Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik

Analyse und Vergleich von Algorithmen zur Bestimmung des optischen Flusses

Diplomarbeit
zur Erlangung des akademischen Grades Diplom-Informatiker

Autor: Martin Schumann
Gutachter: Prof. Dr. Hans-Dieter Burkhard
Gutachter: Prof. Dr. Ralf Reulke
Betreuer: Dr. Manfred Hild

Berlin, den 28. April 2011

Inhaltsverzeichnis

1. Einleitung

Die Natur hat im Laufe der Zeit viele sehr effektive Methoden herausgebildet, die es Lebewesen ermöglichen, sich in ihrer Umwelt zu orientieren und fortzubewegen. Zur Wahrnehmung der Umgebung und Bestimmung der Eigenbewegungen von Lebewesen gibt es neben der Auswertung von Beschleunigungs- und Drucksensoren viele weitere Verfahren. Fledermäuse und etliche Wasserbewohner nutzen die Reflektionen von Schallsignalen. Vögel setzen auf die Auswertung des Erdmagnetfeldes. Fast alle Lebewesen bedienen sich neben diesen Verfahren jedoch zusätzlich der visuellen Auswertung ihres Umfeldes. Das ist ein Indiz für die Mächtigkeit und universelle Einsetzbarkeit der visuellen Wahrnehmung. Bereits sehr einfache Lebewesen, wie zum Beispiel Einzeller, nutzen optische Sinneszellen zur Beobachtung ihrer Umwelt und Koordinierung ihrer Bewegung [WS07]. Die Erkennung der Beleuchtungsstärke und deren Änderungen ermöglicht bereits einfache Bewegungsmuster. Dazu zählen die Phototaxis (durch unterschiedliche Beleuchtungsstärken wird die Richtung der Fortbewegung von Lebewesen beeinflusst) oder die Photokinese (dabei hängt die Bewegungsgeschwindigkeit von der Beleuchtungsstärke ab). Ein wesentlicher Faktor ist aber auch die Auswertung von Lageänderungen von Bildmerkmalen auf den optischen Sensoren. Diese lassen einen Rückschluss auf Bewegungen von Objekten in der Umwelt oder Bewegungen des optischen Sensors selbst zu. Die Interpretation von sich ändernden optischen Sensorinformationen als Geschwindigkeit von Bildmerkmalen wird optischer Fluss genannt. Viele Untersuchungen wurden an Honigbienen durchgeführt. Diese nutzen den optischen Fluss, um ihren Flug zu steuern [SZLC96], die Höhe zu halten [PRF10] und Entfernungen abzuschätzen [EB95].

Es existieren verschiedene Algorithmen zur Bestimmung des optischen Flusses. Unterteilen lassen diese sich in zwei wesentliche Kategorien. Zum einen sind das biologisch inspirierte Verfahren. Dazu zählen der Reichardt-Detektor [Rei61] und auf Neuronen basierte Algorithmen, wie sie in der Arbeit [Wol07] vorgestellt sind. Zum anderen nicht biologisch inspirierte Verfahren. Diese lassen sich aufteilen in Block-Matching Algorithmen [Sin90], gradientenbasierte Verfahren [HS80] [LK81]

und solche, die auf Frequenz- und Phaseninformationen entsprechender Filter basieren, zum Beispiel die Phasenkorrelation [KH75]. Es existieren auch Mischformen, wie das Anwenden gradientenbasierter Verfahren auf Phaseninformationen [FJ90].

Mit diesen Algorithmen ist der optische Fluss in der gesamten Robotik nutzbar. Eine große Anzahl von möglichen Anwendungen ist denkbar, wie zum Beispiel Einkopplung der Werte in sensomotirische Schleifen, Kontrolle, Änderung oder Stabilisierung der eigenen Bewegung, sowie Reaktionen auf verschiedene Bewegungsreize aus der Umwelt. Flussinformationen können damit andere Sensorik, wie GPS-Positionserkennung, Ultraschall, Infrarot oder Beschleunigungssensoren ergänzen oder sogar ersetzen.

Anwendungen in der Robotik sind visuelles Homing [VM05] (Rückkehr zu einem vorher definierten Ausgangspunkt), optischer Fluss zur Start-, Lande- und Höhenkontrolle bei Flugzeugen [RF04] sowie allgemein zur Flugkontrolle [OGB04], Erkennen und Ausweichen von Hindernissen [SK07] [GC09], Pfadintegration und Positionsabschätzung [LS04] und Lagestabilisierung und Navigation bei Quadrokoptern [ZSWS10]. Eine Anwendung außerhalb des Gebietes der Navigation ist die Bildsegmentierung [IRP94].

1.1. Zielsetzung

Für diese Anwendungen des optischen Flusses, ist eine robuste, genaue und konsistente Detektion in den verschiedensten Einsatzumgebungen unerlässlich. An dieser Stelle setzt die vorliegende Arbeit an. Es werden Algorithmen untersucht, die es ermöglichen den optischen Fluss anhand von Bilddaten zu bestimmen. Viele wissenschaftliche Arbeiten untersuchen Algorithmen in Bezug auf komplexe Szenen oder sehr einfache, synthetische Videodaten [BSL+09] mit dem Ziel der Erkennung einzelner oder vieler kleiner verteilter Ereignisse in einer gesamten Bildszene. Die vorliegende Arbeit zielt jedoch ganz speziell auf Eingangsdaten ab, die typischerweise entstehen, wenn die Eigenbewegung einer Kamera im Verhältnis zu ihrer Umwelt bestimmt werden soll. Der Schwerpunkt der Untersuchungen liegt dabei auf der Stabilität und Konsitenz der Ausgabedaten in Bezug auf Störfaktoren, die üblicherweise beim Abfilmen von Umgebungsszenarien auftreten. Dazu zählen zum Beispiel Bildrauschen, Helligkeits- und Kontrastschwankungen, unterschiedliche Geschwindigkeiten mit denen sich das aufgenommene Bild verschiebt, verschiedene Richtungen dieser Verschiebung, etc.
Damit ergeben sich zu beantwortende Fragestellungen, wie:

Gibt der Algorithmus den optischen Fluss des Bildes, unabhängig von der Ausprägung und Verteilung kontrastreicher Kanten im Bild, an? Das ist wichtig, wenn der beobachtete Bildausschnitt beispielsweise von einer gut ausgeprägten Pflasterstruktur hin zu einer eintönigen Betonfläche schwenkt.
Oder:
Welche Ausgabe hat der Algorithmus, wenn sich der Bildausschnitt mit einer sehr schnellen Geschwindigkeit bewegt, die mitunter außerhalb des Detektionsradius liegt? Auch dieses Verhalten ist wichtig, wenn die Ausgabe in Regelschleifen eingekoppelt ist. Weniger gravierend ist es, wenn der Betrag der detektierten Geschwindigkeit nicht dem Soll entspricht. Die Richtung des Geschwindigkeitsvektors sollte allerdings der Sollrichtung entsprechen, damit keine Regelung in die falsche Richtung stattfindet, in Folge dessen der zu stabilisierende Roboter gänzlich ausbricht.

Diese und andere Fragen werden im Abschnitt 5 beantwortet.

1.2. Aufbau der Arbeit

Zu diesem Zweck ist die Arbeit in folgende Teile untergliedert:
Im ersten Teil *Theoretische Vorbetrachtung* werden mathematische Schreibweisen definiert und grundlegende Definitionen erläutert, die für die gesamte Arbeit gelten.
Es folgt der Abschnitt *Algorithmen zur Vorverarbeitung des Bildmaterials*. Dort wird eine Auswahl an Algorithmen präsentiert, die in geeigneter Weise das Eingangsbildmaterial für die eigentlichen Optischer-Fluss-Algorithmen aufbereiten.
Im dritten Teil werden aus der großen Zahl existierender Algorithmen zur Bestimmung des optischen Flusses geeignete ausgewählt. Dabei werden die Auswahlkriterien sowie die ausgewählten Algorithmen selbst erläutert.
Abschließend wird die Testumgebung beschrieben, Bildmaterial zur Untersuchung ausgewählt, Analysemethoden und -kriterien festgelegt und die Testergebnisse präsentiert.

2. Theoretische Vorbetrachtungen

2.1. Nomenklatur

Für Berechnungen wird in dieser Arbeit auf die folgenden einheitlichen Schreibweisen zurückgegriffen.

n, M	kursive, nicht fette Buchstaben sind Skalare
$\mathbf{x}$	Vektoren sind klein geschrieben und fett dargestellt $\mathbf{x} \coloneqq (x, y)^T$.
$\mathbf{P}$	Buchstaben für Matrizen sind groß und fettgedruckt.
$\mathbf{0}$	Nullvektor
$\mathbf{x}^*$	Mit den Alleinstellungsmerkmalen von Vektoren werden auch dessen Komponenten bezeichnet $\mathbf{x}^* \coloneqq (x^*, y^*)^T$.
$\mathbf{x} \cdot \mathbf{v}$	Das einfache Multiplikationszeichen bezogen auf zwei Vektoren berechnet das Skalarprodukt. $\mathbf{x} \cdot \mathbf{v} \coloneqq \mathbf{x}^T\mathbf{v}$
$c\mathbf{x}$	Bei der Multiplikation eines Vektors mit einem Skalar, wird das Skalar mit jedem Element des Vektors multipliziert. $c\mathbf{x} \coloneqq \begin{bmatrix} cx \\ cy \end{bmatrix}$
$< \mathbf{A}, \mathbf{B} >$	Diese Rechenoperation definiert ein Skalarprodukt zweier Matrizen. Beide Matrizen müssen die gleiche Größe $m \cdot n$ besitzen. $< \mathbf{A}, \mathbf{B} >= Spur(\mathbf{A}^T, \mathbf{B}) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} b_{ij}$
$\lvert\mathbf{v}\rvert_n$	n-Norm des Vektors $\mathbf{v}$ mit der Dimension R: $\lvert\mathbf{v}\rvert_n = \sqrt[n]{\sum_{i=1}^{R} v_i^n}$
$\lvert\mathbf{v}\rvert$	Zweiernorm des Vektors $\mathbf{v}$: $\lvert\mathbf{v}\rvert = \lvert\mathbf{v}\rvert_2$
$\lvert\mathbf{M}\rvert_n$	n-Norm der $R \times S$-Matrix $\mathbf{M}$: $\lvert\mathbf{M}\rvert_n = \sqrt[n]{\sum_{i=1}^{R} \sum_{j=1}^{S} m_{ij}^n}$
$\lvert a\rvert$	Betrag des Skalars a
$((m_{ij}))$	Doppelklammern um ein indiziertes Skalar ziehen die Matrix $\mathbf{M}$ mit allen Komponenten m_{ij} auf.
$I(\mathbf{x}, t)$	Intensitätsfunktion eines Videos

$\delta\mathbf{x}$	ist definiert als: $\delta\mathbf{x} \coloneqq \begin{bmatrix}\delta x \\ \delta y\end{bmatrix}$
∇	∇ ist ein Vektor der partiellen Ableitungen nach allen räumlichen Koordinaten, also $\nabla = \begin{bmatrix}\frac{\partial}{\partial x} \\ \frac{\partial}{\partial y}\end{bmatrix}$. Beispiel: $\nabla I \coloneqq \begin{bmatrix}\frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y}\end{bmatrix}$
rand(x)	Diese Funktion gibt einen Zufallswert im Interval von $[0, x]$ zurück.
$\langle x \rangle$	Für eine zeitabhängige Variable $x \coloneqq x(t)$ erzeugen die spitzen Klammern deren Durchschnitt über die Zeit.

2.2. Bildrepräsentation

Für die vorliegende Arbeit wird an dieser Stelle eine einheitliche mathematische Beschreibung des gegebenen Ausgangsbildmaterials definiert.

2.2.1. Diskrete Repräsentation

Bei der diskreten Bildrepräsentation ist ein Video als Folge von Einzelbildern gegeben:

$$\{\mathbf{P}^{(t)}\} = \{\mathbf{P}^{(0)}, \mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(l-1)}\}$$

mit $l, t \in \mathbb{N}$ und $0 \leq tl$, wobei l die Anzahl der Einzelbilder der Bildfolge ist. In diesem Fall besteht jedes Element der Folge nur aus einem einzigen Bild $\mathbf{P}$, welches die Helligkeitswerte der Aufnahme repräsentiert. Es handelt sich also um Graustufenbilder. Ein einzelnes Element dieser Bilderfolge kann aber auch drei Farbkanäle enthalten. Dabei handelt es sich lediglich um drei verschiedene Bilder, die jeweils einen Farbkanal enthalten. Das farbige Video entspricht also der Folge:

$$\{(\mathbf{P}_r, \mathbf{P}_g, \mathbf{P}_b)^{(t)}\} = \{(\mathbf{P}_r, \mathbf{P}_g, \mathbf{P}_b)^{(0)}, (\mathbf{P}_r, \mathbf{P}_g, \mathbf{P}_b)^{(1)}, \ldots, (\mathbf{P}_r, \mathbf{P}_g, \mathbf{P}_b)^{(l-1)}\} \,,$$

wobei jedes der Bilder $\mathbf{P}_r$, $\mathbf{P}_g$ und $\mathbf{P}_b$ genauso definiert ist wie $\mathbf{P}$.

Das einzelne Graustufenbild liegt als zweidimensionale $M \times N$-Matrix vor:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ p_{M1} & p_{M2} & \cdots & p_{MN} \end{bmatrix}$$

$M, N \in \mathbb{N}$ sind Höhe und Breite des Bildes. Diese ist in jedem Einzelbild $\mathbf{P}$ der Bildfolge gleich. Der Wertebreich $\mathbb{G}$ der Matrixelemente $p_{ij} \in \mathbb{G}$, mit $i = 1, .., M$ und $j = 1, .., N$, wird Grauwertmenge genannt. In der Praxis sind die Grauwertmengen endlich, wobei die Anzahl der Elemente häufig einer 2er-Potenz entspricht. So hat ein 8bit-Graustufenbild die Grauwertmenge $\mathbb{G} := \{0, \ldots, 255\}$. Die 0 entspricht Schwarz, 255 Weiß. Die Werte dazwischen sind den entsprechenden Grauwerten linear zugeordnet.

2.2.2. Kontinuierliche Repräsentation

Die diskrete Definition eines Videos ist nicht für alle Algorithmen geeignet. So ist es für die gradientenbasierten Verfahren (siehe 4.2) notwendig, das Video in Raum und Zeit als stetige, dichte Funktion zu definieren. Diese Funktion ist gegeben als:

$$I(\mathbf{x}, t) \ ,$$

wobei $\mathbf{x} := (x, y)^T$ die räumlichen Bildkoordinaten sind und t die Zeitvariable ist. Der Wertebereich (Grauwertmenge) $\mathbb{G}$ dieser Funktion ist gegeben als $\mathbb{G} \subset \mathbb{R}$ und $\mathbb{G} := [0, 1]$. Die Bildbreite wird auf der x-Achse beschränkt auf das Intervall $[0, N]$ und auf der y-Achse als Höhe auf $[0, M]$. Auch hier werden die einzelnen Farbkanäle mit je einer Funktion $I_r(\mathbf{x}, t)$, $I_g(\mathbf{x}, t)$ und $I_b(\mathbf{x}, t)$ beschrieben. I selbst stellt nur Grauwerte des Bildes dar. Die Berechnung von I aus I_r, I_g und I_b wird in Kapitel 3.1 gezeigt.

2.3. Optischer Fluss

2.3.1. Definition

Der optische Fluss ist ein Vektorfeld, welches jedem Pixel eines Bildes einen Geschwindigkeitsvektor zuordnet, mit dem es sich im Verlauf der Zeit über das Bild bewegt [HS80]:

$$\mathbf{v}(\mathbf{x}, t) = \begin{bmatrix} u(\mathbf{x}, t) \\ v(\mathbf{x}, t) \end{bmatrix} \ .$$

Der Geschwindigkeitsvektor für einen einzelnen Bildpunkt zu einer bestimmten Zeit wird mit

$$\mathbf{v} := \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.1}$$

bezeichnet. $\mathbf{v}$, u und v sind, wie es der Definition schon zu entnehmen ist, abhängig von der Zeit und der Bildkoordinate. Der Vektor $\mathbf{v}$ lässt sich in Abhängigkeit davon definieren:

$$\mathbf{v}(\mathbf{x}, t) \coloneqq \begin{bmatrix} u(\mathbf{x}, t) \\ v(\mathbf{x}, t) \end{bmatrix} .$$

Die Funktion $u(\mathbf{x}, t)$ beziehungsweise u entspricht der x-Komponente des Geschwindigkeitsvektors $\mathbf{v}(\mathbf{x}, t)$ bzw. $\mathbf{v}$. Ist für eine Stelle $\mathbf{x}$ also die Verschiebung Δx der Bildinformation in einer Zeiteinheit Δt bekannt, dann gilt $u(\mathbf{x}, t) = \frac{\Delta x}{\Delta t}$. Für die y-Komponente und v gilt das gleiche.

Die Betrachtung der Maßeinheit des optischen Flusses legt zu Grunde, dass zwei Pixel ($\mathbf{x}^{(t_1)}$ und $\mathbf{x}^{(t_2)}$) stets einen definierten Abstand auf der Bildfläche zueinander haben. Die Einheit des optischen Flusses ist damit gegeben als Weg (Abstand der Pixel $|\mathbf{x}^{(t_2)} - \mathbf{x}^{(t_1)}|$ in denen sich ein und dasselbe Bildmerkmal zu den Zeitpunkten t_1 und t_2 befindet) pro Zeiteinheit (Zeitraum $t_2 - t_1$ in dem sich das Bildmerkmal verschoben hat):

$$[\mathbf{v}(\mathbf{x}, t)] = \frac{\left[\ \left|\mathbf{x}^{(t_2)} - \mathbf{x}^{(t_1)}\right|\ \right]}{[t_2 - t_1]}$$

In der folgenden Arbeit wird die Einheit als *Pixel / Frame* geschrieben.

2.3.2. Bedingungen

Für die Definition des optischen Flusses müssen einige Nebenbedingungen erfüllt sein. Die erste ist die *Helligkeitsbeständigkeit*. In einer kleinen Zeiteinheit, in der ein Objekt seine Lage im Bild ändert, darf dessen Heilligkeitseindruck, den es in der Kamera hinterlässt, nicht, beziehungsweise nur vernachlässigbar gering, schwanken. Treten keine Helligkeitsschwankungen durch Bewegungen einzelner Objekte auf und ändert sich die Position der Kamera zu ihrer Umwelt nicht, führt das zu folgendem Zusammenhang:

$$I(\mathbf{x}, t + \delta t) = I(\mathbf{x}, t)$$

oder anders ausgedrückt:

$$I_t(\mathbf{x}, t) = 0 ,$$

wobei $I_t = dI/dt$ die partielle Ableitung der Intensitätsfunktion nach der Zeit ist.

Das Korrespondenz- und Aperturproblem darf nicht eintreten. Beide werden in den folgenden Abschnitten beschrieben.

