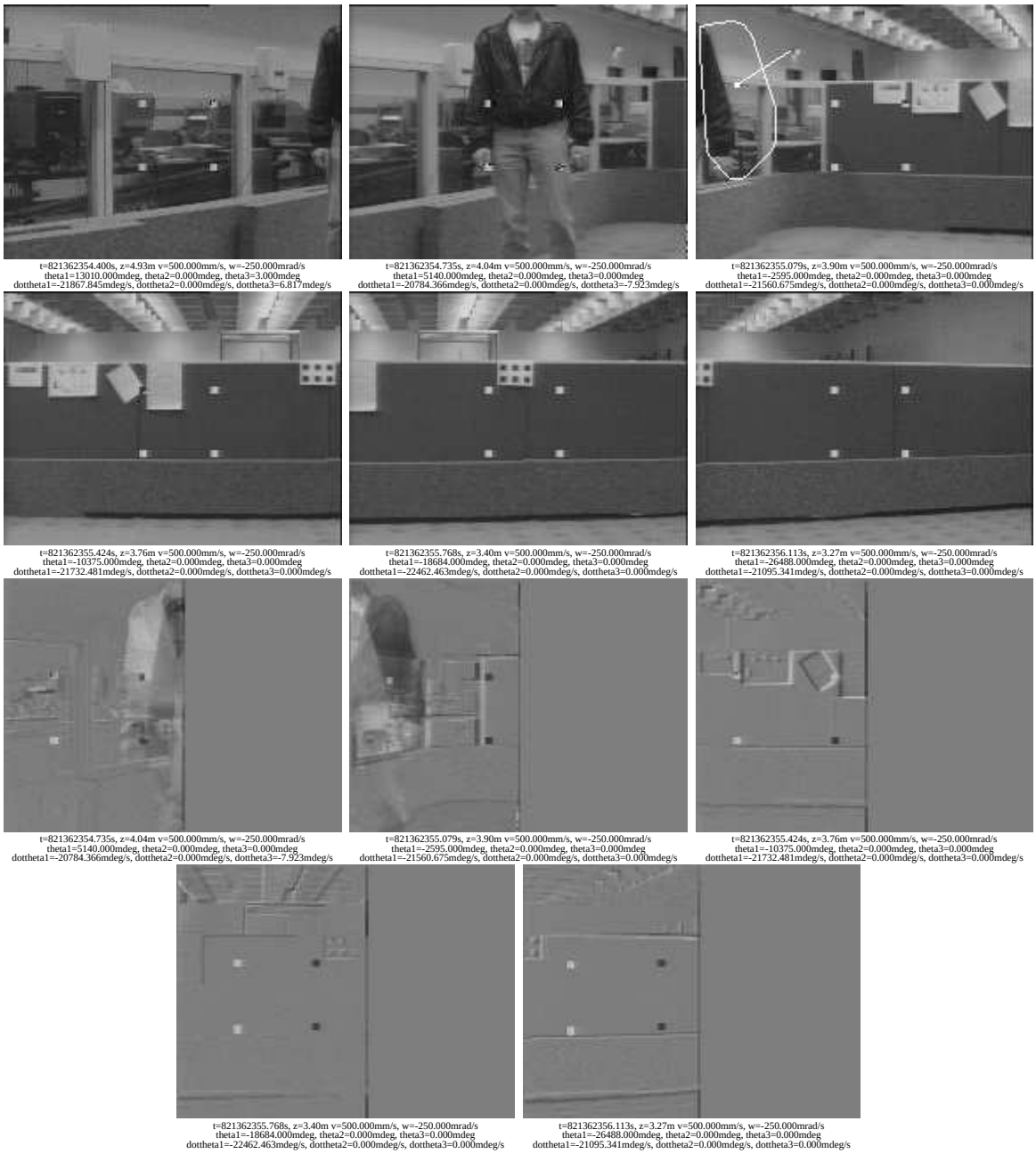

Abbildung 6.11: Kompensation der Eigenbewegung für große Kamerabewegungen.

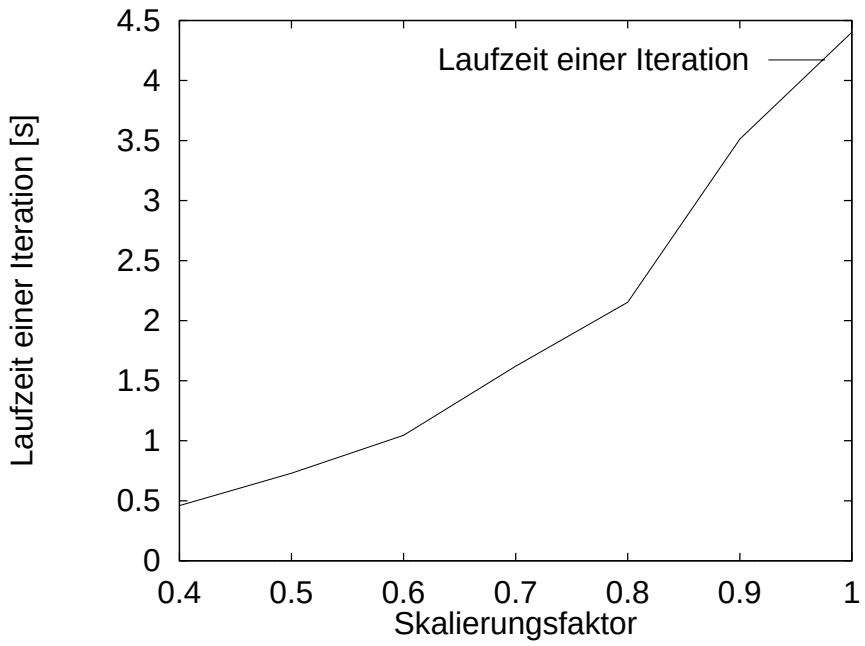

Abbildung 6.12: Gemittelte Laufzeit einer Iteration des Algorithmuses für ein 384×288 großes Bild bei verschiedene Skalierungsfaktoren.

bei Verschiebungen, die die Hälfte des Bildes ausmachten, die Eigenbewegung der Kamera zu kompensieren.

6.6 Zeitmessungen

Das Laufzeitverhalten des Programms EMO wurde anhand der in Abbildung 6.2 und Abbildung 6.3 gezeigten Bildsequenz für verschiedene Skalierungsfaktoren getestet. Mit Laufzeit wird im folgenden die verstrichene Echtzeit gemeint, die der Algorithmus benötigt. Die Laufzeiten wurden auf einem Sun 20 Computer gemessen. Das Ergebnis dieser Analyse ist in Abbildung 6.12 zusammengefaßt. Es wurden Laufzeitmessungen für die Skalierungsfaktoren 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 und 1.0 auf einer Bildsequenz mit einer Bildgröße von 384×288 durchgeführt. Dabei wurde nur die Laufzeit, die zur Bearbeitung eines Bildes benötigt wurde, berücksichtigt. Die Zeit, die zum Grabben oder Einlesen des Bildes benötigt wird, ist hier nicht enthalten. Insgesamt wurden 30 Laufzeiten für eine Sequenz gemessen. In der ersten Iteration wird lediglich die Initialisierung des Algorithmuses vorgenommen. Daher wird diese Laufzeit nicht berücksichtigt. Es wurde über die verbleibenden 29 Laufzeiten gemittelt. Diese gemittelten Laufzeiten sind in Abbildung 6.12 graphisch dargestellt.

Als kleinster Skalierungsfaktor wurde hier 0.4 gewählt, da eine weitere Verkleinerung des Bildes nicht sinnvoll gewesen wäre. Das Programm wurde zwar so entwickelt, daß bei einer kleineren Bildgröße auch die verwendeten Operatoren skaliert werden. Doch ist dies nur bis zu einem bestimmten Grad möglich. Die kleinste Größe der Strukturelemente der morphologischen Operatoren beträgt zum Beispiel 3×3.