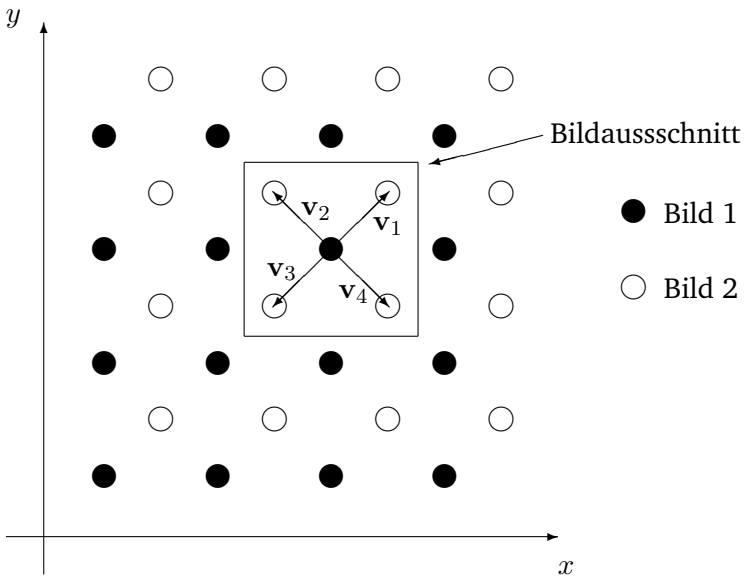

Abbildung 2.1.: *Veranschaulichung des Korrespondenzproblems*

2.3.2.1. Korrespondenzproblem

Das Korrespondenzproblem soll anhand des Beispiels in Abbildung 2.1 näher erläutert werden. Es zeigt Bildmerkmale zweier aufeinander folgender Einzelbilder eines Videos. Diese sind überlagert dargestellt (Bild 1 und Bild 2). In beiden Bildern sind die Merkmale gleich ausgeprägt, in Bild zwei dient die fehlende Füllung der Kreise lediglich als Unterscheidungsmerkmal. In dem gegebenen Bildausschnitt ist eine Unterscheidung, ob es sich bei dem optischen Fluss um $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ oder $\mathbf{v}_4$ handelt, nicht möglich. Diese Unterscheidungsunsicherheit wird als Korrespondenzproblem bezeichnet.

2.3.2.2. Aperturproblem

Ein weiteres Problem stellt die Begrenzung des Bildausschnitts dar. Dadurch gehen wesentliche Informationen, die zur Bestimmung des optischen Flusses notwendig sind, verloren. Genau wird das in Abbildung 2.2 gezeigt. Auch hier sind zwei aufeinanderfolgende Bilder überlagert. Da die Bildinformationen, die die Ausschnitte 1 und 3 zeigen, nicht für die genaue Bestimmung des optischen Flusses ausreichen, lässt sich zwischen den Flussvektoren $\mathbf{v}_1$, $\mathbf{v}_2$ und $\mathbf{v}_3$, beziehungsweise $\mathbf{v}_5$, $\mathbf{v}_6$ und $\mathbf{v}_7$ nicht unterscheiden. Lediglich Bildausschnitt 2 zeigt die Ecke als hinreichende Information, um den tatsächlichen Flussvektor $\mathbf{v}_4$ zu bestimmen.

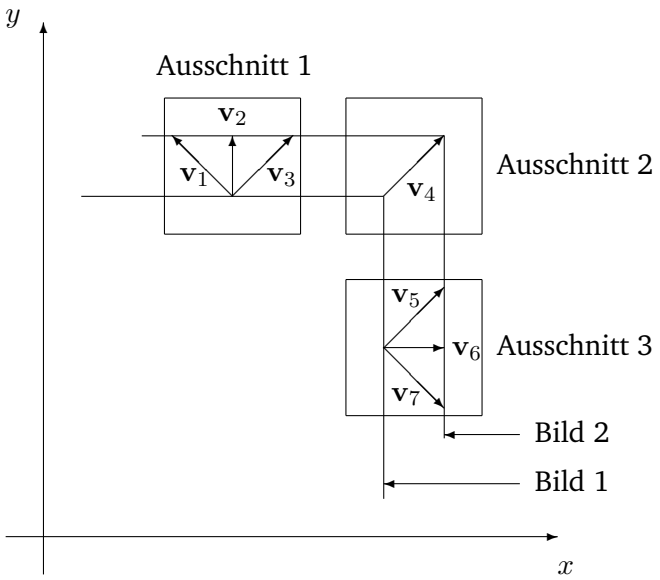

Abbildung 2.2.: *Veranschaulichung des Apperturproblems*

2.3.3. Flussfelder

Für die Entstehung der hier betrachteten Flussfelder des optischen Flusses wird in Abbildung 2.3 eine Anordnung der Kamera zu ihrer Umwelt definiert. Bei den folgenden Betrachtungen wird unterschieden, ob sich die angegebenen Koordinaten und Größen auf das Weltkoordinatensystem (x', y', z') aus Abbildung 2.3 beziehen oder auf die Bildkoordinaten aus Kapitel 2.2 (x, y). Die Geraden $g_{x'}$, $g_{y'}$ und $g_{z'}$ verlaufen parallel zur entsprechenden Achse des Koodinatensystems und schneiden sich genau in der Mitte des Bildsensors. Die Kamera nimmt eine Oberfläche auf, die parallel zur x'-y'-Ebene liegt (im Bild mit einem hellgrauen Schachbrettmuster dargestellt). Die Abbildungsgrenzen der Kamera sind auf der x'-Achse x'_{min} und x'_{max} und auf der y'-Achse y'_{min} und y'_{max}. $P_{x'y'}(\hat{x}', \hat{y}')$ ist der Schnittpunkt der Geraden $g_{z'}$ mit der abgebildeten Oberfläche. Dieser Punkt wird, bei Sicht der Kamera parallel zur z'-Achse, im Bildkoodinatensystem bei $(\hat{x}, \hat{y})$ abgebildet. S sei der Schnittpunkt der Geraden $g_{x'}$, $g_{y'}$ und $g_{z'}$.

Die Projektion der Umwelt auf den Kamerasensor erfolgt vereinfacht über ein Lochkameramodell, wie es in Abbildung 2.4 dargestellt ist. $a = d - f$ ist die Objektweite, f die Bildweite beziehungsweise Brennweite und $F = (x'_0, y'_0, z'_0)$ das Projektionszentrum. d entspricht damit dem Abstand $\overline{P_{x'y'}S}$. Die Projektion der Umgebung auf die Sensorfläche kann in diesem einfachen Fall durch die Kollinearitätsgleichun-

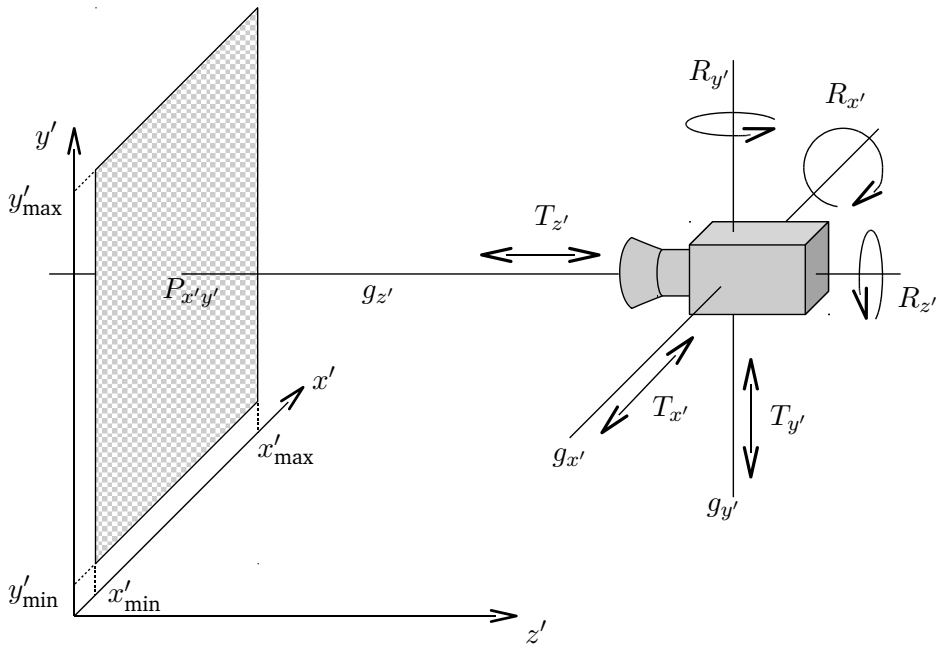

Abbildung 2.3.: *Kamera-Ebene-Anordnung*

gen

$$x = f\frac{r_{11}(x' - x'_0) + r_{12}(y' - y'_0) + r_{13}(z' - z'_0)}{r_{31}(x' - x'_0) + r_{32}(y' - y'_0) + r_{33}(z' - z'_0)} + \hat{x} \tag{2.2}$$

$$y = f\frac{r_{21}(x' - x'_0) + r_{22}(y' - y'_0) + r_{23}(z' - z'_0)}{r_{31}(x' - x'_0) + r_{32}(y' - y'_0) + r_{33}(z' - z'_0)} + \hat{y} \tag{2.3}$$

beschrieben werden, wobei r_{ij} die Komponenten einer 3×3 Rotationsmatrix $\mathbf{R} = ((\, r_{ij}\,))$ mit $1 \leq i, j \leq 3$ sind, die die Blickrichtung der Kamera in den Weltkoordinaten definiert. Eine Rotation um die Gerade $g_{z'}$ mit dem Winkel α wird von der Matrix

$$\mathbf{R}_{z'}(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

und eine Rotation um die Gerade $g_{y'}$ von Matrix

$$\mathbf{R}_{y'}(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

beschrieben. Entspricht $\mathbf{R}$ einer Einheitsmatrix, findet keine Drehung statt.

Abbildung 2.5 zeigt nun Optische-Fluss-Felder, wie sie bei bestimmten Eigenbewegungen der Kamera entstehen. Für die Berechnung der Vektorfelder werden zwei Bilder mit dem Zusammenhang

$$I(\mathbf{x}^*, t - \Delta t) = I(\mathbf{x}, t)$$

im Bildkoordinatensystem genutzt. Jeder abgebildete Bildpunkt hat in der Zeitspanne Δt seine Position um $\Delta\mathbf{x} := (x - x^*, y - y^*)^T$ geändert. Aus der Definition des optischen Flusses folgt direkt:

$$\mathbf{v} := (\mathbf{x} - \mathbf{x}^*) \cdot \frac{1}{\Delta t} \tag{2.4}$$

mit

$$\mathbf{x}^* := \mathbf{x} - \Delta\mathbf{x} \; . \tag{2.5}$$

2.3.3.1. Translation in x'- und y'-Richtung

Das Flussbild in 2.5a entsteht durch Translation der Kamera entlang der Geraden $g_{x'}$ um $\Delta x'$. Aus Gleichung 2.2 und $\mathbf{R}(0)$ folgt

$$x^* = f\frac{x' - {x'_0}^*}{z' - z'_0} + \hat{x} \; , \tag{2.6}$$

$$x = f\frac{x' - x'_0}{z' - z'_0} + \hat{x} \tag{2.7}$$

und aus 2.3

$$y^* = f\frac{y' - y'_0}{z' - z'_0} + \hat{y} \; , \tag{2.8}$$

$$y = f\frac{y' - y'_0}{z' - z'_0} + \hat{y} \; . \tag{2.9}$$

Die Verschiebung der Kamera in ihrer Umwelt um $\Delta x'$ verändert das Projektionszentrum der Kamera:

$$({x'_0}^*, {y'_0}^*, {z'_0}^*) := (x'_0 - \Delta x', y'_0, z'_0) \; . \tag{2.10}$$

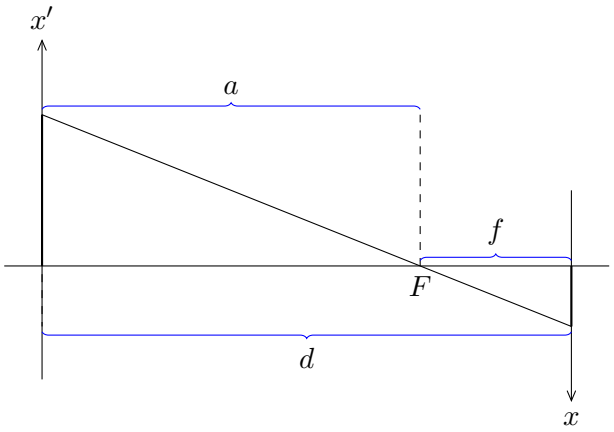

Abbildung 2.4a.:
Darstellung des Strahlengangs bei der Bildaufnahme

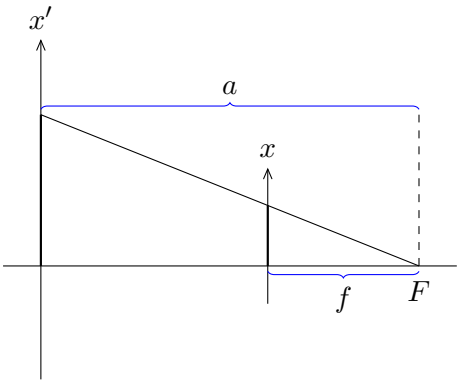

Abbildung 2.4b.:
Äquivalente Darstellung durch Ausnutzen des Strahlensatzes

Abbildung 2.4.: *Abbildungsgeometrie der Kamera*

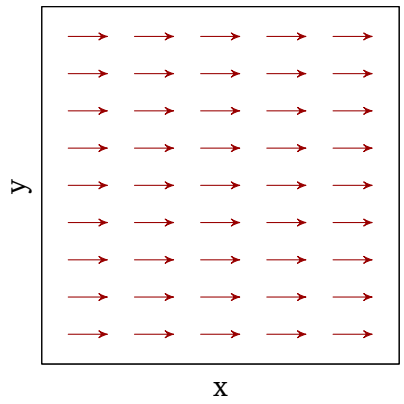

Abbildung 2.5a.:
Translation entlang $g_{x'}$ mit $\Delta x' = 6$.
$U = 6$, $V = 0$, $W = 1$, $Z = 0$

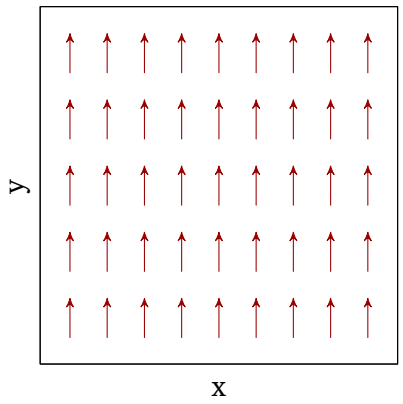

Abbildung 2.5b.:
Translation entlang $g_{y'}$ mit $\Delta y' = 6$.
$U = 0$, $V = 6$, $W = 1$, $Z = 0$

Abbildung 2.5.: *Charakteristische Flussfelder*

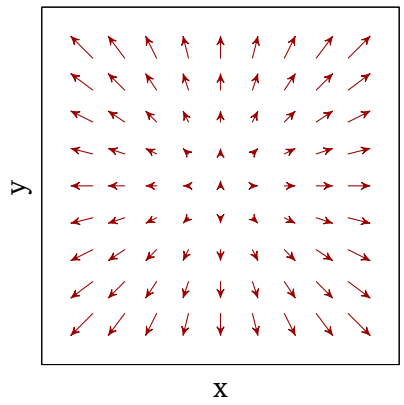

Abbildung 2.5c.:
Translation entlang $g_{z'}$ auf die x'-y'-Ebene hinzu mit $d = 500$, $d^* = 600$ und $f = 30$. $U = 0$, $V = 0$, $W = 0,824561$, $Z = 0$

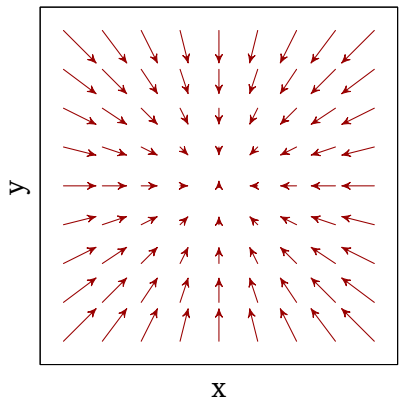

Abbildung 2.5d.:
Translation entlang $g_{z'}$ von der x'-y'-Ebene weg mit $d = 600$, $d^* = 500$ und $f = 30$. $U = 0$, $V = 0$, $W = 1,212766$, $Z = 0$

Abbildung 2.5.: *Charakteristische Flussfelder*

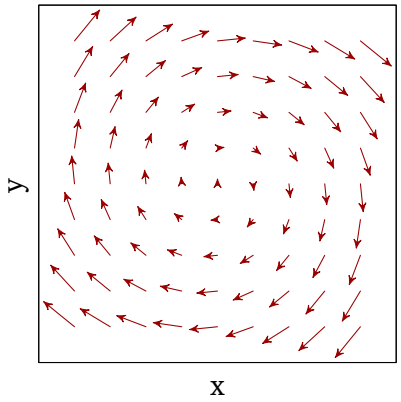

Abbildung 2.5e.:
Rotation um $g_{z'}$ mit $\alpha = 0,2$. $U = 0$, $V = 0$, $W = 1$, $Z = 0,2$

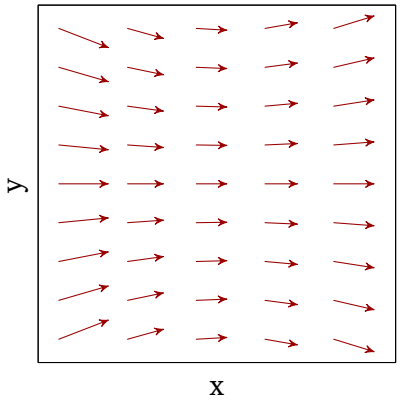

Abbildung 2.5f.:
Rotation um $g_{y'}$ mit $\alpha = 0,15$, $f = 30$. $U = 3,3612812$, $V = 0$, $W = 0,9644543$, $Z = 0$

Abbildung 2.5.: *Charakteristische Flussfelder*

Aus 2.6, 2.7, 2.8, 2.9 und 2.10 erhält man

$$\mathbf{x}^* = \begin{bmatrix} x^* \\ y^* \end{bmatrix} \coloneqq \begin{bmatrix} x + c\Delta x' \\ y \end{bmatrix} \quad \text{mit } c = f/(z - z_0) \ .$$

c ist bei der gesamten Translationsbewegung konstant. Für $\Delta\mathbf{x}$ aus 2.5 folgt daraus:

$$\Delta\mathbf{x} \coloneqq \begin{bmatrix} -c\Delta x' \\ 0 \end{bmatrix} \ . \tag{2.11}$$

Die Translation der Kamera $-c\Delta x'$ entlang der Geraden $g_{x'}$ in der Zeitspanne Δt lässt sich aus 2.4, 2.5 und 2.11 mit einem gegebenen Geschwindigkeitsvektor $\mathbf{v}$ wie folgt berechnen:

$$\mathbf{v} = \left(\mathbf{x} - \left(\mathbf{x} - \begin{bmatrix} -c\Delta x' \\ 0 \end{bmatrix} \right) \right) \cdot \frac{1}{\Delta t} \ .$$

Durch Umstellung erhält man:

$$-c\Delta x' = u\Delta t \ .$$

Will man eine Aussage über die Translation in x-Richtung des gesamten Bildes

treffen, kann man einen Wert $U \mathrel{\widehat{=}} -c\Delta x'$ wie folgt definieren:

$$U \coloneqq \frac{1}{MN} \int_0^M \int_0^N u(\mathbf{x}, t)\Delta t \; dx \; dy \; .$$

Die Translation in y-Richtung aus Abbildung 2.5b berechnet sich analog: $-c\Delta y' = v\Delta t$. Für die Translation des gesamten Bildes in y-Richtung wird:

$$V \coloneqq \frac{1}{MN} \int_0^M \int_0^N v(\mathbf{x}, t)\Delta t \; dx \; dy$$

definiert.

2.3.3.2. Translation in z'-Richtung

Eine Translation in z'-Richtung zeigen Abbildungen 2.5c und 2.5d. Für diesen Fall folgt aus 2.2 und $\mathbf{R}(0)$

$$x^* = f\frac{x' - x'_0}{z' - z'^*_0} + \hat{x} \; , \tag{2.12}$$

$$x = f\frac{x' - x'_0}{z' - z'_0} + \hat{x} \tag{2.13}$$

und aus 2.3

$$y^* = f\frac{y' - y'_0}{z' - z'^*_0} + \hat{y} \; , \tag{2.14}$$

$$y = f\frac{y' - y'_0}{z' - z'_0} + \hat{y} \; . \tag{2.15}$$

Die Verschiebung der Kamera in ihrer Umwelt um $\Delta z'$ verändert wieder das Projektionszentrum der Kamera:

$$(x'^*_0, y'^*_0, z'^*_0) \coloneqq (x'_0, y'_0, z'_0 - \Delta z') \; . \tag{2.16}$$

Durch Umstellung erhält man

$$\mathbf{x}'^* \coloneqq \begin{bmatrix} x' \\ y' \end{bmatrix} s$$

mit

$$s = \frac{z - z_0}{z - {z_0}^*} = \frac{a}{a^*} = \frac{d - f}{d^* - f} \,. \tag{2.17}$$

Für die Berechnung des Faktors s aus einem gegebenen Vektor $\mathbf{v}$, lassen sich die Distanzen von (x^*, y^*) und (x, y) zu $(\hat{x}, \hat{y})$, wie sie in 2.6 abgebildet sind, nutzen. Der Fakter s berechnet sich daraus wie folgt:

$$s = \frac{r^*}{r} \quad \text{mit} \quad r = \|\mathbf{x} - \hat{\mathbf{x}}\| \quad \text{und} \quad r^* = \|\mathbf{x}^* - \hat{\mathbf{x}}\| \,. \tag{2.18}$$

Aus 2.17, 2.18 und 2.5 ergibt sich:

$$\frac{d - f}{d^* - f} = \frac{\|\mathbf{x} - \hat{\mathbf{x}} - \mathbf{v}\Delta t\|}{\|\mathbf{x} - \hat{\mathbf{x}}\|} \,.$$

Wenn f im Vergleich zu d sehr klein ist, gilt:

$$\frac{d}{d^*} \approx \frac{d - f}{d^* - f} \,.$$

Die Translation des gesamten Bildes entlang der Geraden $g_{z'}$ entspricht dem Verhältnis der Durchschnitte von $\|\mathbf{x} - \hat{\mathbf{x}} - \mathbf{v}\Delta t\|$ und $\|\mathbf{x} - \hat{\mathbf{x}}\|$ über alle $\mathbf{x}$:

$$W := \frac{\int_0^M \int_0^N \|\mathbf{x} - \hat{\mathbf{x}} - \mathbf{v}\Delta t\| \, dx \, dy}{\int_0^M \int_0^N \|\mathbf{x} - \hat{\mathbf{x}}\| \, dx \, dy} \,.$$

Es ist also nur möglich, das Verhältnis $\frac{d}{d^*} \approxeq W$ zu bestimmen. Eine genaue Distanzdifferenz Δd lässt sich nur mit einem zusätzlichen Entfernungsmesser errechnen. Das Verhältnis $\frac{d}{d^*}$ ist jedoch sehr aussagekräftig und lässt sich bereits vielfältig anwenden, wie zum Beispiel für das Abschätzen einer Zeit bis zum Auftreffen auf Objekte, die dem Beobachter entgegen kommen [Lee76].

2.3.3.3. Rotation um $g_{z'}$

Eine Rotation der Kamera um die Gerade $g_{z'}$ zeigt Abbildung 2.5e. Das Bild wird mit dem Winkel $-\alpha$ um den Rotationsmittelpunkt $\hat{\mathbf{x}}$ gedreht. Das entspricht einer Drehung der Kamera um α. Für die Rotationsmatrix der Kollinearitätsgleichung be-

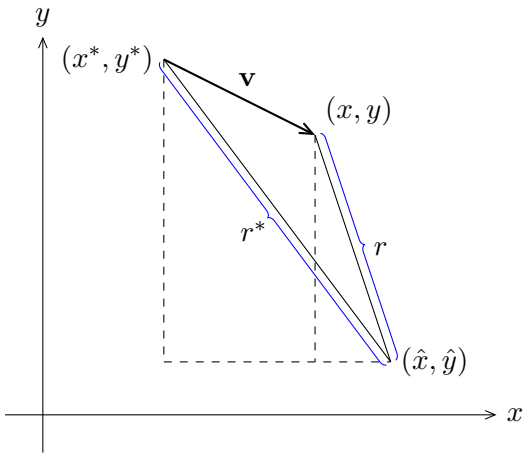

Abbildung 2.6.: *Herleitung von $\frac{d}{d^*}$ für einen gegebenen Vektor* $\mathbf{v}$

deutet das $\mathbf{R} \coloneqq \mathbf{R}_{z'}(\alpha)$. Aus 2.2 folgt daraus

$$x^* = f\frac{(x' - x'_0)\cos\alpha - (y' - y'_0)\sin\alpha}{z' - z'_0} + \hat{x}\ , \tag{2.19}$$

$$x = f\frac{x' - x'_0}{z' - z'_0} + \hat{x} \tag{2.20}$$

und aus 2.3

$$y^* = f\frac{(x' - x'_0)\sin\alpha + (y' - y'_0)\cos\alpha}{z' - z'_0} + \hat{y}\ , \tag{2.21}$$

$$y = f\frac{y' - y'_0}{z' - z'_0} + \hat{y}\ . \tag{2.22}$$

Durch Umstellung erhält man:

$$\mathbf{x}^* \coloneqq \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} (\mathbf{x} - \hat{\mathbf{x}}) + \hat{\mathbf{x}}\ .$$

Abbildung 2.7 veranschaulicht die Berechnung von α für ein gegebenen Flussvektor $\mathbf{v}$:

$$\alpha = \gamma - \beta$$

mit

$$\gamma = \arctan\left(\frac{y - \hat{y}}{x - \hat{x}}\right)$$

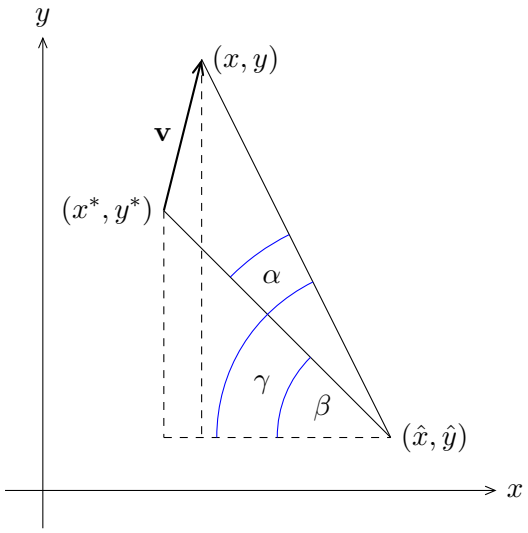

Abbildung 2.7.: *Herleitung von α bei einer Rotation der Kamera um die Gerade $g_{z'}$*

und

$$\beta = \arctan\left(\frac{y^* - \hat{y}}{x^* - \hat{x}}\right) .$$

Für $\mathbf{x}^* = \mathbf{x} - \mathbf{v}\Delta t$ kann die Rotation des gesamten Bildes $Z \mathrel{\widehat{=}} \alpha$ berechnet werden:

$$Z \coloneqq \frac{1}{MN} \int_0^M \int_0^N \arctan\left(\frac{y - \hat{y}}{x - \hat{x}}\right) - \arctan\left(\frac{y - \hat{y} - v(\mathbf{x}, t)\Delta t}{x - \hat{x} - u(\mathbf{x}, t)\Delta t}\right) \, dx \, dy \, .$$

2.3.3.4. Rotation um $g_{y'}$

In Abbildung 2.5f ist das Vektorbild dargestellt, welches durch Rotation der Kamera um die Gerade $g_{y'}$ entsteht. α ist jetzt der Winkel, mit dem die Kamera um diese Gerade gedreht wird und ϕ die Auslenkung zwischen der Senkrechten auf der x'-y'-Ebene durch Punk S. Die Rotation der Kamera wird durch die Matrix $\mathbf{R} \coloneqq \mathbf{R}_{y'}(\alpha)$ beschrieben. Vor der Bewegung ist die Kamera um $\phi - \alpha$ und danach um ϕ ausgelenkt. Aus 2.2 folgt daraus

$$x^* = f\frac{(x' - x'_0)\cos(\phi - \alpha) + (z' - z'_0)\sin(\phi - \alpha)}{(z' - z'_0)\cos(\phi - \alpha) - (x' - x'_0)\sin(\phi - \alpha)} + \hat{x} \, , \tag{2.23}$$

$$x = f\frac{(x' - x'_0)\cos\phi + (z' - z'_0)\sin\phi}{(z' - z'_0)\cos\phi - (x' - x'_0)\sin\phi} + \hat{x} \tag{2.24}$$

und aus 2.3

$$y^* = f\frac{y' - y'_0}{(z' - z'_0)\cos(\phi - \alpha) - (x' - x'_0)\sin(\phi - \alpha)} + \hat{y}\ , \tag{2.25}$$

$$y = f\frac{y' - y'_0}{(z' - z'_0)\cos\phi - (x' - x'_0)\sin\phi} + \hat{y}\ . \tag{2.26}$$

$x' - x'_0$ lässt sich aus Gleichung 2.24 herleiten:

$$x' - x'_0 = (z' - z'_0)\frac{(x - \hat{x})\cos\phi - f\sin\phi}{f\cos\phi + (x - \hat{x})\sin\phi}\ .$$

Damit lässt sich 2.23 zu

$$x^* = f\frac{(x - \hat{x})\cos\alpha - f\sin\alpha}{(x - \hat{x})\sin\alpha + f\cos\alpha} + \hat{x}$$

und 2.25 zu

$$y^* = f\frac{y - \hat{y}}{(x - \hat{x})\sin\alpha + f\cos\alpha} + \hat{y}$$

vereinfachen. Damit ist die Drehung nur abhängig von der Roation α der Kamera.

2.3.4. Approximation komplexer Flussbilder

Als komplexe Felder des optischen Flusses werden solche angenommen, die nicht einer einfachen Translation in x- beziehungsweise y-Richtung, wie in Abbildung 2.5a und 2.5b dargestellt, entsprechen. Zur Unterscheidung, ob es sich bei dem Vektorfeld um andere Bewegungen als die beiden oben genannten Translationen handelt, bedarf es eines dichten Vektorfeldes, bei dem für jeden Pixel ein Geschwindigkeitsvektor vorliegt. Die Bestimmung solch eines Feldes ist jedoch sehr rechenaufwendig und mit der nötigen Genauigkeit in der Praxis kaum möglich. Deshalb ist es wünschenswert, diese Berechnungen in ihrer Komplexität zu verringen. Anacona und Poggio zeigen in ihrer Arbeit [AP93] zwei Detektoranordnungen, die mittels einzelner eindimensionaler Detektoren Divergenz oder Bewegungen in der Ebene erkennen können. Abbildung 2.8 zeigt diese Anordnungen. 2.8a ermöglicht die Erkennung von Bewegungen, die eine Kamera auf einer Ebene parallel zur aufgenommenen Fläche durchführt. Dazu gehört auch die Drehung um eine Senkrechte auf dieser Ebene. Mit der Konstruktion aus 2.8b lässt sich hingegen die Translation entlang der Geraden $g_{z'}$ bestimmen. Führt man beide zusammen, erhält man einen Aufbau, der Translationen (U, V, W) und Rotation (Z) unterscheiden kann. Diese

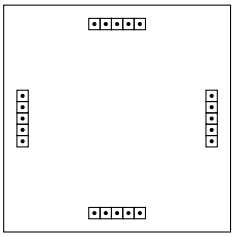

Abbildung 2.8a.: Detektoranordnung zum Erkennen von Translation und Rotation in der Ebene (U, V, Z)

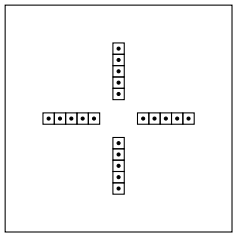

Abbildung 2.8b.: Detektoranordnung zum Erkennen von Divergenz (W)

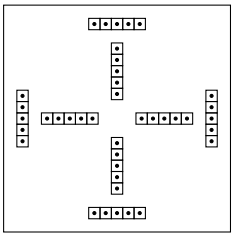

Abbildung 2.8c.: Zusammenführung der Detektoranordnungen zum Erkennen von Translation und Rotation (U, V, W, Z)

Abbildung 2.8.: *Detektoranordnungen zur Bewegungserkennung*

Zusammenführung ist in Abbildung 2.8c zu sehen.

Wie im Experiment in Kapitel 5.4.1 deutlich wird, ist die Leistungsfähigkeit eines Detektors, der lediglich mit eindimensionalen Eingangsdaten arbeitet, kleiner als einer, dessen Eingabe aus zweidimensionalen Daten besteht. Die Anordnung aus Abbildung 2.8c kann durch Zusammenführen von jeweils zwei Zeilendetektoren in eine überführt werden, die anstatt aus acht 1D-Detektoren aus vier 2D-Detektoren besteht.

Für die weiteren Betrachtungen wird ein elementarer Detektor definiert, der in der Lage ist, aus einem Bildausschnitt eines Videos die x- und y-Komponente, also die Größen U und V, des optischen Flusses zu bestimmen. Der Detektor lässt sich als Funktion wie folgt beschreiben:

$$D(\mathbf{x}_0, m, n) := \begin{bmatrix} U(\mathbf{x}_0, m, n) \\ V(\mathbf{x}_0, m, n) \end{bmatrix} ,$$

mit

$$U(\mathbf{x}_0, m, n) = \frac{1}{mn} \int_{x_0}^{x_0+n} \int_{y_0}^{y_0+m} u(\mathbf{x}, t)\Delta t \; dx \; dy$$

und

$$V(\mathbf{x}_0, m, n) = \frac{1}{mn} \int_{x_0}^{x_0+n} \int_{y_0}^{y_0+m} v(\mathbf{x}, t)\Delta t \; dx \; dy \; .$$

$\mathbf{x}_0$ bestimmt die Position des Detektors im Bild und m und n die Größe.

Die Größen U und V der Flussfelder aus Abbildung 2.5a und 2.5b lassen sich mit dem Elementardetektor $D(\mathbf{0}, M, N) = \begin{bmatrix} U & V \end{bmatrix}^T$ trivial bestimmen. W und Z der Felder aus Abbildung 2.5c bis 2.5f können approximiert werden. Eine entsprechende Anordnung von mehreren Elementardetektoren, ist in der Abbildung 2.9 zu sehen. Die Komponenten der Detektoren sind gegeben mit:

$$D_1 = \begin{bmatrix} U_1 \\ V_1 \end{bmatrix}, \qquad D_2 = \begin{bmatrix} U_2 \\ V_2 \end{bmatrix},$$
$$D_3 = \begin{bmatrix} U_3 \\ V_3 \end{bmatrix} \text{ und} \qquad D_4 = \begin{bmatrix} U_4 \\ V_4 \end{bmatrix}.$$

Aus dieser Detektoranordnung lassen sich die Größen U, V, W und Z wie folgt annähern:

$$\begin{aligned} U \mathrel{\hat{=}} U_D &= \frac{U_1 + U_2 + U_3 + U_4}{4}, \\ V \mathrel{\hat{=}} V_D &= \frac{V_1 + V_2 + V_3 + V_4}{4}, \\ W \mathrel{\hat{=}} W_D &= \frac{c - (U_3 - U_1 + V_2 - V_4)}{c}, \\ Z \mathrel{\hat{=}} Z_D &= V_1 - U_2 - V_3 + U_4, \end{aligned}$$

wobei c von dem Abstand der Detektoren zum Mittelpunkt abhängt. Die Werte für U_D, V_D, W_D und Z_D für die Flussbilder aus Abbildung 2.5a bis 2.5f können der Tabelle 2.2 entnommen werden. Die Berechnung erfolgt mit den Werten: $M = N = 40$ und den Detektoren:

$$\begin{aligned} D_1 &:= D(\begin{bmatrix} \frac{80}{3} & \frac{40}{3} \end{bmatrix}^T, \frac{40}{3}, \frac{40}{3}), \\ D_2 &:= D(\begin{bmatrix} \frac{40}{3} & \frac{80}{3} \end{bmatrix}^T, \frac{40}{3}, \frac{40}{3}), \\ D_3 &:= D(\begin{bmatrix} 0 & \frac{40}{3} \end{bmatrix}^T, \frac{40}{3}, \frac{40}{3}), \\ D_4 &:= D(\begin{bmatrix} \frac{40}{3} & 0 \end{bmatrix}^T, \frac{40}{3}, \frac{40}{3}). \end{aligned}$$

Der Tabelle 2.2 ist zu entnehmen, dass die durch die Detektoranordnung approximierten Werte mit den tatsächlichen, genau bestimmten, korrespondieren. Die Erkennung komplexer Flussfelder lässt sich also auf eine Bestimmung des optischen

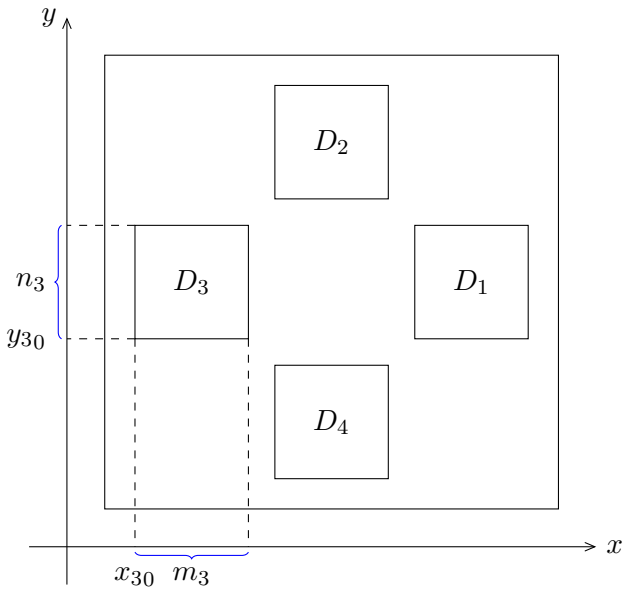

Abbildung 2.9.: *Detektoranordnung zur Appraximation komplexer Flussfelder*

Flusses in einer translatorischen Richtung zurückführen. Da die Berechnung der horizontalen Komponente U eines Elementardetektors analog zu der vertikalen V erfolgt, kann man sich bei den folgenden Betrachtungen also auf die Untersuchung der horizontalen Komponente beschränken. Alle Algorithmen werden daher nur auf die Leistungsfähigkeit bei der Detektion dieser einen Bewegungskomponente untersucht.