Das quadratische Zeitverhalten ist deutlich zu erkennen. Das heißt, eine Verkleinerung des

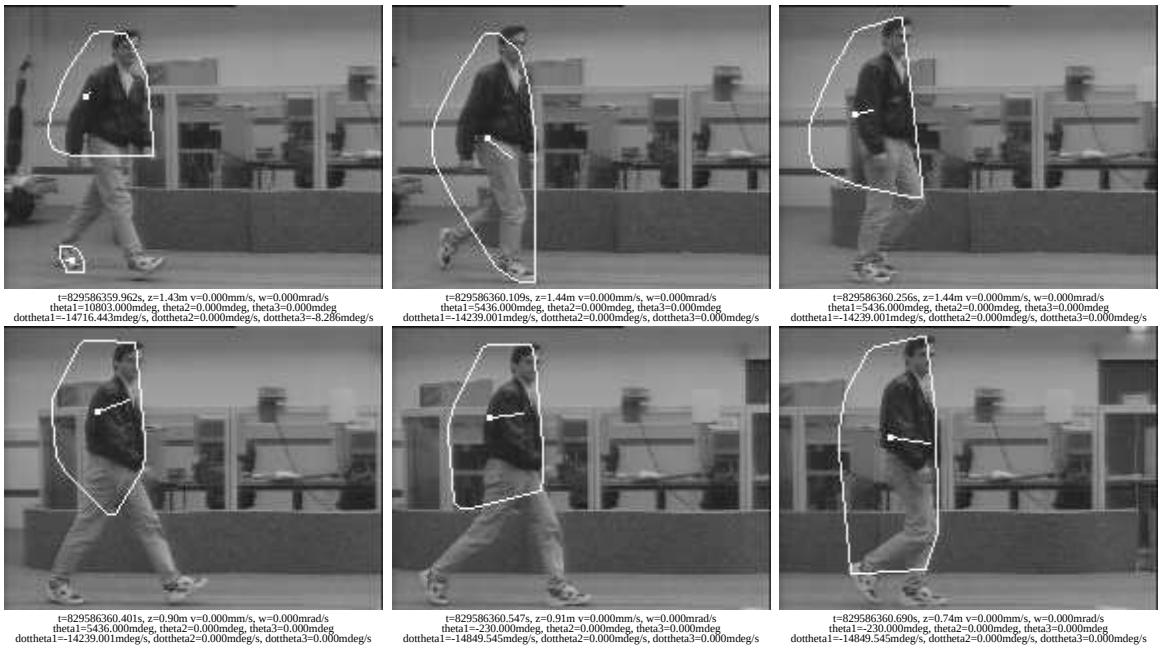

Abbildung 6.13: Extraktion bewegten Objektes bei rotatorischer Kamerabewegung. Hier sind nur die Bilder 20 bis 25 der Sequenz abgebildet (Sequenz 6 der Appendix).

Bildes auf 50% ergibt eine viermal schnellere Laufzeit. Natürlich ist die Laufzeit des Algorithmuses auch von der Anzahl der bewegten Objekte und der Anzahl der markanten Punkte im Bild abhängig. Die Laufzeit, die durch die Anwendung der Bildverarbeitungsoperatoren benötigt wird, ist jedoch deutlich größer. Denn bei der Korrelation markanter Punkte wird jeweils nur eine Region des Bildes betrachtet, und die bewegten Objekte werden durch die Punkte ihres umschließenden Polygons repräsentiert.

6.7 Ergebnisse

Der entwickelte Algorithmus ist in der Lage, translatorische und rotatorische Kamerabewegungen zu kompensieren. Abbildung 6.13 zeigt die Ausgabe des Algorithmuses bei einer Drehbewegung der Kamera um die Schwenk-Achse. Weitere Videosequenzen, auf denen der Algorithmus bewegte Objekte extrahiert, sind in der Appendix zusammengefaßt (siehe Abbildungen E.1 bis E.8). Die Videosequenzen zeigen, daß der Algorithmus bewegte Objekte extrahiert. In einigen Fällen werden aber auch fälschlicherweise bewegte Objekte extrahiert, obwohl sich an der Stelle kein bewegtes Objekt befindet. Dem Algorithmus gelingt es in einigen Videosequenzen auch ein bewegtes Objekt zu extrahiert, obwohl das bewegte Objekt teilweise verdeckt wird oder den Sichtbereich der Kamera verläßt.

Kapitel 7

Zusammenfassung und Ausblick

Es wurde ein robuster Algorithmus entwickelt, der bewegte Objekte aus einer mit dynamischer Kamera aufgenommenen Videosequenz extrahiert. Verschiedene, bereits aus der Literatur bekannte Verfahren bewegte Objekte zu extrahieren und Veränderungen einer Bildsequenz zu detektieren, wurden vorgestellt. Eine Bewertung der Verfahren wurde vorgenommen. Der Ansatz von Murray et al. [Murray et al. 94a] zunächst die Eigenbewegung der Kamera zu kompensieren wurde gewählt. Morphologische Operationen werden wie bei Murray et al. [Murray et al. 94a] eingesetzt, um Fehler in der Prädiktion der Kamerabewegung zu unterdrücken. In der vorliegenden Arbeit wurde im Gegensatz zu Murray et al., die nur rotatorische Kamerabewegungen kompensieren, beliebige Kamerabewegungen kompensiert.

Um beliebige Kamerabewegungen zu kompensieren, wird angenommen, daß die Entfernung zum stationären Hintergrund groß ist im Vergleich zu den Tiefenunterschieden innerhalb des stationären Hintergrundes. Die betrachtete Szene wird als Ebene in konstantem Abstand zur Kamera modelliert. Die Entfernung zum Hintergrund wird aus den Bilddaten und der Eigenbewegung der Kamera abgeschätzt.

Schwingungen der Kamera werden durch eine einfache Verschiebung des Bildes eliminiert. Diese Verschiebung des Bildes kann für einfache Kamerabewegungen auch den Einfluß von ungenauen Daten in der Berechnung der Kamerabewegung verringern. Um die Entfernung zum Hintergrund abzuschätzen und eine eventuell erforderliche Verschiebung des Bildes zu berechnen, werden markante Punkte aus den Bildern extrahiert und diese durch Korrelation einander zugeordnet.

Da die Eigenbewegung der Kamera kompensiert wird, kann wie auf einer mit stationärer Kamera aufgenommenen Bildsequenz weitergearbeitet werden. Zwischen je zwei Bildern der Sequenz werden die Veränderungen durch die Berechnung des Differenzbildes ermittelt. Es wurden Verfahren aus der Literatur gesucht, die Regionen des Differenzbildes zu bewegten Objekten zusammenzufassen. Aus den verschiedenen Verfahren wurde ein Algorithmus entwickelt, der bewegte Objekte extrahiert. Es wurde eine Heuristik entwickelt, Regionen, die durch die Anwendung morphologischer Operationen getrennt wurden zu einem Objekt zusammenzufassen. Zwischen den zusammengefaßten Regionen wird eine Bewegungshypothese aufgestellt. Trifft die Hypothese zu, das heißt, die Prädiktion des Objektes enthält Veränderungen im Differenzbild, so wird die Hypothese validiert. Falls die Hypothesen in folgenden Bildern validiert werden können, hat der Algorithmus ein bewegtes Objekt erkannt. Bereits extrahierte bewegte Objekte werden nach Anwendung der Heuristik eingesetzt, die Regionen nochmals zusammenzufassen.

Der Algorithmus wurde in ANSI-C auf einer Sun 20 implementiert. Dazu wurde die Stereo-Kamera und das Fahrzeug des Robotik-Labors modelliert. Aus den Statusinformationen der Kamera und des Fahrzeugs wurde die Transformationsmatrix der Kamerabewegung berechnet. Es wurde eine Reihe von unterschiedlichen Videosequenzen mit der Schwarz-Weiß-Kamera des Fahrzeuges Athos aufgenommen. Das Verhalten des Algorithmus wurde anhand dieser Sequenzen getestet und verbessert. Die durchgeführten Experimente wurden ausführlich beschrieben. Eine Iteration des Algorithmuses benötigt zur Zeit etwa 6s auf einer Sun 20 mit einer Bildgröße von 384×288.

Es gibt noch eine Vielzahl von Möglichkeiten, den entwickelten Algorithmus zu optimieren und somit die Ausführungszeit einer Iteration zu verkürzen. Einzelne Bildverarbeitungsoperatoren können paralellisiert werden [Bräunl et al. 95]. Für die Bildverarbeitung könnte zum Beipiel der Mas-Par Parallelrechner eingesetzt werden. Neben der Parallelisierung einzelner Operatoren ist auch ein Pipelining der Operatoren möglich.

Eine Aufmerksamkeitssteuerung könnte entwickelt werden. Bei schnellen bewegten Objekten könnten die Bilder in kurzen Zeitabständen aufgenommen werden, während sie bei langsam bewegten Objekten in langen Zeitabständen aufgenommen werden könnten.

Anstatt die Eigenbewegung des Fahrzeuges aus den Statusinformationen des Fahrzeuges und der Kamera zu berechnen, könnte die Eigenbewegung aus der Bildsequenz berechnet werden ([Fermüller 95], [Born 94]). Dann wäre der Algorithmus auch auf Bildsequenzen einsetzbar, die nicht mit einem Fahrzeug des Robotik-Labors aufgenommen wurden.

Falls die Ausführungszeit des Algorithmuses hinreichend verringert werden kann, sollten zur Berechnung der Eigenbewegung der Kamera direkt die Daten aus dem Kontrollalgorithmus des Fahrzeuges und der Kamera herangezogen werden. Interessant wäre auch die Anwendung des Algorithmuses bei einer dem Auge nachempfundenen Hardware [Wallace 94].