2.4. Neuronenmodell

2.4.1. Neuron

Künstliche neuronale Netze dienen dem Zweck, das Verhalten realer biologischer neuronaler Netze zu simulieren und dessen Eigenschaften zu studieren. Da die reale Nervenzelle in ihrer Funktion sehr komplex ist [Tho01], wird diese hier nur vereinfacht modelliert. Derartige künstliche Neuronen gehen auf McCulloch und Pitts [MP43] zurück. Für eine ausführliche Einführung zu diesem Thema kann auf die Arbeit [Hil07] oder auch [Wol07] verwiesen werden.

Die hier genutzen neuronalen Verfahren basieren auf amplitudenkodierten künstlichen Neuronen. In Kürze beschrieben lässt sich das Ausgangssignal eines Neurons

Bewegung der Kamera	U, V, W, Z	U_D, V_D, W_D, Z_D
Translation entlang $g_{x'}$ (s. 2.5a)	$U = 6$ $V = 0$ $W = 1$ $Z = 0$	$U_D = 6$ $V_D = 0$ $W_D = 1$ $Z_D = 0$
Translation entlang $g_{y'}$ (s. 2.5b)	$U = 0$ $V = 6$ $W = 1$ $Z = 0$	$U_D = 0$ $V_D = 6$ $W_D = 1$ $Z_D = 0$
Translation entlang $g_{z'}$ mit $d = 600$, $d^* = 500$ und $f = 30$ (s. 2.5d)	$U = 0$ $V = 0$ $W = 1,212766$ $Z = 0$	$U_D = 0$ $V_D = 0$ $W_D = 1,7$ $Z_D = 0$
Translation entlang $g_{z'}$ mit $d = 500$, $d^* = 600$ und $f = 30$ (s. 2.5c)	$U = 0$ $V = 0$ $W = 0,824561$ $Z = 0$	$U_D = 0$ $V_D = 0$ $W_D = 0,15$ $Z_D = 0$
Rotation um $g_{z'}$ mit $\alpha = 0,2$ (s. 2.5e)	$U = 0$ $V = 0$ $W = 1$ $Z = 0,2$	$U_D = 0$ $V_D = 0$ $W_D = 1,08$ $Z_D = 2,65$
Rotation um $g_{y'}$ mit $\alpha = 0,15$ und $f = 30$ (s. 2.5f)	$U = 5,573622$ $V = 0$ $W = 0,923066$ $Z = 0$	$U_D = 5.072$ $V_D = 0$ $W_D = 0.919$ $Z_D = 0$

Tabelle 2.2.: *Beispiele der Werte U_D, V_D, W_D und Z_D*

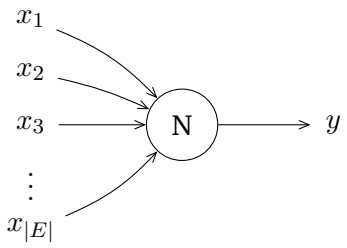

Abbildung 2.10.: *Schematische Darstellung eines* Sigma-Neurons

y durch eine Übergangsfunktion aus dessen Eingangsaktivität a berechnen:

$$y \coloneqq f(a) \ .$$

Mögliche Übergangsfunktionen f zeigt Abbildung 2.11. Alle Eingangssignale x_i des Neurons ergeben durch Wichtung die Eingangsaktivität:

$$a \coloneqq \sum_{i=0}^{|E|-1} w_i x_i \ ,$$

wobei E die Menge der verknüpften Eingangsneuronen ist und w_i die Gewichtungsparameter des entsprechenden Eingangssignals. Da die Eingangswerte aufsummiert werden, wird dieses Neuronenmodell auch als *Sigma-Neuron* bezeichnet. Die grafische Darstellung eines Neurons zeigt Abbildung 2.10.

In seltenen Fällen wird das *Sigma-Pi-Neuron* (vgl. [Wol07]) genutzt. Für die Eingangsaktivität dieses Neuronentyps wird aus der Menge E der Eingangsneuronen eine Menge P_K mit K verschiedenen Teilmengen S_j ($j = 1, \ldots, K$) von E gebildet. Die gewichtete Summe der Produkte der Elemente dieser Teilmengen S_i führt zur Aktivierung:

$$a \coloneqq \sum_{S_j \in P_K} \left(w_j \prod_{i \in S_j} x_i \right) \ .$$

Die Berechnung von y erfolgt daraus analog zum *Sigma-Neuron*.

Mit sogenannten *Sensorneuronen* ist es möglich, physikalische Größen aus der Umwelt in das Netz einzukoppeln. Sie nehmen in der Regel Werte zwischen ± 1 an. Des Weiteren gibt es Neuronen mit einem festen Ausgangswert. Wenn nicht anders gekennzeichnet, wird das *Sigma-* und *Sigma-Pi-Neuron* in dieser Arbeit mit dem Tangens Hyperbolicus als Übergangsfunktion genutzt.

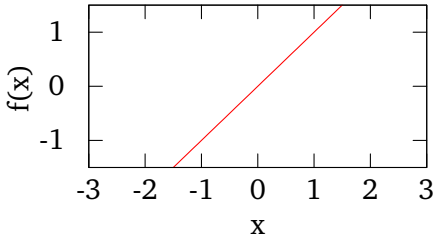

Abbildung 2.11a.:
Identitätsfunktion

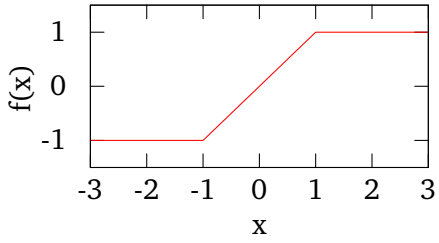

Abbildung 2.11b.:
Beschränkte Identitätsfunktion

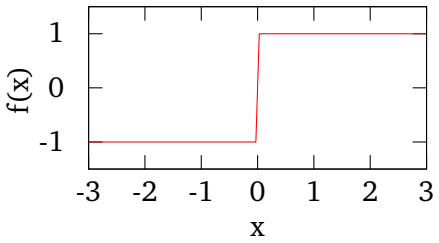

Abbildung 2.11c.:
Sprungfunktion

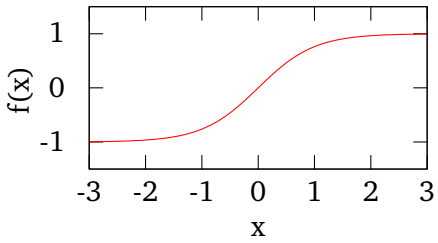

Abbildung 2.11d.:
Tangens Hyperbolicus

Abbildung 2.11.: *Auswahl an Übergangsfunktionen f eines Neurons*

2.4.2. Neuronales Netz

Verknüpft man mehrere Neuronen, erhält man ein zusammenhängendes neuronales Netz. Dieses wird in dieser Arbeit zeitdiskret modelliert.

Die Neuronen des Netzes werden durch den Vektor $\mathbf{x} = \begin{bmatrix} x_0 & \dots & x_{R-1} \end{bmatrix}^T$ repräsentiert, wobei R die Anzahl der Neuronen in dem Netz ist. Jede gerichtete Verknüpfung zwischen zwei Neuronen wird durch ein Gewicht w_{ij} repräsentiert, wobei der Index i angibt, dass die Verknüpfung von Neuron x_i kommt und zu Neuron x_j geht. Keine Eingangsverknüpfung von x_i zu x_j ist gleichbedeutend mit $w_{ij} = 0$. Alle Gewichte w_{ij} werden in der $R \times R$-Matrix $\mathbf{W} := ((w_{ij}))$ mit $0 \leq i, j < R$ gefasst. Die Ausgänge aller Neuronen eines Netzes mit Sigma-Neuronen zu einem Zeitpunkt $t + 1$ kann damit aus den Ausgangssignalen $\mathbf{x}$ zum Zeitpunkt t berechnet werden:

$$\mathbf{x}^{(t+1)} := \mathbf{y}^{(t+1)} := f(\mathbf{W}\mathbf{x}^{(t)}) \, .$$

Durch diese Berechnugsvorschrift sind auch Rekurrenzen möglich, die von den vorgestellten Verfahren ebenfalls genutzt werden.

3. Algorithmen zur Vorverarbeitung des Bildmaterials

Algorithmen, die Bilder verarbeiten, nutzen in der Regel besondere Eigenschaften des Bildes. Zu diesen Eigenschaften gehören Ausprägung oder Anzahl von Kanten, Stetigkeit oder charakteristische Bildmerkmale, Kontrast etc. Diese Eigenschaften lassen sich durch Vorverarbeitung gezielt verstärken oder unerwünschte Effekte ausblenden. Wie sich zeigen wird, hängt die Qualität der Ausgabe zum Teil sehr stark von dieser Vorverarbeitung ab und muss daher in die Untersuchungen einfließen. Im Folgenden werden einige Verfahren vorgestellt, deren Nutzung sich zum Teil intuitiv aus dem Prinzip der Algorithmen ergibt oder grundlegend bei der Bildverarbeitung genutzt werden [Dav05] [Hab91].

3.1. Farbinformationen

Bei einem Algorithmus muss entschieden werden, ob Farbinformationen wesentlich zur Leistungsfähigkeit beitragen. In der Regel spielt Farbe bei der Erkennung von Objekten oder der Bestimmung ihrer Eigenschaften eine wesentliche Rolle. Geht es also nicht um derartige Differenzierungen, wie es bei den meisten in Kapitel 4 beschriebenen Verfahren der Fall ist, reduziert die alleinige Betrachtung der Grauwerte die Komplexität deutlich, ohne die Qualität der Ausgabe zu verringern. Grundsätzlich wird das Graubild durch Verrechnung der Farbkanäle bestimmt:

$$I(\mathbf{x}, t) \coloneqq f(I_r(\mathbf{x}, t), I_g(\mathbf{x}, t), I_b(\mathbf{x}, t)) \ ,$$

wobei I_r, I_g und I_b, wie in Kapitel 2.2.2 beschrieben, die Intensitätsfunktionen des entsprechenden Rot-, Grün- und Blaukanals sind. Die Funktion f ist allgemein definiert durch:

$$f(I_r, I_g, I_b) = rI_r + sI_r + tI_b \ ,$$

mit $0 \leq r, s, t \leq 1$ und $r + s + t = 1$. Ein trivialer Weg zur Reduktion auf Grauwerte ist es, sämtliche Berechnungen nur auf einem Farbkanal durchzuführen. Für die Koeffizienten r, s und t bedeutet das:

$$r = 1 \text{ und } s = t = 0 \,,$$
$$s = 1 \text{ und } r = t = 0 \text{ oder}$$
$$t = 1 \text{ und } r = s = 0 \,.$$

Der Vorteil bei diesem Verfahren ist, dass es keine zusätzlichen Rechnungen erfordert und damit sehr schnell ist. Häufig führt es allein auch schon zu guten Ergebnissen. In Abhängigkeit der Umwelt und der Art der Farbkanalaufteilung kann man zwischen den verschiedenen Kanälen rot, grün, oder blau wählen.

Eine weitere Möglichkeit ist die Gewichtung aller Farbkanäle, also $0 < r, s, t$. Sollen alle Kanäle in gleichem Maße gewichtet werden, bedeutet das für die Koeffizienten $r = s = t = \frac{1}{3}$.

3.2. Schwellwert

Häufig beeinflusst die Helligkeit eines Bilders oder dessen Änderung über die Zeit, die Effizienz eines Algorithmus. Eine Lösung wäre, die Intensitätsverteilung des Bildes zu normieren. Ein mögliches Ziel der Normierung könnte sein, dass das Bild den vollen Intensitätsumfang ausnutzt, also die minimal auftretende Helligkeit im Bild tatsächlich der kleinsten darstellbaren Intensität I_{min} entspricht. Für das Maximum der auftretenden Helligkeit und der maximal darstellbaren Helligkeit I_{max} gilt das gleiche. Der Mittelwert aller Intensitäten soll sich dabei genau bei der Hälfte des Werteumfangs befinden. Um aus dem Eingangsbild I das normierte I_n zu bestimmen, existiert eine Übergangsfunktion:

$$I_n(\mathbf{x}, t) \coloneqq f(I(\mathbf{x}, t)) \,.$$

Abbildung 3.1a zeigt das Histogramm eines nicht normierten Eingangsbildes und 3.1c das gleiche Bild, nach den oben genannten Kriterien, normiert. Als Übergangs-

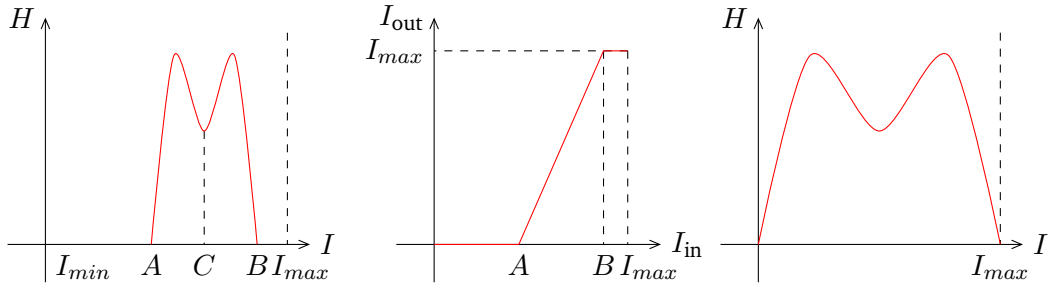

Abbildung 3.1a.: Histogramm des Ausgangsbildes

Abbildung 3.1b.: Übergangsfunktion

Abbildung 3.1c.: Histogramm des Ausgangsbildes

Abbildung 3.1.: *Histogrammbeispiel mit linearer Übergangsfunktion ($I_{min} = 0$)*

funktion genügt in diesem Falle die Funktion

$$f(I) = \begin{cases} 0 & I \leq A \\ I\frac{B-A}{I_{max}} - \frac{AB-A^2}{I_{max}} & A < I < B \\ I_{max} & I \geq B \end{cases} \quad ,$$

wie sie in Abbildung 3.1b zu sehen ist. Nicht immer lassen sich diese Anforderungen durch eine lineare Funktion umsetzen, was das Finden der entsprechenden Koeffizienten stark erschwert. Das ist der Fall, wenn, anders als im Beispiel, das Ausgangshistogramm nicht symmetrisch ist.

Möchte man die Komplexität beim Auffinden der nichtlinearen Übergangsfunktion umgehen, kann das Eingangsbild durch einen Schwellwert in ein Binärbild überführt werden. Die Übergangsfunktion wird dabei deutlich vereinfacht:

$$f(I) = \begin{cases} 0 & I \leq T \\ 1 & I > T \end{cases} \quad .$$

Der eigentliche Aufwand reduziert sich auf die Bestimmung des Schwellwertes T. Dabei kann die Umrechnung in ein Binärbild durch geeignete Berechnung des Schwellwertes verschiedene Funktionen erfüllen. Ein Bild wird in charakteristisches Auftreten von Intensitätshäufungen geteilt, wenn der Schwellwert auf gut ausgeprägte Minima im Histogramm gesetzt wird. In Abbildung 3.1a wäre das beispielhaft das Minimum an der Stelle C. Das angestrebete Ziel, von der Bildhelligkeit unabhängig

zu werden, erreicht man in bester Weise durch Belegung des Schwellwertes T mit dem durchschnittlichen Helligkeitswert des Bildes.

Das Schwellwertverfahren ist wenig rechenintensiv und führt dennoch zu einer guten Unabhängigkeit von Helligkeitsschwankungen im Ausgangsbild.

3.3. Differenzierung

3.4. Weichzeichnung

Ein unerwünschtes Verhalten in Videorohdaten ist das Rauschen. Dieses lässt sich unterdrücken, indem jeder Pixel durch einen Mittelwert aus seiner Umgebung ersetzt wird. Ein weiterer Effekt dieses Verfahrens ist, dass das Bild weichgezeichnet wird. Das ist insbesondere für Algorithmen, die auf eine Stetigkeit des Bildmaterials angewiesen sind, von besonderem Vorteil.

Zur Berechnung solcher Filter eignet sich die diskrete Bildrepäsentation aus 2.2.1. Für ein gegebenes Bild $\mathbf{P}$ ergibt sich durch Anwendung eines Filters F das resultierende Bild $\mathbf{P}^{(F)}$:

$$\mathbf{P}^{(F)} := F(\mathbf{P}, \mathbf{M})$$

mit

$$F(\mathbf{P}, \mathbf{M}^{R\times S}) = \left(\left(\frac{1}{|\mathbf{M}^{R\times S}|_1} < \mathbf{U}_{ij}^{R\times S}, \mathbf{M}^{R\times S} > \right)\right) ,$$

$$\text{für } 1 \leq i \leq M \text{ und } 1 \leq j \leq N \ ,$$

wobei $\mathbf{U}_{ij}^{R\times S}$ eine $R \times S$-Umgebung ist, die Pixel p_{ij} als Zentrum hat, und $\mathbf{M}^{R\times S}$ eine für den Filter charakteristische Matrix. Nachfolgend werden die Dimensionen der Matrizen $\mathbf{M}$ und $\mathbf{U}$ nicht mehr notiert und immer fest als $R \times S$ vorausgesetzt. Die Nachbarschaft des Pixels ist definiert durch:

$$\mathbf{U}_{ij} := \left(\left(p_{i+r-\lfloor \frac{R+1}{2} \rfloor,\ j+s-\lfloor \frac{S+1}{2} \rfloor} \right)\right)$$

$$\text{für } 1 \leq r \leq R \text{ und } 1 \leq s \leq S$$

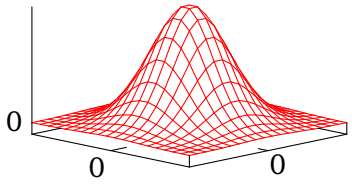

Abbildung 3.2a.:
Gaußverteilung

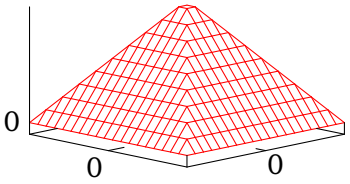

Abbildung 3.2b.:
Dreiecksverteilung

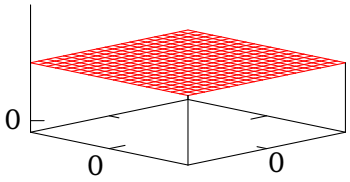

Abbildung 3.2c.:
Gleichverteilung

Abbildung 3.2.: *Qualitative Verteilungen aus denen sich die Größen für die Elemente der Matrix* $\mathbf{M}$ *ergeben*

oder anders geschrieben:

$$\mathbf{U}_{ij} := \begin{bmatrix} p_{i-R_l,\,j-S_l} & \cdots & p_{i-R_l,\,j-1} & p_{i-R_l,\,j} & p_{i-R_l,\,j+1} & \cdots & p_{i-R_l,\,j+S_h} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \iddots & \vdots \\ p_{i-1,\,j-S_l} & \cdots & p_{i+1,\,j-1} & p_{i-1,\,j} & p_{i-1,\,j+1} & \cdots & p_{i-1,\,j+S_h} \\ p_{i,\,j-S_l} & \cdots & p_{i,\,j-1} & p_{ij} & p_{i,\,j+1} & \cdots & p_{i,\,j+S_h} \\ p_{i+1,\,j-S_l} & \cdots & p_{i+1,\,j-1} & p_{i+1,\,j} & p_{i+1,\,j+1} & \cdots & p_{i+1,\,j+S_h} \\ \vdots & \iddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{i+R_h,\,j-S_l} & \cdots & p_{i+R_h,\,j-1} & p_{i+R_h,\,j} & p_{i+R_h,\,j+1} & \cdots & p_{i+R_h,\,j+S_h} \end{bmatrix}$$

mit $R_l = \lfloor \frac{R-1}{2} \rfloor$, $R_h = \lceil \frac{R-1}{2} \rceil$, $S_l = \lfloor \frac{S-1}{2} \rfloor$ und $S_h = \lceil \frac{S-1}{2} \rceil$. Hier wird vorerst angenommen, dass auch die Pixel außerhalb der Bildgrenzen $1 \leq i \leq M$ und $1 \leq j \leq N$ definiert sind. Eine differenziertere Betrachtung dazu erfolgt in Kapitel 3.4.

Die Matrix $\mathbf{M}$ beschreibt also eine Verteilung innerhalb einer Umgebung, mit der die Pixel des Ursprungsbildes zum gefilterten Bild verrechnet werden. Für diese Filtermatrix lassen sich viele Verteilungen konstruieren. Abbildung 3.2 zeigt als Beispiel eine Gauß-, Dreiecks- und Gleichverteilung. Im Folgenden wird auf die Gleich- und Gaußverteilung näher eingegangen.

Durchschnitt

Der Durchschnitt, und damit die Gleichverteilung in der Matrix $\mathbf{M}$, über alle Pixel in einer $R \times S$-Umgebung ist durch die Matrix

$$\mathbf{M}_M \coloneqq \Big(\Big(m_{ij}\Big)\Big)$$
$$\text{mit } m_{ij} = 1$$
$$\text{für } 1 \leq i \leq R \text{ und } 1 \leq j \leq S$$

gegeben.

Gaußfilter

Werden die Pixel in der Umgebung entsprechend einer Standardnormalverteilung herangezogen, spricht man von einem Gaußfilter. Die Verteilung wird von Matrix $\mathbf{M}_G$:

$$\mathbf{M}_G \coloneqq \Big(\Big(e^{-\frac{1}{\sigma}\left((\frac{R+1}{2}-i)^2+(\frac{S+1}{2}-j)^2\right)}\Big)\Big)$$
$$\text{für } 1 \leq i \leq R \text{ und } 1 \leq j \leq S$$

repräsentiert. Das Maximum der Verteilung ist auf eins normiert. Mit σ kann die Varianz der Verteilung auf die Umgebungsgröße angepasst werden. Abbildung 3.2a veranschaulicht diese Verteilung. Die Matrizen $M_G^{3\times3}$ ($\sigma = 0.5$) bzw. $M_G^{5\times5}$ ($\sigma = 2$) zeigen die Verteilung für die entsprechenden Umgebungen:

$$\mathbf{M}_G^{3\times3} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 10 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \mathbf{M}_G^{5\times5} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 4 & 6 & 4 & 1 \\ 1 & 6 & 10 & 6 & 1 \\ 1 & 4 & 6 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \quad \text{(in Zehnteln)}.$$

Randbetrachtung

An den Rändern eines Bildes ist nicht jeder Pixel einer Umgebung definiert. In diesem Fall muss eine gesonderte Betrachtung durchgeführt werden. Sie wird bei allen Algorithmen eine Rolle spielen, die auf eine Umgebung eines Pixels zurückgreifen und eben auch jene Pixel an den Rändern des Bildes für die Berechnungen nutzen. Drei einfache Strategien zur Begegnung des Problems sind:

1. Auffüllen aller nicht definierten Pixel mit dem minimal möglichen Intensitätswert.
2. Nicht definierte Pixel werden mit dem maximal möglichen Intensitätswert gefüllt.
3. Alle nicht definierten Pixel werden aus der Umgebungsbetrachtung herausgelassen.

Die Definition für die Umgebung $\mathbf{U}_{ij}$ mit den Elemtenten u_{rs} aus Kapitel 3.4 lässt sich durch die Randbetrachtung erweitern auf:

$$\mathbf{U}_{ij}^{(R)} \coloneqq \Big(\Big(u_{rs}^{(R)}(i,j) \Big) \Big)$$

$$\text{für } 1 \leq r \leq R \text{ und } 1 \leq s \leq S$$

Fall 1 führt bei einer Intensitätswertemenge von $[I_{min} : I_{max}]$ zu folgender Funktion:

$$u_{rs}^{(R_1)}(i,j) = \begin{cases} I_{min} & i + r - \left\lfloor \frac{R+1}{2} \right\rfloor \notin \{1 \dots M\} \ \vee \ j + s - \left\lfloor \frac{S+1}{2} \right\rfloor \notin \{1 \dots N\} \\ u_{rs} & sonst \end{cases}$$

und Fall 2 zu:

$$u_{rs}^{(R_2)}(i,j) = \begin{cases} I_{max} & i + r - \left\lfloor \frac{R+1}{2} \right\rfloor \notin \{1 \dots M\} \ \vee \ j + s - \left\lfloor \frac{S+1}{2} \right\rfloor \notin \{1 \dots N\} \\ u_{rs} & sonst \end{cases} .$$

Bei Fall 3 kann eine der Umgebung $\mathbf{U}_{ij}^{(R_1)}$, $\mathbf{U}_{ij}^{(R_2)}$ genutzt werden. Der wesentliche Schritt erfolgt aber durch Anpassung der Matrix $\mathbf{M}$ des Filters F. Durch die Berücksichtigung der Ränder wird $\mathbf{M}^{R \times S}$ mit den Elementen m_{rs} neu definiert. Sie ist jetzt nicht mehr für jeden Pixel identisch, sondern von dessen Position abhängig:

$$\mathbf{M}_{ij}^{(R)} \coloneqq \Big(\Big(m_{rs}^{(R)}(i,j) \Big) \Big)$$

$$\text{für } 1 \leq r \leq R \text{ und } 1 \leq s \leq S$$

mit

$$m_{rs}^{(R)}(i,j) = \begin{cases} 0 & i + r - \left\lfloor \frac{R+1}{2} \right\rfloor \notin \{1 \dots M\} \ \vee \ j + s - \left\lfloor \frac{S+1}{2} \right\rfloor \notin \{1 \dots N\} \\ m_{rs} & sonst \end{cases} .$$

3.5. Überführung in Zeilen- und Spaltenvektoren

Kapitel 2.3.4 hat gezeigt, dass horizontale und vertikale Komponenten des optischen Flusses für weitere Berechnungen mitunter ausreichend sind. Dieses Wissen legt nahe, die Bildinformationen vor dem eigentlichen Algorithmus zur Bestimmung des Flusses auf diese Komponenten zu reduzieren, wodurch der Rechenaufwand deutlich verringert wird. Ein Bild $\mathbf{P}$ der Dimension $M \times N$ wird demnach von zwei Vektoren $\mathbf{p}_x$ und $\mathbf{p}_y$ repräsentiert, wobei $\mathbf{p}_x$ die Dimension N und $\mathbf{p}_y$ M besitzt. $\mathbf{p}_x$ und $\mathbf{p}_y$ werden wie folgt gebildet:

$$\mathbf{p}_x = \frac{1}{M} \begin{bmatrix} \sum_{i=0}^{M} p_{i1} \\ \sum_{i=0}^{M} p_{i2} \\ \vdots \\ \sum_{i=0}^{M} p_{iN} \end{bmatrix}, \quad \mathbf{p}_y = \frac{1}{N} \begin{bmatrix} \sum_{j=0}^{N} p_{1j} \\ \sum_{j=0}^{N} p_{2j} \\ \vdots \\ \sum_{j=0}^{N} p_{Mj} \end{bmatrix}.$$

Die oben in zweidimensionaler Form definierten Filter lassen sich auf diese Vektoren in einer entsprechenden eindimensionalen Definition anwenden. Die Definition des Filters F aus Kapitel 3.4 ändert sich zu F_{1D}[1]:

$$\mathbf{p}^{(F_{1D})} \coloneqq F_{1D}(\mathbf{p}, \mathbf{v})$$

mit

$$F_{1D}(\mathbf{p}, \mathbf{v}^R) = \frac{1}{|\mathbf{v}^R|_1} \left(\left(\mathbf{u}_i^R \cdot \mathbf{v}^R \right) \right),$$

$$\text{für } 1 \leq i \leq \text{Dimension von } \mathbf{p}\,.$$

$\mathbf{v}^R$ ist der für den Filter charakteristische Vektor und $\mathbf{u}_i^R$ die Umgebung mit p_i als

[1] Mit $1D$ wird die Funktion als eindimensionale Variante von F gekennzeichnet.

Zentrum und der Größe R:

$$\mathbf{u}_i^R := \begin{bmatrix} p_{i-R_l} \\ \vdots \\ p_{i-1} \\ p_i \\ p_{i+1} \\ \vdots \\ p_{i+R_h} \end{bmatrix},$$

mit $R_l = \lfloor \frac{R-1}{2} \rfloor$ und $R_h = \lceil \frac{R-1}{2} \rceil$. Auch hier gilt die Annahme, dass der Vektor $\mathbf{p}$ über die Bildgrenze hinaus noch mit Werten definiert ist. Die Betrachtungen für den Rand lassen sich analog zu Abschnitt 3.4 für diese eindimensionale Form herleiten.

4. Algorithmen zur Bestimmung des optischen Flusses

4.1. Auswahl der Algorithmen

Es gibt eine Vielzahl Algorithmen, die in der Lage sind den optischen Fluss zu bestimmen. In dieser Arbeit können natürlich nicht alle untersucht werden. Daher werden nur diese hier behandelt, die sich bereits als leistungsfähig erwiesen haben. Ein Vergleich verschiedener Verfahren stellen Barron, Fleet und Beauchemin in ihrer Arbeit *Performance of Optical Flow Techniques* [BFB94] an. Sie untersuchen die Algorithmen anhand verschiedener Testbildsequenzen. Diese teilen sich auf in synthetische und reale Daten. Die snythetischen Daten bestehen aus zwei sinusioden Mustern und zwei schwarzen Quadraten auf weißem Grund mit je einem unterschiedlichen festen optischen Fluss für das gesamte Bild. [1] Als reale Bildsequenzen wird auf eine Auswahl verschiedener Szenen zurückgegriffen. Darunter befinden sich Translationen und Divergenzen einer Bildfolge, sowie komplexe Überflüge oder abgefilmtes Verkehrsgeschehen. Kapitel 2.3.4 hat gezeigt, dass komplexere Flussfelder auf translatorische Komponenten reduziert werden können. Im Fokus steht hier auch die reale Anwendbarkeit der Algorithmen. Aus diesem Grund werden aus der Publikation [BFB94] die Ergebnisse der realen Translationsszenen als Grundlage für die Auswahl der hier untersuchten Verfahren genutzt[2]. Die Ergebnisse der Untersuchungen werden in dieser Veröffentlichung anhand der Größen *durchschnittlicher Fehler*, *Standardabweichung* und *Dichte* verglichen. Der Fehler ist die Winkeldifferenz des tatsächlichen dreidimensionalen Geschwindigkeitsvektors $\begin{bmatrix} u & v & \Delta t \end{bmatrix}^T$ in Raum und Zeit und dem durch den Algorithmus bestimmten Vektor. Die *Standardabweichung* bezieht sich auf diesen Fehler. Mit der Größe *Dichte* wird angegeben, wie-

[1] Sinusoid 1: Wellenlänge: 6 Pixel; optischer Fluss: $\mathbf{v} = \begin{bmatrix} 1,585 & 0,863 \end{bmatrix}^T$ in Pixel pro Frame
Sinusoid 2: Wellenlänge: 16 Pixel; optischer Fluss: $\mathbf{v} = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$ in Pixel pro Frame
Square 1: Kantenlänge: 40 Pixel; optischer Fluss: $\mathbf{v} = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$ in Pixel pro Frame
Square 2: Kantenlänge: 40 Pixel; weichgezeichnet; optischer Fluss: $\mathbf{v} = \begin{bmatrix} \frac{4}{3} & \frac{4}{3} \end{bmatrix}^T$ in Pixel pro Frame

[2] Die realen Bildsequenzen mit Translation sind im Paper mit *SRI Trees* und *Translation Tree* benannt.

viele Bildpunkte prozentual tatsächlich mit einem durch den Algorithmus bestimmten Geschwindigkeitsvektor belegt werden können. Da bei den Einzeldetektoren aus Kapitel 2.3.4 grundsätzlich die genauen einzelnen Werte innerhalb des Detektors ohne Belang sind, muss die *Standardabweichung* und die *Dichte* zum Vergleich nicht mit herangezogen werden. Der Ergebnisstabelle *Summary of the Translating Tree 2D Velocity Results* ist daher zu entnehmen, dass die Algorithemen *Fleet and Jepson*, *Uras et al.* und *Lucas and Kanade* die besten Ergebnisse liefern. Diese sollen mit bioinspirierten oder neuronal umgesetzten Algorithmen verglichen werden. Dazu gehören der Reichardt-Detektor [Rei61] und die ein- oder zweilagigen Neuronenzeilen aus der Arbeit von Wollstein [Wol07]. Des Weiteren sollen die Möglichkeiten der *Slow-Feature-Analysis* [WS02] in Bezug auf die Detektion des optischen Flusses untersucht werden.

4.2. Gradientenbasierte Verfahren

Aus der *Helligkeitsbeständigkeits*-Bedingung aus Kapitel 2.3.2 folgt der Zusammenhang:

$$I(\mathbf{x}, t) = I(\mathbf{x} + \delta\mathbf{x}, t + \delta t) \; . \tag{4.1}$$

Durch Umschreibung der rechten Seite der Gleichung mittels Taylor-Entwicklung erhält man:

$$I(\mathbf{x} + \delta\mathbf{x}, t + \delta t) = I(\mathbf{x}, t) + \nabla I \cdot \delta\mathbf{x} + I_t \delta t + R_2 \; , \tag{4.2}$$

wobei

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$

ein Vektor der partiellen Ableitungen von I nach x und y ist und R_2 ein Rest höherer Ordnung. Aus Formel 4.1 und 4.2 folgt mit hinreichend großer Genauigkeit:

$$\nabla I \cdot \delta\mathbf{x} + I_t \delta t = 0 \; . \tag{4.3}$$

Für $u(\mathbf{x}, t) = \frac{dx}{dt}$ und $v(\mathbf{x}, t) = \frac{dy}{dt}$ erhält man aus 4.3:

$$\nabla I \cdot \mathbf{v} + I_t = 0 \; . \tag{4.4}$$

Die Gleichung 4.4 hat mehrere Lösungen, die im u-v-Diagramm, wie in Abbildung 4.1 grafisch dargestellt, auf einer Geraden liegen. Ein gradientenbasiertes Verfahren zur Bestimmung des optischen Flusses muss also zusätzliche Bedingungen stellen,

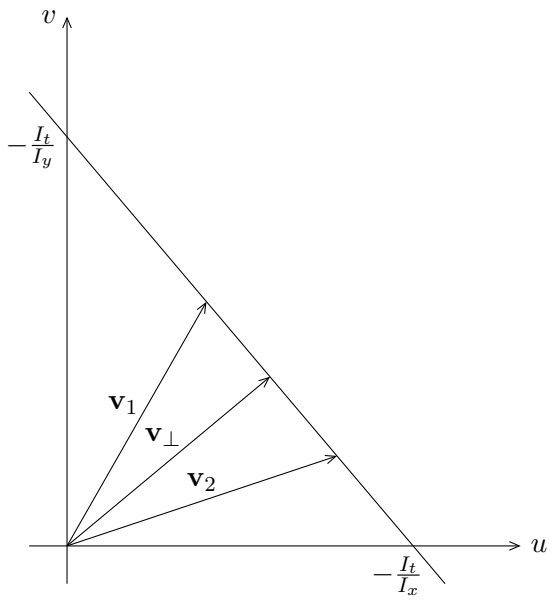

Abbildung 4.1.: *Lösungen für die Geschwindigkeitskomponenten u und v des optischen Flusses*

um ein eindeutiges Ergebnis für $\mathbf{v}$ berechnen zu können. Die Länge, also die euklidische Norm, des Vektors $\mathbf{v}$ zu minimieren, ist eine mögliche Randbedingung. Damit erhält man $\mathbf{v}_\perp$ als einzige Lösung. Die Gerade

$$v = \frac{I_y}{I_x} u \tag{4.5}$$

ist senkrecht zur der aus Gleichung 4.4. $\mathbf{v}_\perp$ ist damit der Schnittpunkt der Geraden 4.4 und 4.5:

$$\mathbf{v}_\perp \coloneqq -\frac{I_t \nabla I}{|\nabla I|^2}$$

4.2.1. First-Order

Mit dem Begriff *First-Order* wird hier das Verfahren bezeichnet, welches den optischen Fluss eines Bildes aus einem Durchschnitt aller $\mathbf{v}_\perp(\mathbf{x}, t)$ dieses Bildes bestimmt:

$$\mathbf{v_P} \coloneqq \frac{1}{MN} \int_0^M \int_0^N \mathbf{v}_\perp(\mathbf{x}, t)\, dx\, dy\ .$$

4.2.2. Zeilen-First-Order

Der gradientenbasierte Ansatz aus Kapitel 4.2 lässt sich einfach für eine eindimensionale Intensitätsfunktion $I(x,t)$ ändern. Aus dem Ansatz

$$I(x,t) = I(x+\delta x, t+\delta t)$$

lässt sich analog zu dem zweidimensionalen Fall die Geschwindigkeit

$$u = -\frac{I_t}{I_x}$$

herleiten. Die Geschwindigkeitsinformation der gesamten Zeile wird durch den Durchschnitt aller $u(x,t)$ zu einem Zeitpunkt t bestimmt.

4.2.3. Verfahren von Horn und Schunck

Horn und Schunck haben in ihrer Arbeit [HS80] eine weitere zusätzliche Bedingung gestellt, die es ermöglicht, die Gleichung 4.4 eindeutig zu lösen. Diese Bedingung besagt, dass sich die Geschwindigkeitsvektoren in ihrem Betrag und ihrer Richtung, in einem räumlich begrenzten, kleinen Bereich im Bild nur wenig unterscheiden. Dieser Zusammenhang wird auch häufig als Glattheit bezeichnet und kann für einen Pixel wie folgt formuliert werden:

$$E_1 = |\nabla \mathbf{v}|^2 \ .$$

Die zweite Bedingung ist, dass der Geschwindigkeitsvektor eines Pixels möglichst nah an der Geraden 4.4 liegt:

$$E_2 = \nabla I \cdot \mathbf{v} + I_t \ .$$

E_1 und E_2 müssen für das gesamte Bild minimiert werden. Fasst man beide Bedingungen zusammen erhält man:

$$E = \int_0^M \int_0^N \nabla I \cdot \mathbf{v}(\mathbf{x},t) + I_t + \alpha |\nabla \mathbf{v}(\mathbf{x},t)|^2 \, dx \, dy \ ,$$

wobei $\alpha \in \mathbb{R}$ und $\alpha > 0$ ein Parameter ist, der angibt wie glatt der Fluss ist. Diese Bedingung kann für das Vektorfeld eines Zeitpunktes t iterativ gelöst werden. Auf

die genaue Herleitung dieses Verfahrens wird hier verzichtet, da dieser Algorithmus, wie in Kapitel 4.1 beschrieben, zu keinen besseren Ergebnissen führt. Horn und Schunck haben für diesen Algorithmus eine Durchschnittsbildung über einen $2 \times 2 \times 2$-Block in Raum und Zeit des Videos vorgeschlagen. Diese Maßnahme könnte den *First-Order*-Algorithmus ebenfalls verbessern.