Anhang A

Das EMO-Bildformat

EMO ermöglicht es, die aufgenommenen Bildsequenzen abzuspeichern. Die Bilder werden im PGM-Format [Murray et al. 94b] abgespeichert. Das PGM-Format beginnt mit der Zeichenkette `P5` in der ersten Zeile. Es können einige Zeilen mit Kommentaren folgen. Eine Kommentarzeile hat als erstes Zeichen ein `#`. Danach folgt eine Zeile mit der Breite und Höhe des Bildes und dem maximalen Grauwert des Bildes als normaler Text. Nach dieser Zeile beginnt das eigentliche Bild. Jedes Pixel des Bildes hat 8 Bit. Die einzelnen Pixel des Bildes sind zeilenweise gespeichert.

Die Bilder einer Sequenz werden im PGM-Format gespeichert, da so eine einfache Betrachtung der Bilder über eines der gebräuchlichen Bildbetrachtungsprogramme leicht möglich ist. Die für EMO wichtigen Statusinformationen werden in Kommentarzeilen als normaler Text gespeichert. Im folgenden ist ein Beispiel für ein EMO-Bildkopf gegeben.

```
P5
#EMO-Picture File Version 1.0
#t=821361181.153742,frame=565
#model=stereo2,f=792.000000
#theta1=0.000000,theta2=0.000000,theta3=0.000000,theta4=0.000000
#dotTheta1=0.000000,dotTheta2=0.000000,dotTheta3=0.000000,dotTheta4=-8.114852
#v=500.000000,w=-250.000000
384 288 221
...
```

In der zweiten Zeile steht die Versionsnummer des EMO-Moduls. Die Zeit (in Sekunden), zu der das Bild aufgenommen wurde, folgt in der dritten Zeile. Neben der Zeit ist noch die Framenummer des Framegrabbers angegeben. In der nächsten Zeile ist dann das verwendete Kameramodell angegeben. Es existieren die folgenden Kameramodelle `stereo1`, `stereo2`. Dabei bezeichnet `stereo1` bzw. `stereo2` die linke bzw. rechte Stereokamera. Nach dem Kameramodell ist die Brennweite in Pixeln angegeben. Es folgen schließlich die Position der Kameragelenke in Milligrad und die Geschwindigkeiten der Kameragelenke in Milligrad pro Sekunde. Als letztes folgt die lineare Geschwindigkeit des Fahrzeuges in Millimeter pro Sekunde und die Winkelgeschwindigkeit in Millirad pro Sekunde.

Anhang B

Bildsequenzbetrachtungsprogramm

Um die von EMO aufgenommenen Bildsequenzen zu betrachten wurde ein Betrachtungsprogramm geschrieben. Das Programm hat den Namen `show`. Dem Programm können eine Reihe von Parametern beim Aufruf des Programms übergeben werden. Die Parameter werden durch `-Xparameter` in beliebiger Reihenfolge spezifiziert. Dabei steht `X` für den Namen des Parameters und `parameter` gibt den String an, der als Parameter `X` übergeben wird. Die folgenden Parameter können spezifiziert werden.

- `F` Spezifiziert den Dateinamen des Bildes. An den Dateinamen fügt das Programm `show` die Erweiterung `N.pgm` an. Dabei steht `N` für die Nummer des Bildes. Die Bilder werden beginnend mit der Nummer 0 der Reihe nach eingelesen.
- `T` Spezifiziert die Zeit, die zwischen dem Anzeigen aufeinander folgender Bilder gewartet wird. Die Zeit wird in Nanosekunden angegeben. Ist dieser Parameter nicht spezifiziert, so wird nicht gewartet.
- `L` Gibt die Anzahl der Bilder an, die maximal eingelesen werden. Ist dieser Parameter nicht spezifiziert, so werden alle Bilder beginnend bei 0 eingelesen. Höchstens jedoch 100 Bilder.
- `W` Gibt die Größe des Fensters an, in dem die Bildsequenz angezeigt wird. Die Größe wird in der Form `w,h` angegeben. Dabei steht `w` für die Breite des Bildes und `h` steht für die Höhe des Bildes.

Ein Aufruf des Programms könnte zum Beispiel wie folgt aussehen:

```
show -Fbild -T1000000 -L20 -W384,288
```

Dieser Aufruf liest der Reihe nach die Bilder `bild0.pgm` bis maximal Bild `bild20.pgm` ein. Das Programm wartet 1ms bevor das nächste Bild angezeigt wird. Das Fenster, in dem die Bildsequenz angezeigt wird, hat die Größe 384×288.

Anhang C

Aufbau des Programms

Die für die vorliegende Arbeit entwickelte Software wurde unter dem Namen EMO in die Projektverwaltung [Vogt et al. 96] eingecheckt. Der Source-Code wurde in die folgenden Dateien aufgeteilt.

`class.c`	Routinen für die Verwaltung von Äquivalenzklassen
`class.h`	Include-Datei zu `class.c`
`emo.c`	Hauptprogramm von EMO
`emo_edges.c`	Vista-Erweiterungen, die Polygone bearbeiten
`emo_edges.h`	Include-Datei zu `emo_edges.c`
`emo_error.c`	Routinen für die Fehlerbehandlung
`emo_error.h`	Include-Datei zu `emo_error.c`
`emo_lib.c`	Routinen der EMO-Bibliothek
`emo_lib.h`	Include-Datei zu `emo_lib.c`
`emo_process.c`	Routinen, für die Verarbeitung zweier Bilder einer Iteration
`emo_process.h`	Include-Datei zu `emo_process.c`
`emo_vista.c`	Vista-Erweiterungen, die Bilder bearbeiten
`emo_vista.h`	Include-Datei zu `emo_vista.c`
`emotest.c`	Demo-Programm zur Benutzung der EMO-Bibliothek
`matrix.c`	Routinen zur Matrix-Rechnung
`matrix.h`	Include-Datei zu `matrix.c`
`shared.c`	Routinen zur Erzeugung von Shared-Memory-Segmenten
`shared.h`	Include-Datei zu `shared.c`
`show.c`	Bildbetrachtungsprogramm
`timer.c`	Routinen für die Zeitmessung
`timer.h`	Include-Datei zu `timer.c`
`view.c`	Routinen der VIEW-Bibliothek
`VIEW.h`	Include-Datei zu `view.c`
`EMO.h`	Include-Datei zur EMO-Bibliothek

Anhang D

VIEW - Visualisierung für die Programmentwicklung

Um die Entwicklung des beschriebenen Algorithmuses zur Extraktion bewegter Objekte zu unterstützen, wurde ein Visualisierungsprogramm geschrieben. Da das Programm jedoch auch zur Entwicklung anderer Bilder verarbeitender Software eingesetzt werden kann, soll das Programm hier kurz beschrieben werden. Im folgenden wird dieses Programm kurz VIEW genannt.

In einem Algorithmus, der Bilder verarbeitet, werden in der Regel viele verschiedene Operatoren auf die Bilder angewandt, um diese zu bearbeiten. Kanten werden extrahiert, Bilder werden geglättet oder Störungen entfernt, um nur einige wenige Beispiele zu nennen. Die Operatoren hängen in der Regel von einigen Parametern ab. Doch welche Parameter sind für die vorliegenden Daten die richtigen? Wie stark sollte das Bild geglättet werden? Sollten viele Kanten oder nur die stärksten bestimmt werden? Werden alle Störungen unterdrückt? Können die Ergebnisse der einzelnen Bearbeitungsschritte direkt am Bildschirm verfolgt werden, so ist dem Entwickler eine Entscheidunghilfe bei der genauen Wahl der Parameter gegeben.

Existierende Software zur Bildverarbeitung geben dem Anwender in der Regel auch die Möglichkeit, Bilder zu visualisieren. Diese Möglichkeit besteht zum Beispiel bei den für die vorliegende Arbeit zur Verfügung stehenden Bildverarbeitungsprogrammen HORUS [Eckstein 94] und Vista [Pope et al. 94a]. Das Programm VIEW wurde entwickelt, um unabhängig von der eingesetzten Bildverarbeitungssoftware, Grauwertbilder zu visualisieren. Dabei wurde darauf geachtet, daß die Routinen zur Visualisierung relativ effizient implementiert wurden. Das Interface wurde einfach gehalten. Dem Anwender von VIEW soll möglichst viel Arbeit abgenommen werden. Möchte der Entwickler eines Programms bei einem Testlauf einige Bilder nur einmal kurz zur visuellen Kontrolle ausgeben, so sollte dies so einfach wie möglich sein. Daher ist das explizite Öffnen eines Fenstern bei VIEW nicht nötig. Ist das Fenster geschlossen, so wird es von VIEW geöffnet. Die Fenster werden mit XView [Heller 93], [Raalte 93] geöffnet und verwaltet. Zur Darstellung der Bilddaten wurde Xlib [Nye 93], [Nye 92] eingesetzt.