4.2.4. Verfahren von Lucas und Kanade

Lucas und Kanade schlagen in ihrer Arbeit [LK81] eine weitere Bedingung für die Lösung des Gleichungssystems 4.4 vor. Diese besagt, dass der Fluss in einer Umgebung um den Pixel konstant ist. Betrachtet man eine beliebig geformte Umgebung mit der Größe n, lässt sich die Gleichung 4.4 zu einem überbestimmten Gleichungssystem mit n Gleichungen erweitern:

$$\begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \vdots & \vdots \\ I_{x_n} & I_{y_n} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \vdots \\ -I_{t_n} \end{bmatrix} . \tag{4.6}$$

Die Kurzschreibweise I_{x_i} ist äquivalent zu $\frac{\delta I(\mathbf{x}_i,t)}{\delta x}$, analog für I_y und I_t. Die Orte $\mathbf{x}_i$ gehören zur gewählten Umgebung Ω. Dieses System entspricht der Form:

$$\mathbf{A}\mathbf{v} = \mathbf{b}$$

mit

$$\mathbf{b} = \begin{bmatrix} -I_{t_1} & \dots & -I_{t_n} \end{bmatrix}^T \quad \text{und} \quad \mathbf{A} = \begin{bmatrix} I_{x_1} & \dots & I_{x_n} \\ I_{y_1} & \dots & I_{y_n} \end{bmatrix}^T .$$

Das Gleichungssystem lässt sich lösen, wenn die Bedingung gestellt wird, dass die Summe der Fehlerquadrate

$$\sum_{\mathbf{x} \in \Omega} \left(\nabla I(\mathbf{x}, t)\mathbf{v} + I_t(\mathbf{x}, t) \right)^2$$

minimal wird [Luc84]. Die Lösung für $\mathbf{v}$ lässt sich daraus wie folgt berechnen:

$$\mathbf{A}^T\mathbf{A}\mathbf{v} = \mathbf{A}^T(-\mathbf{b})$$

und

$$\mathbf{v} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T(-\mathbf{b}) \quad .$$

Wandelt man das System 4.6 auf diese Weise um, erhält man:

$$\mathbf{v} = \begin{bmatrix} \sum I_{x_i}^2 & \sum I_{x_i} I_{y_i} \\ \sum I_{x_i} I_{y_i} & \sum I_{x_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_{x_i} I_{t_i} \\ -\sum I_{y_i} I_{t_i} \end{bmatrix} .$$

Die Summen laufen jeweils über $i = 1 \ldots n$.

Erweitern lässt sich dieser Algorithmus, indem für die Umgebung des Pixels eine Fensterfunktion $W(\mathbf{x})$ definiert wird, die die Gewichtung der Werte in der Umgebung darstellt. Zu dieser Funktion existiert eine Matrix $\mathbf{W}^{n\times n}$, wie folgt:

$$\mathbf{W} = \Big(\Big(w_{ij} \Big)\Big) ,$$

wobei:

$$w_{ij} = \begin{cases} W(\mathbf{x}_i) & i = j \\ 0 & sonst \end{cases} ,$$

für i und j von $1 \ldots n$. Zu minimieren ist dann

$$\sum_{\mathbf{x}\in\Omega} \left(W(\mathbf{x}) \left(\nabla I(\mathbf{x},t)\mathbf{v} + I_t(\mathbf{x},t)\right) \right)^2 .$$

Das Problem lässt sich dann mit der Form $\mathbf{W}^2\mathbf{A}\mathbf{v} = \mathbf{W}^2\mathbf{b}$ beschreiben. Die Lösung ist: $\mathbf{v} = (\mathbf{A}^T\mathbf{W}^2\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}^2(-\mathbf{b})$.

4.2.5. Verfahren von Uras et al.

Die *Helligkeitsbeständigkeits*-Bedingung aus Kapitel 2.3.2 stellt die Anforderung, dass die Intensität der betrachteten Stellen des Bildes über die Zeit konstant bleibt, also:

$$I_t(\mathbf{x},t) = 0 .$$

In dem Artikel [UGVT88] wird eine weitere, strengere Anforderung gestellt. Sie besagt, dass die räumlichen Gradienten des Bildes im Laufe der Zeit konstant bleiben:

$$\frac{d}{dt}\nabla I(\mathbf{x},t) = 0 .$$

Es dürfen also keine Änderungen, wie Dehnungen oder Rotation der Bildinformationen, auftreten. Für eine Positionsänderung um $\delta\mathbf{x}$ in der Zeiteinheit δt kann daraus dieser Zusammenhang aufgestellt werden:

$$\frac{d}{dt}\nabla I(\mathbf{x},t) = \frac{d}{dt}\nabla I(\mathbf{x}+\delta\mathbf{x}, t+\delta t) \ .$$

Durch Taylor-Entwicklung erhält man aus dieser Gleichung:

$$\frac{d}{dt}\nabla I(\mathbf{x},t) = \frac{d}{dt}\nabla \left(I(\mathbf{x},t) + \nabla I \cdot \delta\mathbf{x} + I_t\delta t + R_2\right) \ .$$

Für $\mathbf{v} = \delta x/\delta t$ lässt sich

$$0 = \nabla\left(\nabla I \cdot \mathbf{v} + I_t\right)$$

herleiten. Oder anders geschrieben:

$$\mathbf{H}\mathbf{v} = -\nabla I_t \ .$$

$\mathbf{H}$ ist die Hesse-Matrix in Bezug auf die Variablen x und y. Die Bedingung

$$\frac{d}{dt}\nabla I = 0$$

trifft genau dann zu, wenn die partiellen Ableitungen von $I(\mathbf{x},t)$ kommutativ sind, also:

$$\nabla\frac{dI}{dt} = \frac{d}{dt}\nabla I \ .$$

Diese Eigenschaft ist umso besser erfüllt, je mehr

$$\frac{|\mathbf{M}\nabla I|}{|\nabla I_t|} \ll 1$$

mit

$$\mathbf{M} \coloneqq (\nabla\mathbf{v}^T) = \begin{bmatrix} \frac{\delta u}{\delta x} & \frac{\delta v}{\delta x} \\ \frac{\delta u}{\delta y} & \frac{\delta v}{\delta y} \end{bmatrix}$$

gilt. Dieses Kriterium wird praktisch umgesetzt, indem aus einem 8×8 Teilbildbereich die Bildpunkte mit den acht kleinsten $\frac{|\mathbf{M}\nabla I|}{|\nabla I_t|}$ Werten bestimmt werden. Daraus wiederum wird der Wert mit der kleinsten Condition-Number *cond*$(\mathbf{H})$ als Geschwindigkeit für den gesamten Bereich gesetzt. Die Arbeit [BFB94] gibt an, dass die Determinante *det*$(\mathbf{H})$ als Auswahl zu stabileren Ergebnissen führt als die Condition-Number.

4.3. Block-Matching-Verfahren

4.3.1. Zweidimensionales Block-Matching

Das Prinzip von *Block-Matching*-Verfahren ist es, kleinen, mehr oder weniger großen Teilbereichen eines Bildes ein möglichst eindeutiges Identifikationsmerkmal zuzuordnen und diese Teilbereiche dann in den zeitlich aufeinanderfolgenden Bildern genau zu lokalisieren [Lar94]. Mit den Differenzen der Positionen dieser Bereiche in den unterschiedlichen Frames kann unmittelbar der optische Fluss bestimmt werden. Der Teilbereich eines Bildes $\mathbf{P}$ wird definiert als $R \times S$-Matrix $\mathbf{R}_{ij}$:

$$\mathbf{R}_{ij} := \left(\left(p_{i+r-\left\lfloor \frac{R+1}{2} \right\rfloor,\, j+s-\left\lfloor \frac{S+1}{2} \right\rfloor} \right) \right),$$

$$\text{für } 1 \leq r \leq R \text{ und } 1 \leq s \leq S\ .$$

R und S bestimmen die Größe des Bereichs und müssen kleiner als die Größe des Bildes selbst sein. Die Indizes i und j geben den Pixel des Bildes an, welcher im Mittelpunkt des Bildausschnitts $\mathbf{R}$ liegt. Für i gilt: $\left\lfloor \frac{R+1}{2} \right\rfloor \leq i \leq M - \left\lfloor \frac{R}{2} \right\rfloor$ und für j: $\left\lfloor \frac{S+1}{2} \right\rfloor \leq j \leq N - \left\lfloor \frac{S}{2} \right\rfloor$.

Zwei Bildausschnitte $\mathbf{R}^{(t-\delta t)}_{i-\delta y,\, j-\delta x}$ zum Zeitpunkt $t - \delta t$ und $\mathbf{R}^{(t)}_{ij}$ zum Zeitpunkt t werden durch eine Funktion $f : \mathbb{G}^2 \times \mathbb{G}^2 \to \mathbb{R}^+ \cup 0$ miteinander verglichen (siehe Kapitel 4.3.3). Die Werte des Vergleichs kann man für eine Auswahl von δx und δy in einer $K \times L$-Matrix $\mathbf{D_{ij}}$ fassen:

$$\mathbf{D}^{(t)}_{ij} := \left(\left(f \left(\mathbf{R}^{(t-\delta t)}_{i+k-\left\lfloor \frac{K+1}{2} \right\rfloor,\, j+l-\left\lfloor \frac{L+1}{2} \right\rfloor}, \mathbf{R}^{(t)}_{ij} \right) \right) \right),$$

$$\text{für } 1 \leq k \leq K \text{ und } 1 \leq l \leq L\ .$$

Gesucht wird also in einem Umfeld von $1 - \left\lfloor \frac{K+1}{2} \right\rfloor \leq \delta y \leq K - \left\lfloor \frac{K+1}{2} \right\rfloor$ für ganzzahlige δy. Für das Suchumfeld in x-Richtung gilt das gleiche. i und j dürfen nur in dem Bereich gewählt werden, in dem alle Umgebungen $\mathbf{R}_{ij}$, die beim Suchen benötigt werden, definiert sind, also:

$$\begin{array}{ccccc}
\left\lfloor \frac{R+1}{2} \right\rfloor - \left(1 - \left\lfloor \frac{K+1}{2} \right\rfloor\right) & \leq & i & \leq & M - \left\lfloor \frac{R}{2} \right\rfloor - \left(K - \left\lfloor \frac{K+1}{2} \right\rfloor\right) \\
\left\lfloor \frac{S+1}{2} \right\rfloor - \left(1 - \left\lfloor \frac{L+1}{2} \right\rfloor\right) & \leq & j & \leq & N - \left\lfloor \frac{S}{2} \right\rfloor - \left(L - \left\lfloor \frac{L+1}{2} \right\rfloor\right)\ .
\end{array}$$

Für alle i und j, die diese Einschränkungen erfüllen, kann so der optische Fluss

bestimmt werden:

$$\mathbf{v}(i,j,t) = \begin{bmatrix} minX(\mathbf{D}_{ij}^{(t)}) - \lfloor \frac{K+1}{2} \rfloor \\ minY(\mathbf{D}_{ij}^{(t)}) - \lfloor \frac{L+1}{2} \rfloor \end{bmatrix} \frac{1}{\delta t} \, .$$

Die Funktionen *minX* und *minY* geben den Spalten- bzw. Zeilenindex eines kleinsten Elementes einer Matrix zurück.

Der Hauptnachteil dieser Art der Algorithmen ist der hohe Rechenaufwand. Die Umgebung jedes Pixels muss mit der Nachbarschaft verglichen werden. Die Laufzeit steigt daher mit der Größe der Umgebung und des Suchradius stark an. Der Algorithmus eignet sich daher nur bedingt für Systeme mit begrenzten Ressourcen. Andererseits kann der optische Fluss eines einmal zuverlässig identifizierten Teilbereichs sehr genau bestimmt werden.

4.3.2. Eindimensionales Block-Matching

Verringert man den Algorithmus auf ein eindimensionales System, fällt der Ressourcenanspruch des Verfahrens deutlich kleiner aus. Analog zur zweidimensionalen Variante kann für einen gegebenen Zeilenvektor $\mathbf{p}_y$, wie er in Kapitel 3.5 definiert ist, eine Umgebung $\mathbf{r}_i$ der Dimension R definiert werden, die als Mittelpunkt Zeile i besitzt. i darf Werte zwischen $\lfloor \frac{R+1}{2} \rfloor$ und $M - \lfloor \frac{R}{2} \rfloor$ annehmen.

$$\mathbf{r}_i \coloneqq \begin{bmatrix} p_{y\,i-R_l} \\ \vdots \\ p_{y\,i-1} \\ p_{y\,i} \\ p_{y\,i+1} \\ \vdots \\ p_{y\,i+R_h} \end{bmatrix} ,$$

mit $R_l = \left\lfloor \frac{R-1}{2} \right\rfloor$ und $R_h = \lceil \frac{R-1}{2} \rceil$. Die Ergebnisse eines durchsuchten Umfeldes der Größe K für einen Element i von $\mathbf{p}_y$ werden in $\mathbf{d_y}_i$ erfasst:

$$\mathbf{d_y}_i \coloneqq \begin{bmatrix} f_{1D}\left(\mathbf{r}^{(t-\delta t)}_{i+1-\left\lfloor \frac{K+1}{2} \right\rfloor}, \mathbf{r}^{(t-\delta t)}_i\right) \\ \vdots \\ f_{1D}\left(\mathbf{r}^{(t-\delta t)}_{i-1}, \mathbf{r}^{(t-\delta t)}_i\right) \\ f_{1D}\left(\mathbf{r}^{(t-\delta t)}_{i}, \mathbf{r}^{(t-\delta t)}_i\right) \\ f_{1D}\left(\mathbf{r}^{(t-\delta t)}_{i+1}, \mathbf{r}^{(t-\delta t)}_i\right) \\ \vdots \\ f_{1D}\left(\mathbf{r}^{(t-\delta t)}_{i+K-\left\lfloor \frac{K+1}{2} \right\rfloor}, \mathbf{r}^{(t-\delta t)}_i\right) \end{bmatrix}.$$

Gesucht wird wieder im Bereich von $1 - \left\lfloor \frac{K+1}{2} \right\rfloor \leq \delta y \leq K - \left\lfloor \frac{K+1}{2} \right\rfloor$. Ein Vektor $\mathbf{d}_x$ lässt sich analog dazu für die x-Richtung definieren. Der optische Fluss der Zeilen- und Spalteninformation $\mathbf{p}_{y\,i}$ und $\mathbf{p}_{x\,j}$, kann durch

$$\mathbf{v}(i,j,t) = \begin{bmatrix} \mathit{minI}(\mathbf{d_x}_j^{(t)}) - \left\lfloor \frac{K+1}{2} \right\rfloor \\ \mathit{minI}(\mathbf{d_y}_i^{(t)}) - \left\lfloor \frac{L+1}{2} \right\rfloor \end{bmatrix} \frac{1}{\delta t}$$

bestimmt werden. *minI* gibt den Index eines kleinsten Elementes des Vektors zurück. i darf wieder nur im Bereich

$$\left\lfloor \frac{R+1}{2} \right\rfloor - \left(1 - \left\lfloor \frac{K+1}{2} \right\rfloor\right) \leq \quad i \quad \leq M - \left\lfloor \frac{R}{2} \right\rfloor - \left(K - \left\lfloor \frac{K+1}{2} \right\rfloor\right)$$

liegen.

4.3.3. Vergleichsfunktionen

Für den Vergleich von Bildbereichen wurde die Funktion f definiert. Sie berechnet aus zwei Matrizen gleicher Größe einen Wert, der eine Aussage über die Gleichheit der Matrizen macht. Der Rückgabewert von f ist positiv und reell. Je kleiner dieser Wert, desto identischer die Bildbereiche. Das Maß der Identität kann verschieden definiert werden. Das naheliegenste Vergleichskriterium ist wohl die Bildinformation der Teilbereiche selbst. Aus ihr lässt sich das Identitätsmaß für zwei $R \times S$-Matrizen $\mathbf{A}$ und $\mathbf{B}$ berechnen. Die eindimensionale Variante dieser Funktion ist mit f_{1D} bezeichnet. Sie berechnen aus zwei Vektoren $\mathbf{a}$ und $\mathbf{b}$ der Dimension R das

Identitätsmaß:

1. kleinste Fehlerquadrate:

$$f(\mathbf{A},\mathbf{B}) = \sum_{i=1}^{R}\sum_{j=1}^{S}(a_{ij}-b_{ij})^2$$
$$f_{1D}(\mathbf{a},\mathbf{b}) = \sum_{i=1}^{R}(a_i-b_i)^2$$

2. kleinste absolute Differenzen:

$$f(\mathbf{A},\mathbf{B}) = \sum_{i=1}^{R}\sum_{j=1}^{S}|a_{ij}-b_{ij}|$$
$$f_{1D}(\mathbf{a},\mathbf{b}) = \sum_{i=1}^{R}|a_i-b_i|$$

3. Vergleich über durchschnittliche Helligkeit:

$$f(\mathbf{A},\mathbf{B}) = \left|\frac{1}{RS}\sum_{i=1}^{R}\sum_{j=1}^{S}a_{ij} - \frac{1}{RS}\sum_{i=1}^{R}\sum_{j=1}^{S}b_{ij}\right|$$
$$f_{1D}(\mathbf{a},\mathbf{b}) = \left|\frac{1}{R}\sum_{i=1}^{R}a_i - \frac{1}{R}\sum_{i=1}^{R}b_i\right|$$

Eine weitere Möglichkeit ist es, aus dem Bildbereich erst einen *Hashwert* zu berechnen und diesen dann mit den oben genannten Verfahren zu vergleichen [TMT00]. Dieser *Hashwert* kann auch spezielle Eigenschaften wie Kanten und Farben besonders berücksichtigen [ZL00].