Die Größe der Fenster kann vom Betrachter verändert werden. Möchte der Betrachter ein Bild gerne etwas größer dargestellt haben, so kann der Betrachter das Fenster auf die gewünschte Größe bringen. Je nach gewünschter Größe des Fensters werden die Daten entweder verkleinert oder vergrößert. Der Anwender braucht sich nicht um eine Größenveränderung der Fenster kümmern. Dem Anwender wird neben der Darstellung der Bildinformationen in einem Fenster noch die Möglichkeit gegeben, Punkte, Linien, Rechtecke und Kreise in vorde-

finierten Farben in ein Fenster einzuzeichnen. Die folgenden Farben `VIEW_YELLOW` `VIEW_BLUE`, `VIEW_GREEN`, `VIEW_RED`, `VIEW_BLACK`, `VIEW_GRAY` und `VIEW_WHITE` stehen zur Verfügung. Die Koordinaten der Punkte, Linien, Rechtecke und Kreise werden immer relativ zu den vom Anwender eingesetzten Bildgrößen spezifiziert. Falls nötig werden die Koordinaten von VIEW intern entsprechend der aktuellen Fenstergröße angepaßt.

```
viewType *view_init(char *image, int imageW,int imageH);
```

Um das VIEW-Modul benutzen zu können, muß es zunächst durch einen Aufruf der Routine `view_init` initialisiert werden. Als Parameter `image` wird der Routine ein Zeiger auf ein Icon übergeben. Die Göße des Icons wird mit den Parametern `imageW` und `imageH` angegeben. Als Parameter `image` kann auch der Wert `VIEW_NOICON` übergeben werden, wenn kein Icon gewünscht wird. Der Anwender erhält ein sog. Handle zurück, das für weitere Routinen benötigt wird.

```
windViewType *view_addGrayWindow(viewType *view, char *name, int show, int w, int
h, unsigned char *data);
```

Mit der Routine `view_addGrayWindow` kann ein Grauwertbild in einem Fenster angezeigt werden. Als Parameter wird der Routine das Handle `view` übergeben. Mit dem Parameter `name` wird der Name des Fensters angegeben. Falls das Fenster noch nicht existiert, wird ein Fenster mit dem Namen `name` geöffnet. Über den Namen des Fensters kann der Anwender es später wieder referenzieren. Ist der Parameter `show` auf `TRUE` gesetzt, so wird das Fenster gleich geöffnet. Andernfalls bleibt das Fenster zunächst geschlossen.

Existiert das Fenster bereits, so wird lediglich das Grauwertbild in dem bereits existierenden Fenster angezeigt. Die anzuzeigende Grafik muß ab der durch den Zeiger `data` angegebenen Adresse zeilenweise abgelegt sein. Ein Byte stellt ein Pixel des Bildes dar. Weiß wird durch den Wert 255 erzeugt und Schwarz durch den Wert 0. Die Breite der Grafikdaten wird durch den Parameter `w`, die Höhe durch den Parameter `h` angegeben. Das Fenster hat die durch die Grafikdaten definierte Größe.

```
windViewType *view_addSizedGrayWindow(viewType *view, char *name, int show, int
windowW, windowH, int w, int h, unsigned char *data);
```

Analog zur Routine `view_addGrayWindow` kann mit der Routine `view_addSizedGrayWindow` ein Fenster geöffnet werden und ein Grauwertbild angezeigt werden. Allerdings bekommt das Fenster die durch die Parameter `windowW` und den Parameter `windowH` angegebene Größe. Das Bild wird entsprechend der Größe des Fensters skaliert, damit es vollständig im Fenster dargestellt werden kann.

```
void view_drawPoint(windViewType *wind int x,int y,int col);
```

Zeichnet einen Punkt mit den durch `x` und `y` spezifizierten Koordinaten im Fenster `wind`. Die Farbe des Punktes wird durch den Parameter `col` angegeben.

```
void view_drawLine(windViewType *wind int x1,int y1,int x2,int y2,int col);
```

Zeichnet eine Line vom Punkt mit den durch `x1` und `y1` spezifizierten Koordinaten zum Punkt mit den durch `x2` und `y2` spezifizierten Koordinaten im Fenster `wind`. Die Farbe der Linie wird

durch den Parameter `col` angegeben.

```
void view_drawRect(windViewType *wind int x,int y,int w,int h,int col);
```

Zeichnet ein Rechteck dessen linke obere Ecke sich im Punkt mit den durch `x` und `y` spezifizierten Koordinaten befindet. Das Rechteck hat die Breite `w` und die Höhe `h`. Die Farbe des Rechtecks wird durch den Parameter `col` angegeben.

```
void view_drawCirc(windViewType *wind int x,int y,int w,int h,int col);
```

Zeichnet einen Kreis. Die Göße des Kreises wird durch ein umschließendes Rechteck angegeben. Die linke obere Ecke des Rechtecks befindet sich im Punkt mit den durch `x` und `y` spezifizierten Koordinaten. Das Rechteck hat die Breite `w` und die Höhe `h`. Die Farbe des Kreises wird durch den Parameter `col` angegeben.

```
int *view_windWidth(windViewType *wind);
```

Die aktuelle Breite des Fensters `wind` kann mit der Routine `view_windWidth` abgefragt werden.

```
int *view_windHeight(windViewType *wind);
```

Die aktuelle Breite des Fensters `wind` kann mit der Routine `view_windHeight` abgefragt werden.

```
int *view_wind2Gray(windViewType *wind, unsigned char *s);
```

Die im Fenster `wind` dargestellten Daten können mit der Routine `view_wind2Gray` ausgelesen werden. Das Grauwertbild wird an der durch `s` angegebenen Adresse abgelegt. Eventuell farbig dargestellte Punkte, Linien, Rechtecke und Kreise werden mit weißen Grauwerten gespeichert.

```
void view_refresh(viewType *view);
```

Mit dem Aufruf der Routine `view_refresh` werden alle durch das über das Handle `view` geöffneten Fenster, falls notwendig, neu gezeichnet.

```
void view_refreshWindow(windViewType *wind);
```

Mit dem Aufruf der Routine `view_refreshWindow` wird das durch den Parameter `wind` spezifizierte Fenster, falls nötig, neu gezeichnet.

```
windViewType *view_findWindow(viewType *view,char *name);
```

Die Routine `view_findWindow` liefert das Handle des Fensters mit dem Namen `name`. Der Routine wird als weiterer Parameter das Handle `view` übergeben.

```
int view_closeWindow(viewType *view,char *name);
```

Mit der Routine `view_closeWindow` kann das durch den Namen `name` spezifizierte Fenster iconisiert werden. Der Routine wird als weiterer Parameter das Handle `view` übergeben.

```
int view_openWindow(viewType *view,char *name);
```

Ein iconisiertes Fenster kann mit der Routine `view_openWindow` wieder geöffnet werden. Der Routine wird als Parameter das Handle `view` und der Name `name` des Fensters übergeben.

```
windViewType *view_destroyWindow(windViewType *wind);
```

Mit der Routine `view_destroyWindow` kann das durch `wind` angegebene Fenster gelöscht werden. Das Fenster wird geschlossen und der von dem Fenster belegte Speicher wieder freigegeben.

```
void *view_exit(viewType *view);
```

Wenn der Anwender das Modul VIEW nicht mehr benötigt, so werden mit einem Aufruf von `view_exit` alle Fenster geschlossen und belegter Speicherbereich freigegeben.

Anhang E

Videosequenzen

Auf den folgenden Seiten sind einige Videosequenzen. Die bewegten Objekte, die das Programm EMO aus den Videosequenzen extrahierte, sind durch eine weiße Kontur markiert. Der Bewegungsvektor des Objektes in der Bildebene ist als Linie in das bewegte Objekt eingezeichnet.

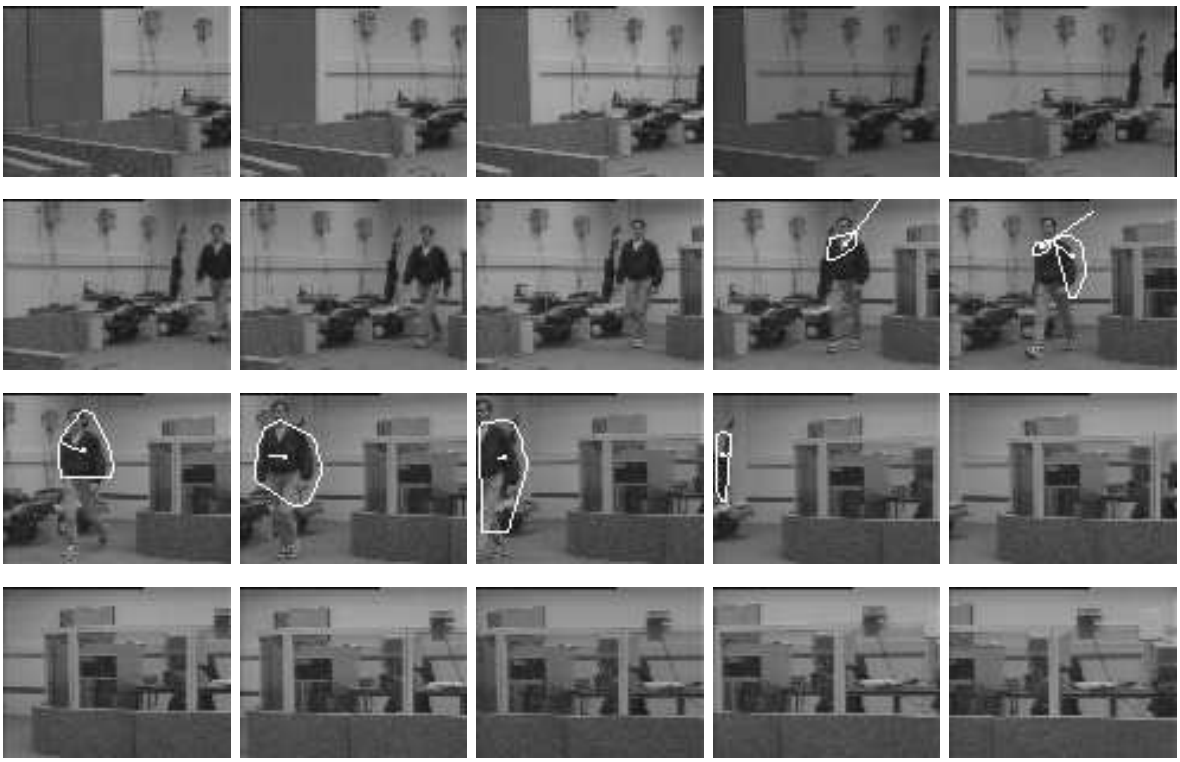

Abbildung E.1: Sequenz 1

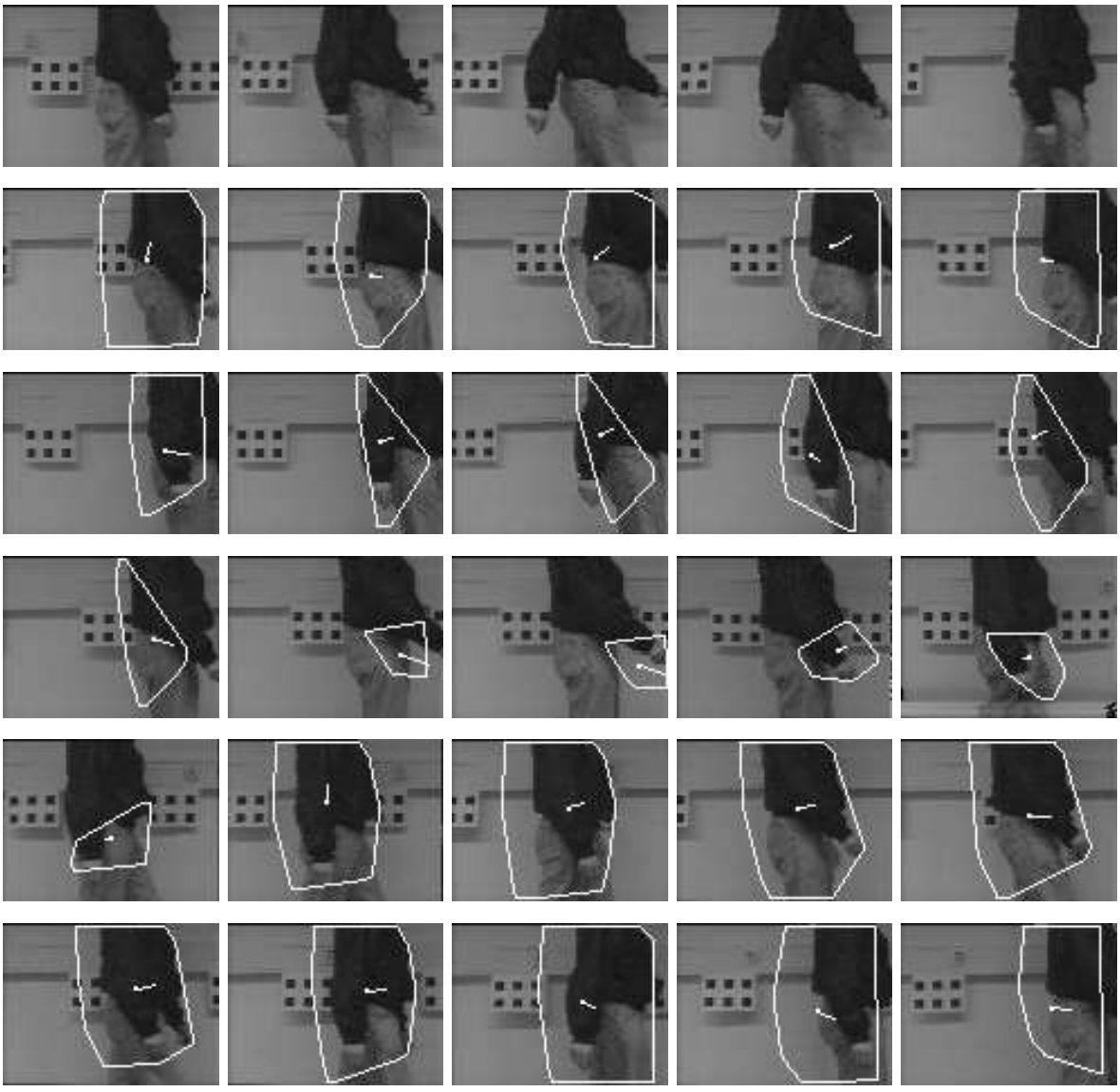

Abbildung E.2: Sequenz 2

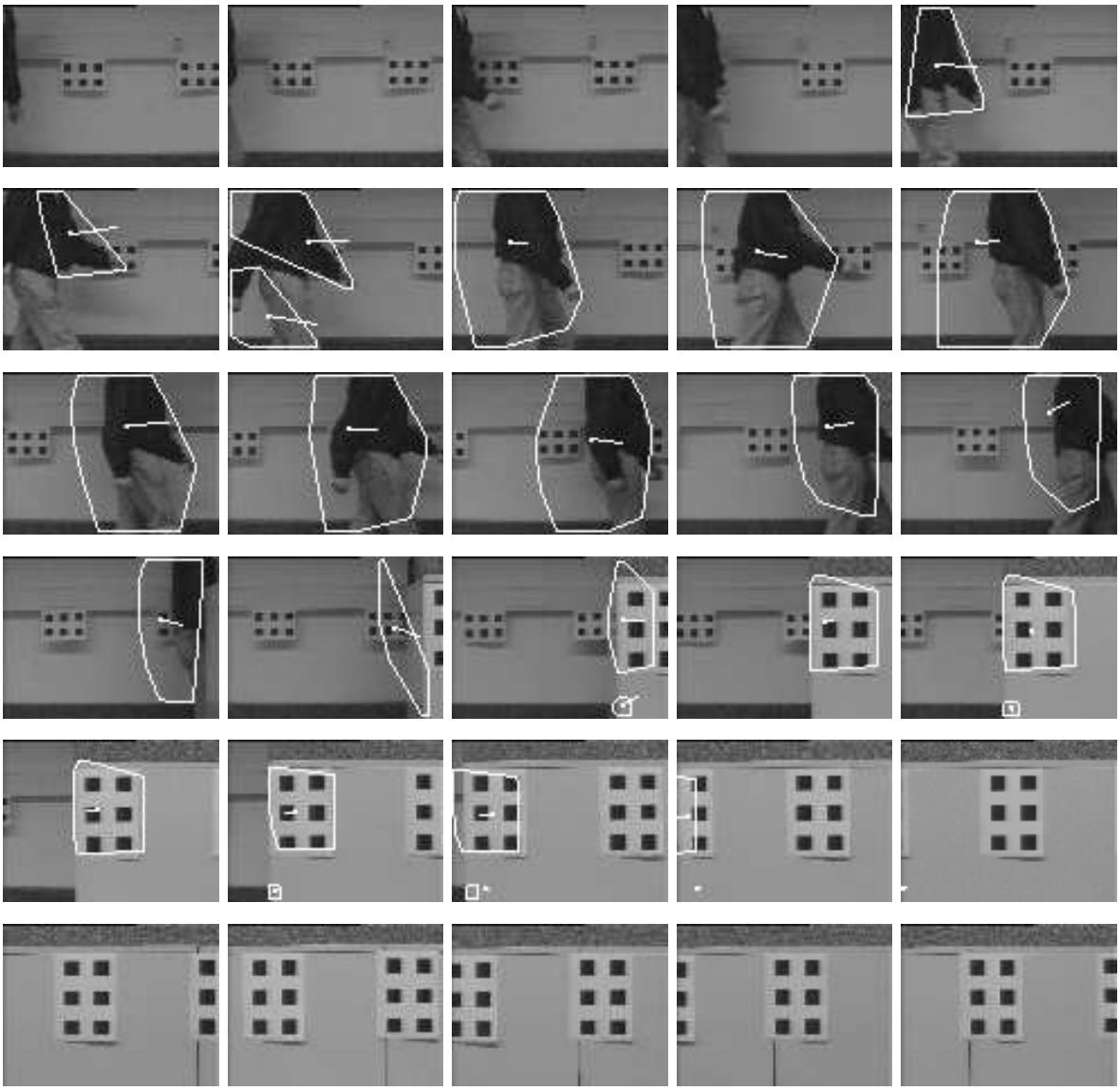

Abbildung E.3: Sequenz 3

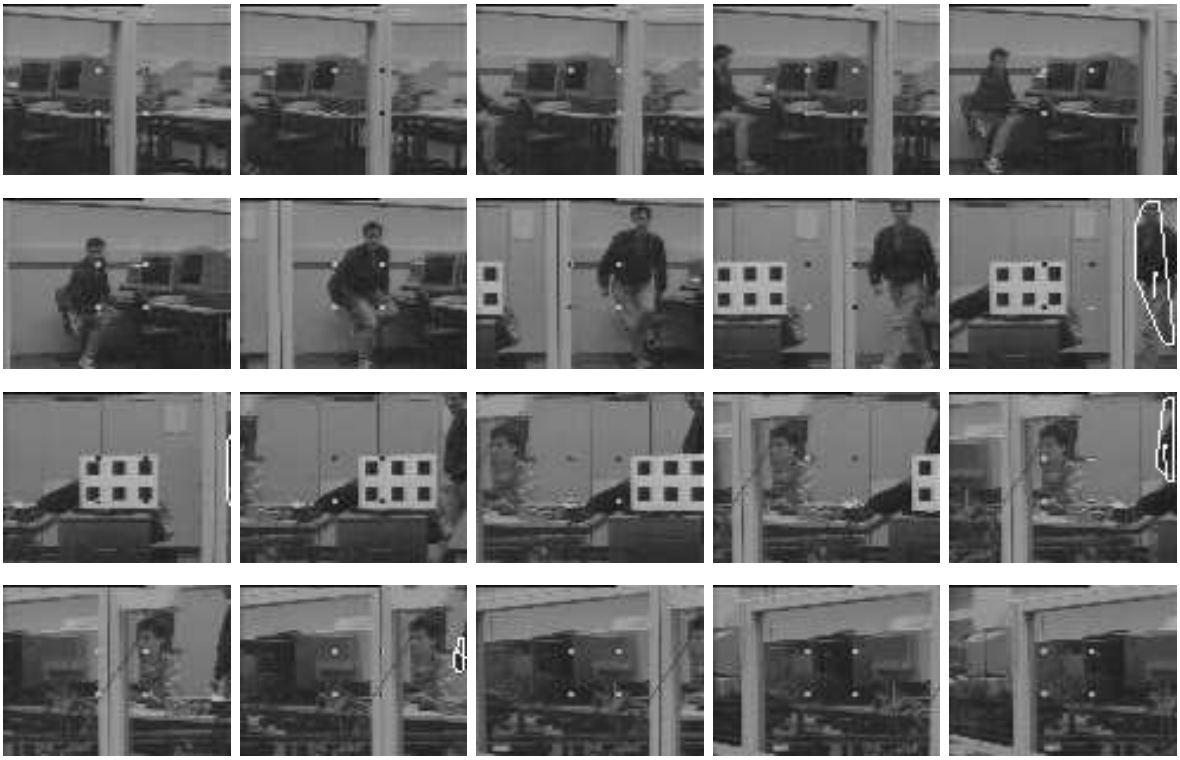

Abbildung E.4: Sequenz 4

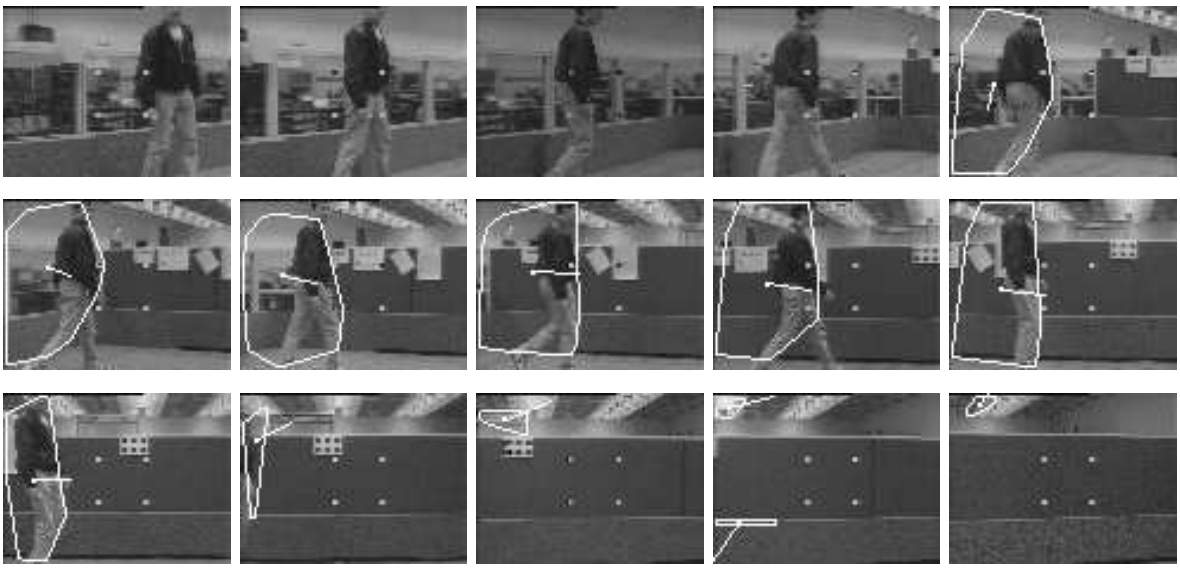

Abbildung E.5: Sequenz 5

Abbildung E.6: Sequenz 6

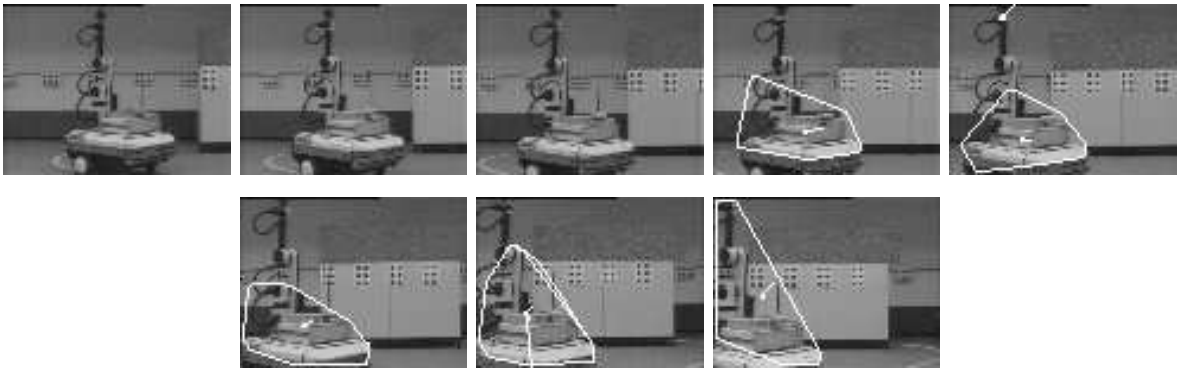

Abbildung E.7: Sequenz 7

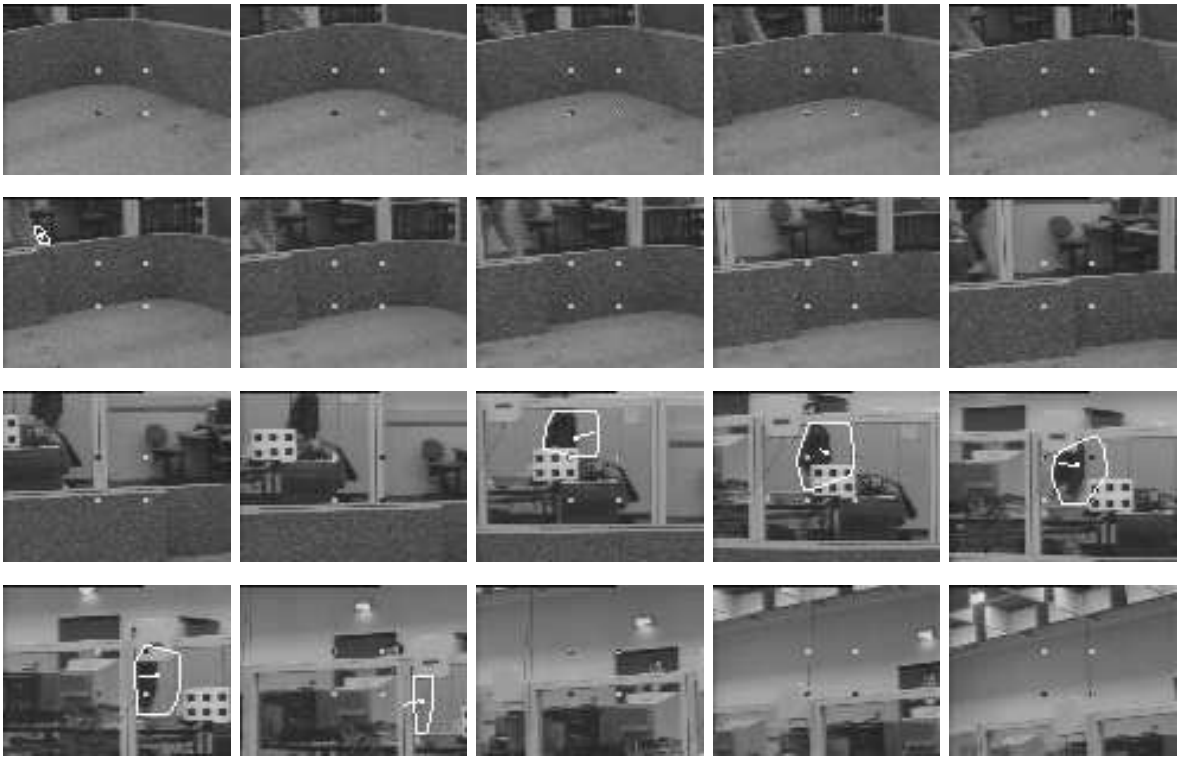

Abbildung E.8: Sequenz 8

Literaturverzeichnis

[Adobe 90] Adobe Systems. PostScript Language Reference Manual 2nd ed.. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1990.

[Aggarwal 86] J. K. Aggarwal. Motion and Time-Varying Imagery - An Overview. Workshop on Motion: Representation and Analysis, Kiawah Island Resort, Charleston/SC, pp. 1-6, May, 1986.

[Aloimonos 93] Yiannis Aloimonos (ed.). Active Perception. Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey, 1993.

[An et al. 88] Chae H. An, Christopher G. Atkeson and John M. Hollerbach. Model-Based Control of a Robot Manipulator. The MIT Press, Cambridge, Massachusetts, 1988.

[Ancona et al. 95] Nicola Ancona and Tomaso Poggio. Optical Flow from 1-D Correlation: Application to a Simple Time-to-Crash Detector. International Journal of Computer Vision, 14, pp. 131-146, Kluwer Academic Publishers, Boston, 1995.

[Bässmann et al. 89] Henning Bässmann und Philipp W. Besslich. Konturorientierte Verfahren in der digitalen Bildverarbeitung. Springer-Verlag, Berlin, 1989.

[Bichsel 1994] Martin Bichsel. Segmenting Simply Connected Moving Objects in a Static Scene. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No. 11, pp. 1138-1142, Nov. 1994.

[Born 94] Christof Born. Determining the Focus of Expansion by Means of Flowfield Projections. in W. G. Kropatsch und H. Bischof (Hrsg.): Tagungsband Mustererkennung, pp. 711-719, 1994.

[Bouthemy et al. 93] Patrick Bouthemy and Edouard Francois. Motion Segmentation and Qualitative Dynamic Scene Analysis from an Image Sequence. International Journal of Computer Vision, 10:2, pp. 157-182, Kluwer Academic Publishers, 1992.

[Bouthemy et al. 93] Patrick Bouthemy and Patrick Lalande. Recovery of moving object masks in an image sequence using local spatiotemporal contextual information. Optical Engineerings, Vol. 32, No. 6, pp. 1205-1212, Society of Photo-Optical Instrumentation Engineers, Jun. 1993.