4.4. Phasenbasierte Algorithmen

4.4.1. Fleet und Jepson

Dieser von Fleet und Jepson in [FJ90] vorgestellte Algorithmus basiert auf der Annahme, dass Phaseninformationen eines Bildes stabiler sind als die Amplituden der Bildinformation [FJ93]. Die Phaseninformationen für einen Ort $\mathbf{x}$ des Bildes zum Zeitpunkt t werden durch einen Gaborfilter bestimmt [Gab46]. Die Ausgabe eines

auf eine Frequenz und eine Richtung eingestellen Filters sind komplex:

$$R(\mathbf{x},t) = \rho(\mathbf{x},t)e^{(i\phi(\mathbf{x},t))} \ .$$

Der optische Fluss jedes Pixels ergibt sich durch Anwendung des First-Order-Verfahrens auf die Phaseninformationen $\phi(\mathbf{x},t))$:

$$\mathbf{v}_\perp := -\frac{\phi_t(\mathbf{x},t)\nabla\phi(\mathbf{x},t)}{|\nabla\phi(\mathbf{x},t)|^2} \ .$$

Die Gaborfilterung lässt die Einstellung von drei Parametern wie folgt:

$$R_{T,\theta,\sigma}(\mathbf{x}_0,t) = \int_{-\infty}^{+\infty} I(\mathbf{x},t)g_\sigma(\mathbf{x}-\mathbf{x}_0)b_{T,\theta}(\mathbf{x})\ dx\ dy$$

zu, wobei $g_\sigma(\mathbf{x})$ ein Fenster ist, welches die Transformation lokal begrenzt und $b_{T,\theta}(\mathbf{x})$ die komplexwertige Basisfunktion. Der Parameter σ bestimmt die Breite der lokalen Begrenzung:

$$g_\sigma(\mathbf{x}) = e^{-\frac{x^2+y^2}{4\sigma}} \ ,$$

T und θ beschreiben die Basisfunktion:

$$b_{T,\theta}(\mathbf{x}) = \cos\left(\mathbf{x}\cdot\begin{bmatrix}\sin\theta\\-\cos\theta\end{bmatrix}\frac{2\pi}{T}\right)/2+0,5-i\left(\sin\left(\mathbf{x}\cdot\begin{bmatrix}\sin\theta\\-\cos\theta\end{bmatrix}\frac{2\pi}{T}\right)/2+0,5\right) \ .$$

4.5. Neuronale Algorithmen

In diesem Kapitel werden Algorithmen vorgestellt, die den optischen Fluss auf Basis eines in Kapitel 2.4.2 vorgestellten neuronalen Netztes bestimmen können.

4.5.1. Reichardt-Detektor

Der Reichardt-Detektor basiert auf dem Prinzip, Kanten zu erkennen und deren räumliche Änderung im Laufe der Zeit zuzuordnen. Vorgestellt hat dieses Prinzip Werner Reichardt in der Arbeit [Rei57] (oder weiterführend in [Rei61]). Aufgebaut ist diese Anordnung aus zwei Halbdetektoren, die je eine Bewegungsrichtung der Kanten erkennen. Abbildung 4.2 zeigt einen Halbdetektor als Übersichtsdiagramm. Zu sehen sind zwei räumlich nebeneinander angeordnete Lichtsensoren. Im oberen Bereich ist eine wahrgenommene Kante dargestellt, die im Verlaufe der Zeit ihre

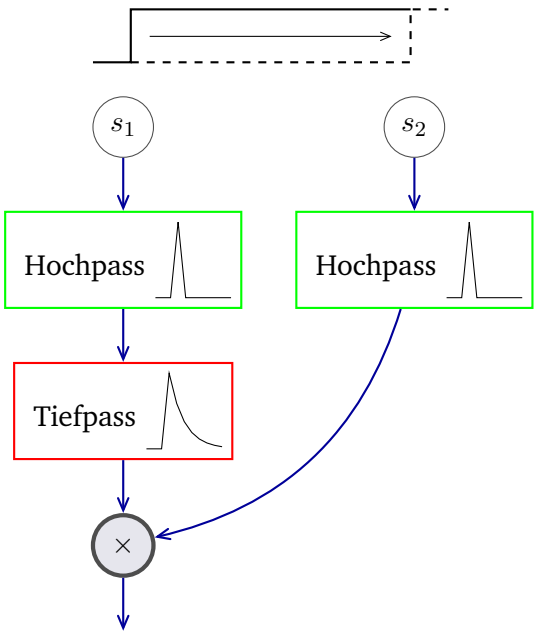

Abbildung 4.2.: *Reichardt-Halbdetektor als Übersichtsdiagramm*

Position relativ zu den Sensoren ändert. Das Signal der Sensoren durchläuft anschließend einige Verarbeitungsschritte. Der erste ist eine Hochpassfilterung. Einer der beiden Signalwege von S_1 und S_2 führt anschließend durch einen Tiefpassfilter, bevor beide miteinander multipliziert werden. In der ersten Stufe erfolgt also die Kantenerkennung. Der Tiefpassfilter übernimmt die Aufgabe eines Gedächtnisses. Er weiß, vor welcher Zeit eine Kante den Sensor passiert hat. Je schneller diese Kante auch den Sensor zwei passiert, desto höher ist das Ergebnis der Multiplikation.

In Abbildung 4.3 sind zwei zu einem Volldetektor kombinierte Halbdetektoren dargestellt. Die beiden gespiegelten Halbdetektoren ermöglichen die Erkennung beider Bewegungsrichtungen der Kanten. Diese Annordung entspricht dem ausgearbeiteten Reichardt-Detektor aus [vSS85].

Ein Halbdetektor lässt sich einfach mit wenigen Sigma-Pi-Neuronen umsetzen [Wol07]. Abbildung 4.4 zeigt diese Umsetzung. s_1 und s_2 stellen die beiden Lichtrezeptoren dar. Die grün umrandeten Neuronen und Gewichte wirken wie ein Hochpassfilter, die rot umrandeten bilden den Tiefpass. Das π-Neuron multipliziert beide Eingangswerte auf. In der Arbeit [Wol07] ist der Reichardt-Detektor auch mit einem Netz umgesetzt, welches nur aus Sigma-Neuronen besteht. Es leistet im wesentlichen das gleiche, nur dass es nur auf eine Kantenrichtung reagiert.

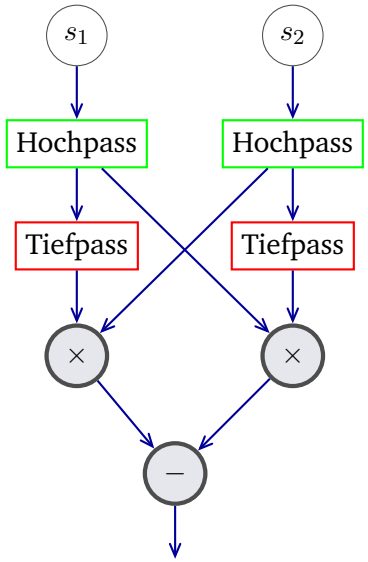

Abbildung 4.3.: *Reichardt-Detektor als Übersichtsdiagramm*

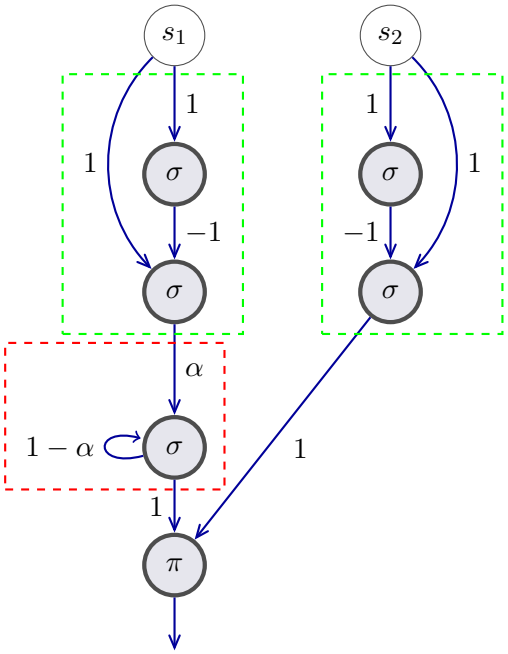

Abbildung 4.4.: *Reichardt-Halbdetektor mit Sigma-Pi-Neuronen nach [Wol07]*

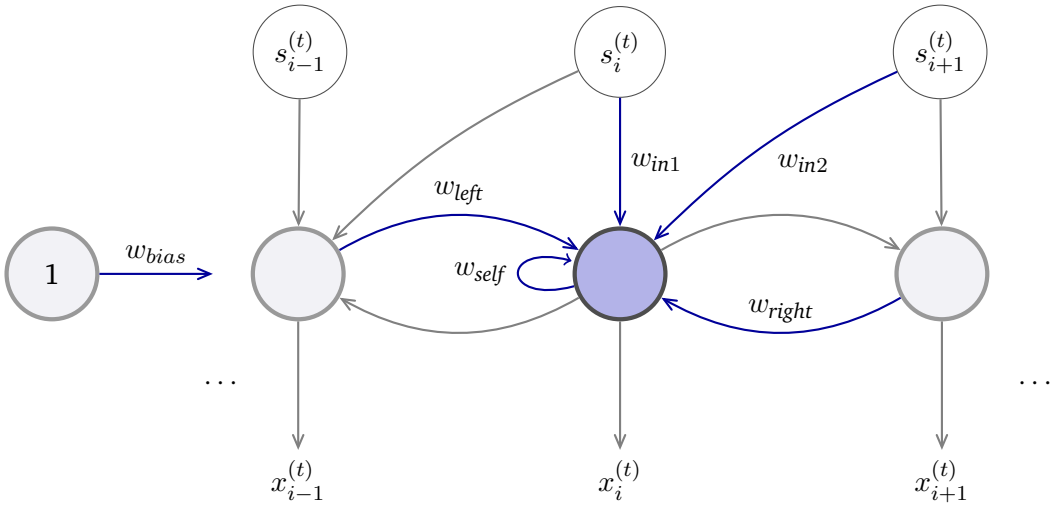

Abbildung 4.5.: *Netztopologie der einfachen Neuronenzeile*

4.5.2. Neuronenzeile

Neben der neuronalen Umsetzung des Reichardt-Detektors enthält die Arbeit [Wol07] neuronale Anordnungen, die Bewegungen mehrerer in Reihe angeordneter Lichtsensoren detektieren können. Wie bei dem Reichardt-Halbdetektor können sie nur eine Bewegungsrichtung erkennen. Für einen Volldetektor müssen wieder zwei dieser Halbdetektoren kombiniert werden. In [Wol07] werden zwei Anordnungen anhand ihrer Netztopologie unterschieden. Beide werden werden folgend näher erläutert.

4.5.2.1. Einfache Neuronenzeile

Die Netztopologie des einlagigen neuronalen Halbdetektors ist in Abbildung 4.5 zu sehen. Die Anzahl der Neuronen entspricht der Anzahl der Lichtsensoren s_i. Das Bias-Neuron links wirkt auf jedes Neuron in der Zeile. Jedes Neuron ist nur mit seinen Nachbarneuronen mit den angegebenen Gewichten verknüpft. An den Rändern der Neuronenzeile entfallen natürlich entsprechende Synapsen. Die Lichtsensoren s_i nehmen Werte im Intervall von $[-1, 1]$ an, was den Helligkeitsintensitäten des Bildsignals entspricht. Als Netzausgabe wird in [Wol07] eine gewichtete Summe über alle Neuronen genutzt:

$$v_l = \frac{1}{I} \sum_{i=0}^{I-1} x_{l\,i}^{(t)} \ .$$

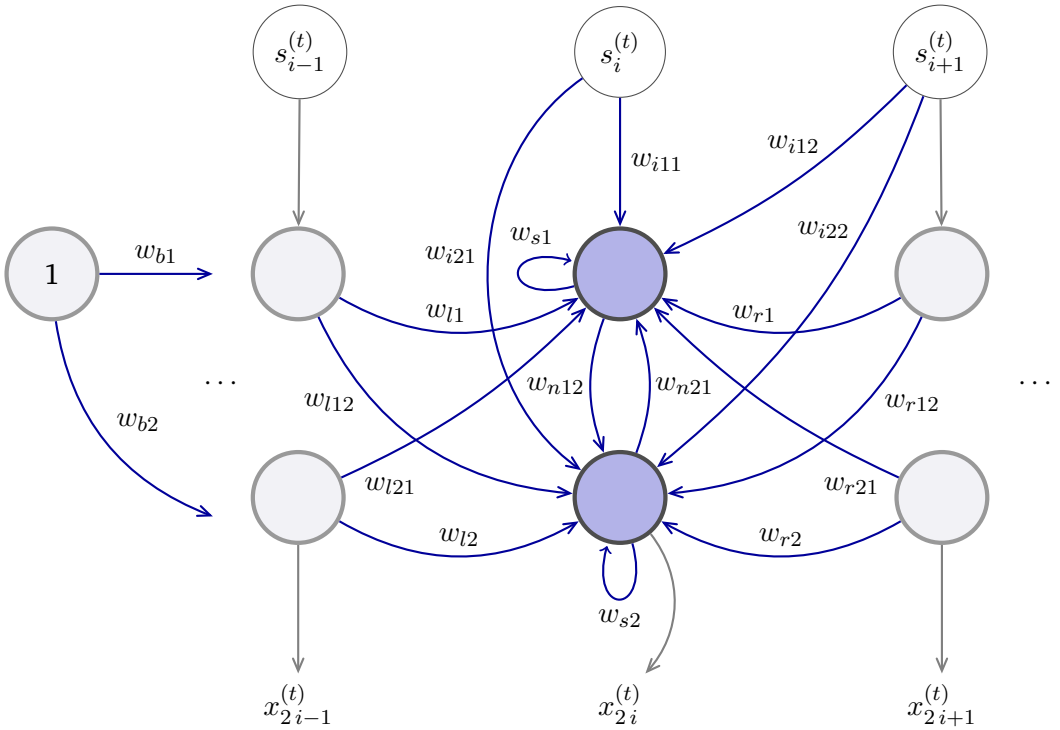

Abbildung 4.6.: *Netztopologie der zweifachen Neuronenzeile*

x_l sind die Neuronen für die Detektion einer Bewegungsrichtung, I die Anzahl der Neuronen. Die entgegengesetzte Geschwindigkeitskomponte v_r lässt sich analog definieren. $v := v_l - v_r$ ist der resultierende optische Fluss.

Die Gewichte $(w_{in1}, w_{in2}, w_{bias}, w_{self}, w_{left}, w_{right})$ werden in [Wol07] durch Evolution für verschiedene Reizmuster bestimmt. Davon sollen hier die Werte für komplexe künstliche Reize

$$(w_{in1}, w_{in2}, w_{bias}, w_{self}, w_{left}, w_{right}) := (-3.8, 3.7, -1.12, 0.21, 2.18, -0.09)$$

und natürliche Reize

$$(w_{in1}, w_{in2}, w_{bias}, w_{self}, w_{left}, w_{right}) := (2.6, -2.7, -1.9, 1.6, 1.5, -1.5)$$

genutzt werden.

4.5.2.2. Doppelte Neuronenzeile

Die Topologie der doppelten Neuronenzeile ist in Abbildung 4.6 zu sehen. Die Netz-

ausgabe für diesen einen Halbdetektor ist:

$$v_l = \frac{1}{I} \sum_{i=0}^{I-1} x_{l\,i}^{(t)} \ .$$

$v = v_l - v_r$ lässt sich analog zur einlagigen Neuronenebene berechnen. Die Gewichte für die verschiedenen Reize sind:

Synapsen / Gewichte	w_{i11}	w_{i21}	w_{i12}	w_{i22}	w_{b1}	w_{b2}
komplexe künstliche Reize	2.4	3.8	2.2	0.4	2.9	-1.6
natürliche Reize	3.9	-3.3	0	1.5	-0.15	-3.6

...

w_{s1}	w_{s2}	w_{n12}	w_{n21}	w_{l1}	w_{l12}
0	1.1	-2.5	$0 - 0.9$	-0.8	0
0	2.4	2.7	0	0	-2.5

... ...

w_{l21}	w_{l2}	w_{r1}	w_{r12}	w_{r21}	w_{r2}
-0.6	0	1.2	-1.9	1.6	0
0.4	1.6	0	0.3	0	-2.5

...

4.5.3. Slow-Feature-Analysis

Die *Slow-Feature-Analysis* ist ein Verfahren, welches aus einem mehrdimensionalen Eingangssignal Komponenten mit der geringsten Änderungsrate extrahiert. Der Hintergrund ist, dass wesentliche Informationen, die vermischt mit unwesentlichen, im Signalstrom der Sensoren auftreten, meist diejenigen sind, welche sich eben nur langsam ändern. Ein Beispiel ist das Rauschen eines Winkelsensors. Wertänderungen durch Rauschen sind in der Regel schneller, als die tatsächlich zu messende Größe, in diesem Fall der Winkel. Dieses Verfahren wird unter neuronalen Algorithmen geführt, da es mit einem neuronalen Netz implementiert werden kann und ein Modell der Bildverarbeitung bei biologischen Lebewesen darstellt [WS02].

Für ein gegebenes mehrdimensionales Eingangssignal $\mathbf{x}(t)$ wird eine Funktion g_j

so bestimmt, dass für das Ausgangssignal $y_j(t) = g_j(\mathbf{x}(t))$ die Bedingung

$$\Delta_j := \Delta(y_j) := \langle \dot{y}_j^2 \rangle_t \tag{4.7}$$

minimiert wird, unter den Randbedingungen

$$\langle y_j \rangle = 0 \,, \tag{4.8}$$
$$\langle y_j^2 \rangle = 0 \,, \tag{4.9}$$
$$\forall i < j : \langle y_i y_j \rangle = 0 \,. \tag{4.10}$$

Die Eingangsdaten müssen dafür wie folgt normiert vorliegen:

$$\langle x_i \rangle = 0 \,,$$
$$\langle x_i^2 \rangle = 1 \,,$$

für jede Komponente x_i von $\mathbf{x}$. Der zu minimierende Term 4.7 führt zu einer minimalen Änderung des Signals y_j über die Zeit. Bedingung 4.8 führt zu einem Mittelwert des Ausgangs von 0, Gleichung 4.9 zu einer Varianz von 1 und 4.10 stellt sicher, dass die verschiedenen Ausgänge y_j inhaltlich verschiedene Informationen tragen und nicht das Gleiche anzeigen. Auf eine genauere Herleitung des Verfahrens soll hier verzichtet werden. Dafür kann auf die Arbeit [WS07] verwiesen werden. Für eine gegebene Menge an Daten $\mathbf{x}(t)$ mit $t_0 < t < t^*$ werden die Parameter der Funktion g_j bestimmt. Dieser Vorgang kann auch als Lernphase betrachtet werden. Wendet man die Funktion g_j auf Daten $\mathbf{x}(t)$ mit $t > t^*$ an, ist das die Ausführungsphase. Die Ausgangssignale y_j sollten in der Ausführungsphase die Informationen repräsentieren, die vorher gelernt wurden.

Für die Detektion des optischen Flusses wird eine Teilmenge aller Pixel des Bildes als Eingangsdaten $\mathbf{x}(t)$ gewählt. Für einen einfachen Test liegen diese bevorzugt auf einer Linie parallel zum auftretenden optischen Fluss. Um tatsächlich eine Bewegung zu detektieren, müssen mindestens zwei Pixel als Eingang verwendet werden:

$$\mathbf{x}(t) := \begin{bmatrix} p_{yx}(t) \\ p_{y,\,x+1}(t) \end{bmatrix} \quad ,$$

für ein festes x und y, so dass alle angesprochenen Pixel innerhalb der Bildgrenzen liegen. Ohne die Information der zeitlichen Änderung der Pixel, kann kein optischer Fluss bestimmt werden. Daher wird das Eingangssignal um genau einen Zeitschritt

temporal expandiert:

$$\mathbf{x}(t) \coloneqq \begin{bmatrix} p_{yx}(t) \\ p_{y,\,x+1}(t) \\ p_{yx}(t+1) \\ p_{y,\,x+1}(t+1) \end{bmatrix} .$$

Die Anzahl der Eingänge kann durch eine quadratische Erweiterung noch erhöht werden:

$$\mathbf{x}(t) \coloneqq \begin{bmatrix} p_{yx}(t) \\ p_{y,\,x+1}(t) \\ p_{yx}(t+1) \\ p_{y,\,x+1}(t+1) \\ p_{yx}(t)^2 \\ p_{y,\,x+1}(t)^2 \\ p_{yx}(t-1)^2 \\ p_{y,\,x+1}(t+1)^2 \\ p_{yx}(t)p_{y,\,x+1}(t) \\ p_{yx}(t)p_{yx}(t+1) \\ p_{yx}(t)p_{y,\,x+1}(t+1) \\ p_{y,\,x+1}(t)p_{yx}(t+1) \\ p_{y,\,x+1}(t)p_{y,\,x+1}(t+1) \\ p_{yx}(t-1)p_{y,\,x+1}(t+1) \end{bmatrix} .$$

Ein zusätzlicher Verarbeitungsschritt der Daten ist das Hinzufügen von Rauschen zu jedem Eingangssignal. Erzeugt wird es auf gleiche Weise wie das Rauschen in Kapitel 5.1.2.1. Analog können die Eingangswerte bei einer längeren Pixelzeile verändert werden.