[Bräunl 95] Thomas Bräunl. Scriptum Robotik I, Manipulatoren. Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner (IPVR), 1995.

[Bräunl et al. 95] Thomas Bräunl, Stefan Feyrer, Wolfgang Rapf, und Michael Reinhardt. Parallele Bildverarbeitung. Addison-Wesley Publishing Company, Bonn, 1995.

[Bräunl 96] Thomas Bräunl. Scriptum Robotik II, Mobile Roboter. Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner (IPVR), 1996.

[Carpenter 88] Roger H. S. Carpenter. Movements of the Eyes, 2nd Edition. Pion Limited, London, 1988.

[Cormen et al. 90] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest. Introduction to Algorithms. The MIT Press, Cambridge, Massachusetts, 1990.

[Craig 89] John J. Craig. Introduction to robotics: mechanics and control, 2nd ed.. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1989.

[Daniilidis et al. 95] K. Daniilidis, M. Hansen, Ch. Krauss und G. Sommer. Auf dem Weg zum künstlichen aktiven Sehen: Modellfreie Bewegungsverfolgung durch Kameranachführung. in Informatik aktuell: G. Sagerer, S. Posch und F. Kummert (Hrsg.): Mustererkennung 1995, pp. 277-284, Springer, 1995.

[Donohoe et al. 1988] Gregory W. Donohoe, Don R. Hush and Nasir Ahmed. Change Detection for Target Detection and Classification in Video Sequences. Proc. Int. Conf. Acoustics, Speech Signal Processing, pp 1084-1087, New York, 1988.

[Dubuisson et al. 95] Marie-Pierre Dubuisson and Anil K. Jain. Contour Extraction of Moving Objects in Complex Outdoor Scenes. International Journal of Computer Vision, 14, pp. 83-105, Kluwer Academic Publishers, 1995.

[Eckstein 94] W. Eckstein. HORUS/C Benutzerhandbuch. Institut für Informatik Technische Universität München, Mär. 1994.

[Fennema et al. 79] Claude L. Fennema and William B. Thompson. Velocity Determination in Scenes Containing Several Moving Objects. Computer Graphics and Image Processing, 9, pp. 301-315, Academic Press, Inc., 1979.

[Fermüller 95] Cornelia Fermüller. Passive Navigation as a Pattern Recognition Problem. International Journal of Computer Vision, 14, pp. 147-158 Kluwer Academic Publishers, Boston, 1995.

[Francois et al. 91] Edouard Francois and Patrick Bouthemy. Multiframe-based identification of mobile components of a scene with a moving camera. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Hawaii, pp. 166-172, June, 1991

[Frazier et al. 90] J. Frazier and R. Nevatia. Detecting Moving Objects from a moving Platform. Proc. DARPA Image Understanding Workshop, Pittsburgh, PA, pp. 348-355, 1990.

[Gall 91] Didier Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. Communications of the ACM, Vol. 34, No. 4, pp. 46-58, Apr 1991

[Gonzales et al. 92] Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing. Addison-Wesley Publishing Company, Inc., Reading, Massachusets, 1992.

[Griffiths et al. 76] H. B. Griffiths und P. J. Hilton. Klassische Mathematik in eeitgemäßer Darstellung, Band 1: Grundlagen, Mengenlehre und Arithmetik. Vandenhoeck & Ruprecht, Göttingen, 1976.

[Heller 93] Dan Heller, Updated for XView Version 3.2 by Thomas Van Raalte. The Definitive Guides to the X Window System: Volume Seven. XView Programming Manual for XView Version 3.2, 3rd Edition. O'Reilly & Associates, Inc., Aug 1993.

[Horn 86] Berthold Klaus Paul Horn. Robot Vision. The MIT Press Cambridge, Massachusetts, 1986.

[Horn et al. 81] Berthold K. P. Horn and Brian G. Schunck. Determining Optical Flow. Artificial Intelligence, 17, pp. 185-203, 1981.

[Hsu et al. 84] Y. Z. Hsu, H.-H. Nagel, and G. Rekers. New Likelihood Test Methods for Change Detection in Image Sequences. Computer Vision, Graphics, and Image Processing, 26, pp. 73-106, Academic Press, Inc. 1984.

[Huang 81] Thomas S. Huang (ed.). Image Sequence Analysis. Springer-Verlag Berlin, 1981.

[Huang 83] Thomas S. Huang (ed.). Image Sequence Processing and Dynamic Scene Analysis. Springer-Verlag Berlin, 1983.

[Jain 84] Ramesh C. Jain. Segmentation of Frame Sequences Objtained by a Moving Observer. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No. 5, pp. 624-629, 1984.

[Jain et al. 95] Ramesh Jain, Rangachar Kasturi and Brian G. Schunck. Machine Vision. McGraw-Hill, Inc., New York, 1995.

[Jain et al. 79a] Ramesh Jain and H.-H. Nagel. On the Analysis of Accumulative Difference Pictures from Image Sequences of Real World Scenes. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 2, pp. 206-214, 1979.

[Jain et al. 79b] Ramesh Jain, W. N. Martin, and J. K. Aggarwal. Segmentation through the Detection of Changes Due to Motion. Computer Graphics and Image Processing, 11, pp. 13-34, Academic Press, Inc., 1979.

[Jones et al. 93] Joseph L. Jones and Anita M. Flynn. Mobile Robots Inspiration to Implementation. A K Peters Wellesley, Massachusetts, 1993.

[Lee 90] Hsi-Jian Lee and Hsi-Chou Deng. Three-Frame Corner Matching and Moving Object Extraction in a Sequence of Images. Computer Vision, Graphics, and Image Processing, 52, pp. 210-238, Academic Press, Inc., 1990.

[Levi 93] Paul Levi. Radiometrische und fotometrische Grundlagen der Bildentstehung. in Bernd Radig (Hrsg.): Verarbeiten und Verstehen von Bildern, R. Oldenbourg Verlag GmbH, München, pp. 67-105, 1993.

[Levi et al. 95] Paul Levi, Thomas Bräunl, Harald Bayer, Susanne Gerl, Günter Mamier, Matthias Muscholl, Alexander Rausch, Marco Sommerau, Michael Vogt, Thilo Will. COMROS Cooperative Mobile Robotersysteme Stuttgart: Basis-Dokumentation. Bericht der Fakultät Informatik, Nr. 7/95, Universität Stuttgart, Sept. 1995.

[Marr 82] David Marr. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. W. H. Freeman and Company, New York, 1982.

[McKerrow 91] Phillip John McKerrow. Introduction to Robotics. Addison-Wesley Publishing Company, Inc., Sydney, Wokingham, 1991.

[Meyer et al. 94] Francois G. Meyer and Patrick Bouthemy. Region-Based Tracking Using Affine Motion Models in Long Image Sequences. CVGIP: Image Unterstanding, Vol. 60, No. 2, Sept, pp. 119-140, Academic Press, Inc., 1994.

[Meyer et al. 92] Francois G. Meyer and Patrick Bouthemy. Region-Based Tracking in an Image Sequence. in G. Sandini (Ed.): Proceedings of the Second European Conference on Computer Vision, pp. 476-484, Santa Margherita Ligure, Italy, May 1992.

[Milgram 79] D. L. Milgram. Region Extraction Using Convergent Evidence. Computer Graphics and Image Processing, 11, pp. 1-12, Academic Press, Inc., 1979.

[Moravec 77] Hans P. Moravec. Towards Automatic Visual Obstacle Avoidance. in Proc. 5th International Joint Conference on Artificial Intelligence, Vision-1: p. 584, 1977.

[Murray et al. 94a] Don Murray and Anup Basu. Motion Tracking with an Active Camera. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No. 5, pp. 449-459, May 1994.

[Murray et al. 94b] James D. Murray and William vanRyper. Encyclopedia of Graphics File Formats. O'Reilly & Associates, Inc., Sebastopol, CA, Jul 1994.

[Nagel 86] Hans-Hellmut Nagel. Image Sequences - Ten (octal) Years - From Phenomenology towards a Theoretical Foundation. Proceedings of the Eighth International Conference on Pattern Recognition, Paris, France, pp. 1174-1185. October 1986.

[Nelson 91] Randal C. Nelson. Qualitative Detection of Motion by a Moving Observer. International Journal of Computer Vision, 7:1, pp. 33-46, Kluwer Academic Publishers, 1991.

[Nye 92] Adrian Nye (Ed.). The Definitive Guides to the X Window System: Volume Two. Xlib Reference Manual for Version 11 R4/R5, 3rd Edition. O'Reilly & Associates, Inc., Jun 1992.

[Nye 93] Adrian Nye. The Definitive Guides to the X Window System: Volume One. Xlib Programming Manual for Version 11 R4/R5, 3rd Edition. O'Reilly & Associates, Inc., Jul 1993.

[Oswald et al. 96] Norbert Oswald, Susanne Gerl und Marc Ebner. Detecting moving objects by a dynamic camera. (eingereicht zur British Machine Vision Conference), 1996.

[Picton 89] P. D. Picton. Tracking and Segmentation of Moving Objects in a Scene. Proc. 3rd Int. Conf. Image Processing and its Applicat., Coventry, England, pp. 389-393, 1989.

[Pope 94] Art Pope. Vista Contributor's Guide. Vista Dokumentation, Jun, 1994.

[Pope et al. 94a] Arthur R. Pope and David G. Lowe. Vista: A Software Environment for Computer Vision Research. Vista Dokumentation, auch in CVPR, 1994.

[Pope et al. 94b] Art Pope, Daniel Ko and David Lowe. Introduction to Vista Programming. Vista Dokumentation, Jun, 1994.

[Raalte 93] Thomas Van Raalte (Ed.). The Definitive Guides to the X Window System: Companion to Volume 7. XView Reference Manual for XView Version 3.2. O'Reilly & Associates, Inc., Aug 1993.

[Radig 81] B. M. Radig. Image Region Extraction of Moving Objects. in T. S. Huang (ed.): Image Sequence Analysis, Springer-Verlag, Berlin, pp. 311-354, 1981.

[Rausch et al. 95] Alexander Rausch, Norbert Oswald and Paul Levi. Cooperative crossing of traffic intersections in a distributed robot system. SPIE Sensor Fusion and Networked Robotics VIII, Vol. 2589, pp. 218-229, Oct. 1995.

[Robosoft 94a] Robosoft. Robuter User's Manual V4.0. Robosoft, Neuilly-Plaisance, France, Jan 1994.

[Robosoft 94b] Robosoft. TO-40 V1.0 User's Manual. Robosoft, Neuilly-Plaisance, France, Jul 1994.

[Sedgewick 92] Robert Sedgewick. Algorithmen. Titel der englischen Ausgabe: Algorithms Addison-Wesley Publishing Company, Bonn, 1992.

[Shah et al. 84] Mubarak A. Shah and Ramesh Jain. Detecting Time-Varying Corners. Computer Vision, Graphics, and Image Processing, 28, pp. 345-355, Academic Press, Inc., 1984.

[Shio et al. 91] Akio Shio and Jack Sklansky. Segmentation of People in Motion. IEEE Workshop on Visual Motion, Princeton, NJ, pp 325-332, 1991.

[Skifstad et al. 89] Kurt Skifstad and Ramesh Jain. Illumination Independent Change Detection for Real World Image Sequences. Computer Vision, Graphics, and Image Processing, 46, pp. 387-399, Academic Press, Inc., 1989.

[Smith et al. 94] S. M. Smith and J. M. Brady. A Scene Segmenter; Visual Tracking of Moving Vehicles. Engineering Applications of Artificial Intelligence, Vol. 7, No. 2, pp. 191-204, Elsevier Science Ltd., 1994.

[Smith et al. 95] S. M. Smith and J. M. Brady. ASSET-2: Real-Time Motion Segmentation and Shape Tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, No. 8, pp. 814-820, Aug 1995.

[Sugimoto et al. 86] Satoshi A. Sugimoto, Hiroshi Matsuki, and Yoshiki Ichioka. Implementation of tracking and extraction of moving objects in successive frames. Applied Optics, Vol. 25, No. 6, pp. 990-996, Mar 1986.

[Thompson 80] William B. Thompson. Combining Motion and Contrast for Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 6, pp. 543-549, Nov. 1980.

[Thompson et al. 87] William B. Thompson and Ting-Chuen Pong. Detecting Moving Objects. IEEE International Conference on Computer Vision, London, England, pp. 201-208, 1987.

[Ullman 79] Shimon Ullman. The Interpretation of Visual Motion. The MIT Press Cambridge, Massachusetts, 1979.

[Vogt et al. 96] Michael Vogt, Marco Sommerau und Matthias Muscholl. Die Praxis der Projektverwaltung - Konfigurationsmanagement im Robotik-Projekt. Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner (IPVR), 1996.

[Wall et al. 84] Karin Wall and Per-Erik Danielsson. A Fast Sequential Method for Polygonal Approximation of Digitized Curves. Computer Vision, Graphics, and Image Processing, 28, pp. 220-227, Academic Press, 1984 .

[Wallace 94] Richard S. Wallace. Miniature Direct Drive Rotary Actuators II: Eye, Finger and Leg. in Proceedings 1994 IEEE International Conference on Robotics and Automation, pp. 1496-1501, IEEE Computer Society Press, Los Alamitos, California, May, 1994.

[Weng et al. 92] Juyang Weng, Narendra Ahuja and Thomas S. Huang. Matching Two Perspective Views. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 8, pp. 806-825, Aug 1992.

[Westberg 92] Lars Westberg. Hierarchical Contour-Based Segmentation of Dynamic Scenes. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 9, pp. 946-952, Sep 1992.

[Williams 80] Thomas D. Williams. Depth from Camera Motion in a Real World Scene. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 6, pp. 511-516, Aug 1980.

[Yalamanchili et al. 82] S. Yalamanchili, W. N. Martin, and J. K. Aggarwal. Extraction of Moving Object Descriptions via Differencing. Computer Graphics and Image Processing, 18, pp. 188-201, Academic Press, Inc. 1982.

[Zheng et al. 95] Qinfen Zheng and Rama Chellappa. Automatic Feature Point Extraction and Tracking in Image Sequences for Arbitrary Camera Motion. International Journal of Computer Vision, 15, pp. 31-76, Kluwer Academic Publishers, Boston, 1995.