4.6. Laufzeitbetrachtung

Ein weiteres Bewertungskriterium ist die Laufzeit der vorgestellten Algorithmen. Diese hängt jedoch stark von der eingesetzten Hardware ab, auf dem die Algorithmen implementiert sind. Insbesondere bei beschränkten Hardwareressourcen muss häufig zwischen Bearbeitungsgeschwindigkeit und Speicherbenutzung abgewogen werden. Des Weiteren sind die hardwareseitigen Rechenmöglichkeiten sehr unterschiedlich. Fließkomma- oder Divisionsrecheneinheiten sind längst nicht immer vorhanden. Auch ist die Art und Weise, wie ein Bild der Recheneinheit zur Verfügung gestellt wird von Bedeutung. Dieses kann Bildpunkt für Bildpunkt, zeilen- oder spal-

tenweise oder durch *Shared-Memory* als ganzes Bild geschehen. Die Algorithmen, die auf Basis neuronaler Netze funktionieren, hängen stark von der Geschwindigkeit der gewählten analogen Netznachbildung ab.

All diese Gegebenheiten verändern die Laufzeit jedes Algorithmus auf der eingesetzten Hardware besonders und zum Teil erheblich. Daher werden an dieser Stelle keine näheren Aussagen über mögliche Laufzeiten der Algorithmen getroffen.

5. Experimente

In diesem Kapitel werden die Algorithmen auf ihre Leistungsfähigkeit beim Detektieren des optischen Flusses in verschiedenen Bildszenarien untersucht. Grundsätzlich ist die Untersuchung eines Verfahrens wie folgt aufgebaut:

1. Untersuchung des Verfahrens auf die grundsätzliche Funktionsfähigkeit ohne Vorverarbeitung an synthetischen Daten.
2. Bestimmung der geeignetsten Parameter und Vorverarbeitungsschritte anhand realer Daten.

Im Anschluss werden alle Algorithmen mit ihren jeweils besten Einstellungen und Vorverarbeitungsstufen miteinander verglichen. Die Untersuchungskriterien dafür sind die Anfälligkeit der Ausgabe in Bezug auf Änderungen in:

- Kontrast,
- Helligkeit,
- Rauschen,
- Anzahl kontrastreicher Kanten.

Wie in Kapitel 2.3.4 begründet, wird nur die horizontale Flusskomponente des Bildausschnitts in den Diagrammen dargestellt und verglichen. Daraus ergeben sich weitere Untersuchtungsmöglichkeiten, wie:

- Verhalten der Algorithmen bei optischen Flüssen größer als 1 *Pixel/Frame*,
- Auswirkungen von Überlagerung mit optischen Flüssen unterschiedlicher Richtungen,
- Einfluss von Rotation im Bildausschnitt bei gleichzeitiger horizontaler Translation.

Implementiert sind die Verfahren mit dem Framework OpenCV [Bra00]. Auf die Verwendung von Standardimplementierungen für einzelne Verfahren wurde jedoch verzichtet.

5.1. Bildmaterial

Die für die Tests benötigten bewegten Bilder entstehen durch Bewegen eines 100×100 Pixel großen Ausschnitts $\mathbf{P}^{(t)} = \left(\left(p_{yx}^{(t)} \right) \right)$ auf einem Basisbild $\mathbf{B} = ((b_{dc}))$ der Größe 500×500. $\mathbf{P}^{(t)}$ kann aus $\mathbf{B}$ abgeleitet werden:

$$\begin{aligned} p_{yx}^{(t)} = \quad & b_{y_0+\lfloor s \rfloor,\, x_0+\lfloor r \rfloor} && (1-s+\lfloor s \rfloor)\,(1-r+\lfloor r \rfloor) && + \ \ldots \\ \ldots + \ & b_{y_0+\lfloor s \rfloor,\, x_0+\lceil r \rceil} && (1-s+\lfloor s \rfloor)\,(r-\lfloor r \rfloor) && + \ \ldots \\ \ldots + \ & b_{y_0+\lceil s \rceil,\, x_0+\lfloor r \rfloor} && (s-\lfloor s \rfloor)\,(1-r+\lfloor r \rfloor) && + \ \ldots \\ \ldots + \ & b_{y_0+\lceil s \rceil,\, x_0+\lceil r \rceil} && (s-\lfloor s \rfloor)\,(r-\lfloor r \rfloor) \end{aligned}$$

mit

$$s = \int_{t_0}^{t} v(\mathbf{x}, t)\ dt\ ,$$

$$r = \int_{t_0}^{t} u(\mathbf{x}, t)\ dt\ ,$$

für

$$1 \leq x, y \leq 100,\ 1 \leq c, d \leq 500,\ x, y, c, d \in \mathbb{N}$$

und

$$1 \leq y_0 + \lfloor s \rfloor, y_0 + \lceil s \rceil, x_0 + \lfloor r \rfloor, x_0 + \lceil r \rceil \leq 500,\ x_0, y_0 \in \mathbb{N}\ .$$

Diese Rechnung führt zugleich eine Interpolation für ungerade r und s durch.

Folgend wird erläutert, welche Bilder für die Untersuchungen genutzt werden, beziehungsweise wie sie zustande kommen.

5.1.1. Synthetische Bilddaten

Da lediglich die grundlegende Funktion der Algorithmen anhand der synthetischen Daten überprüft werden soll, wird ein künstliches Muster genutzt. Aufgrund der guten Differenzierbarkeit und räumlichen Periodizität wird ein zweidimensionaler Sinusoid gewählt. Er entsteht durch Überlagerung von zwei zweidimensional extrudierten eindimensionalen Sinusoiden mit unterschiedlicher Ausrichtung in der

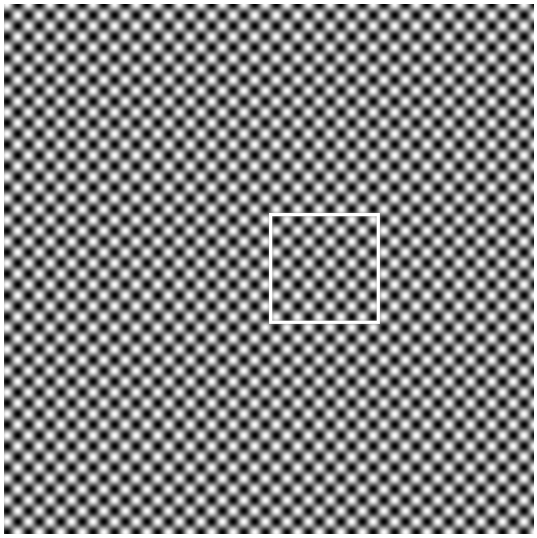

Abbildung 5.1.: *Synthetisches Bild mit $T = 20$. Weiß dargestellt ist der Bildausschnitt, der bewegt wird. $x_0 = 250$ und $y_0 = 200$.*

Fläche. Gebildet werden sie durch:

$$I_{s\ T,\theta}(\mathbf{x}) = \sin\left(\mathbf{x} \cdot \begin{bmatrix} \sin\theta \\ -\cos\theta \end{bmatrix} \frac{2\pi}{T}\right) ,$$

wobei θ die Orientierung und T die Periode ist. Daraus wird das Testmuster erzeugt:

$$b_{2Ddc} \coloneqq I_{s\ T,0}\left(\begin{bmatrix} c \\ d \end{bmatrix}\right) I_{s\ T,\frac{\pi}{2}}\left(\begin{bmatrix} c \\ d \end{bmatrix}\right) .$$

Abbildung 5.1 zeigt das entstandene Bild. Alle Algorithmen, die nur eine Zeile des Bildes nutzen, werden mit dem synthetischen Bild

$$b_{1Ddc} \coloneqq I_{s\ T,\frac{\pi}{2}}\left(\begin{bmatrix} c \\ d \end{bmatrix}\right)$$

getestet.

5.1.2. Reale Bilddaten

Das Ausgangsbild für die Untersuchungen an realen Bilddaten ist in Abbildung 5.2 zu sehen[1]. Dieses Bild wird für die einzelnen Vergleiche verändert.

[1] Luftbild von $52°25'46''$ Nord, $13°32'07''$ Ost aus Google Maps

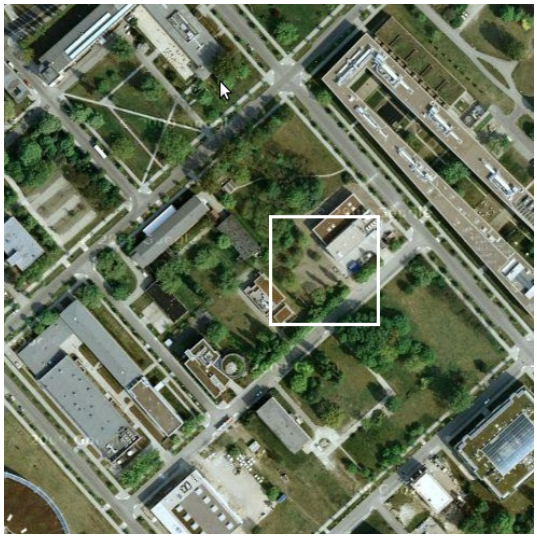

Abbildung 5.2.: *Reales Bild mit Bildausschnitt*

5.1.2.1. Rauschen

Die Funktion r fügt einem Eingangswert ein Rauschen in Abhängigkeit des Parameters p_r hinzu:

$$r(x, p_r) = x(1 + 2(rand(p_r)) - p_r) \ .$$

Ein mit Rauschen versehenes Bild $\mathbf{B}_r$ wird aus dem Bild $\mathbf{B}$ mit der Funktion r bestimmt:

$$b_{r\,dc} = \begin{cases} I_{min} & r(b_{dc}, p_r) < I_{min} \\ I_{max} & r(b_{dc}, p_r) > I_{max} \\ r(b_{dc}, p_r) & \text{sonst} \end{cases} \ .$$

Eine weitere Möglichkeit, eine Aussage über das Rauschverhalten eines Signals zu treffen, ist das *Signal-Rausch-Verhältnis* (kurz: *SNR*). Definiert ist es mit:

$$SNR = \frac{A_{\text{Signal}}}{\sigma_{\text{Signal}}} \ ,$$

A_{Signal} ist dabei die Amplitude des Signals und σ_{Signal} die Standardabweichung des Rauschens. Die Größe p_r kann für das gegebene Bild 5.2 in das SNR-Maß übertragen werden. In Tabelle 5.1 sind für ausgewählte p_r entsprechende SNR-Werte zu sehen. Abbildung 5.3 zeigt ein Bild mit einem *Signal-Rausch-Verhältnis* von 3,8. Für jedes Bild in $\mathbf{P}^{(t)}$ wird dieses Rauschen neu berechnet.

p_r	30 %	40 %	50 %	60 %
SNR	6,3	4,7	3,8	3,2

Tabelle 5.1.: *Umrechnung zwischen p_r und SNR*

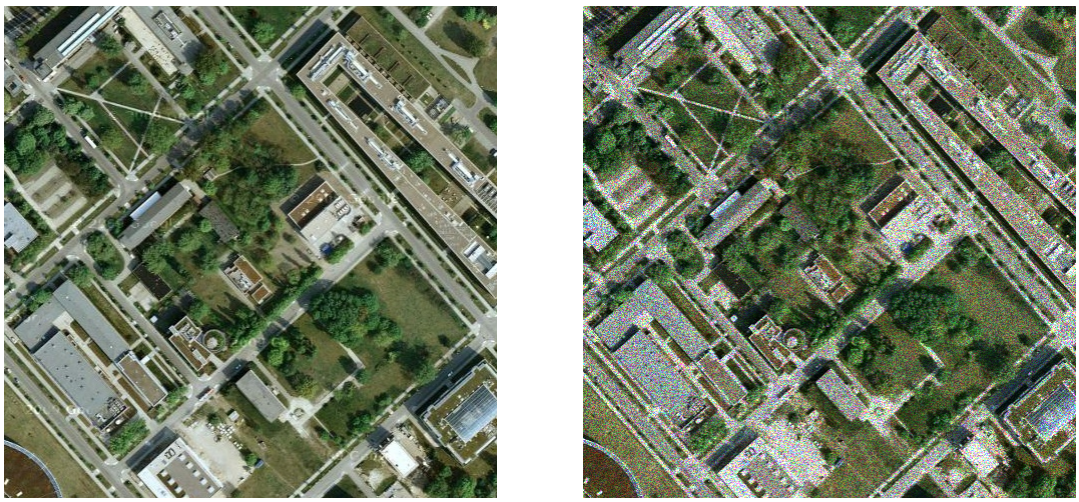

Abbildung 5.3a.: Original

Abbildung 5.3b.: SNR = 3,8

Abbildung 5.3.: *Veranschaulichung eines verrauschten Bildes*

Abbildung 5.4a.:
Original

Abbildung 5.4b.:
30 % Helligkeitsreduzierung

Abbildung 5.4.: *Veranschaulichung einer Helligkeitsreduzierung*

5.1.2.2. Helligkeit

Funktion h ändert die Helligkeit eines Bildpunktes um p Prozent:

$$h(x, p_h) = x + (I_{max} - I_{min})p_h \ .$$

y Die Helligkeitsänderung eines Bildes $\mathbf{B}$ führt zu $\mathbf{B}_h$:

$$b_{hdc} = \begin{cases} I_{min} & h(b_{dc}, p_h) < I_{min} \\ I_{max} & h(b_{dc}, p_h) > I_{max} \\ h(b_{dc}, p_h) & \text{sonst} \end{cases} \ .$$

Der Einfluss einer 30 %igen Verdunklung ist in Abbildung 5.4 zu sehen.

5.1.2.3. Kontrast

Bei einem Bild $\mathbf{B}$ wird der Kontrast durch Histogrammstauchung verringert, was zum Bild $\mathbf{B}_k$ führt. Durch die Histrogrammstauchung wird der vorhandene Intensitätsumfang $[I_{min}, I_{max}]$ auf $[I_{min\,k}, I_{max\,k}]$ mit $I_{min\,k} > I_{min}$ und $I_{max\,k} < I_{max}$ abgebil-

Abbildung 5.5a.:
Original

Abbildung 5.5b.:
50 % Kontrastreduzierung

Abbildung 5.5.: *Veranschaulichung einer Kontrastreduzierung*

det. Die Funktion k reduziert den Intensitätsumfang symmetrisch um p_k Prozent:

$$k(x, p_k) = \left(\frac{I_{max} - I_{min}}{2} - x \right) p_k + x \ .$$

Das kontrastreduzierte Bild $\mathbf{B}_k$ ergibt sich aus:

$$b_{k\,dc} = k(b_{dc}, p_k) \ .$$

Abbildung 5.5 zeigt ein um 50% im Kontrast reduziertes Bild.

5.1.2.4. Anzahl kontrastreicher Kanten

Dieses Attribut eines Bildes ist schwer messbar. Ein Wert, der die Anzahl kontrastreicher Kanten dennoch gut repräsentieren kann, ist die Varianz der ersten räumlichen Ableitungen des Bildes:

$$\sigma^2 = \frac{|Var(\nabla I(\mathbf{x}, t))|_1}{2} \ .$$

Abbildung 5.6 zeigt vier Bilder mit abnehmendem σ^2.

Abbildung 5.6a.: $\sigma^2 = 237$ *Abbildung 5.6b.:* $\sigma^2 = 136$ *Abbildung 5.6c.:* $\sigma^2 = 113$ *Abbildung 5.6d.:* $\sigma^2 = 77$

Abbildung 5.6.: *Bilder mit Unterschiedlicher Anzahl kontrastreicher Kanten*

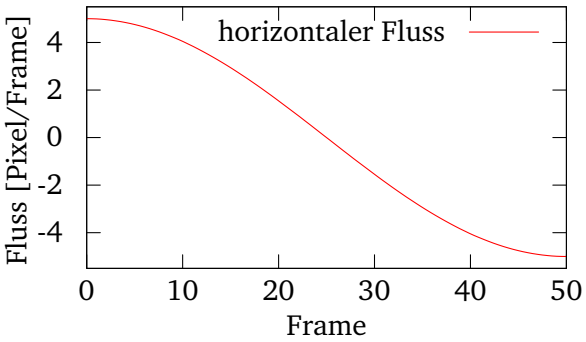

Abbildung 5.7.: *Erzeugter optischer Fluss in horizontaler Richtung*

5.2. Flussarten

Der Bildausschnitt, der in Abbildung 5.2 zu sehen ist, wird mit einer Geschwindigkeit horizontal über das Bild bewegt, die einer halben Periode einer Sinuskurve mit einer Amplitude von $A = 5$ *Pixel/Frame* entspricht. Dieser Verlauf wird gewählt, da er bei einer Messung die Ausgabe der Algorithmen bei sehr kleinen, als auch sehr großen optischen Flüssen, sowie allen Abstufungen dazwischen aufzeigt. Der erzeugte optische Fluss ist in Abbildung 5.7 zu sehen. In Fällen, in denenen störende Effekte auf den optischen Fluss gemessen werden sollen, wird dieser konstant gehalten. Abbildung 5.8 zeigt einen Test bei dem zusätzlich zum horizontalen optischen Fluss ein störender Fluss in vertikaler Richtung erzeugt wird.

Eine weitere häufig auftretende störende Bewegung ist eine Rotation des Bildes. Sie sollte keinen Einfluss auf die Detektion des optischen Flusses in horizontaler Richtung haben, wenn das Zentrum der Drehung der Mitte des Bildes entspricht. Diese Rotation wird erzeugt, indem das Basisbild um das Zentrum des aktuellen

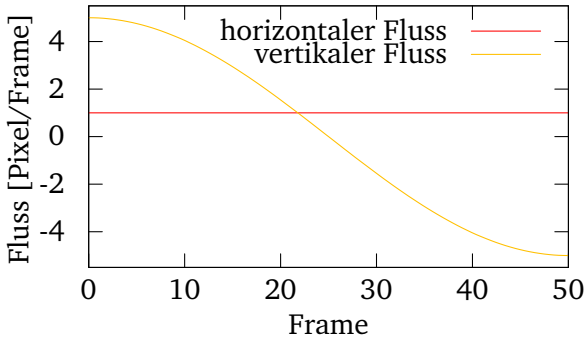

Abbildung 5.8.: *Erzeugter optischer Fluss in horizontaler Richtung und ein störender Fluss in vertikaler Richtung*

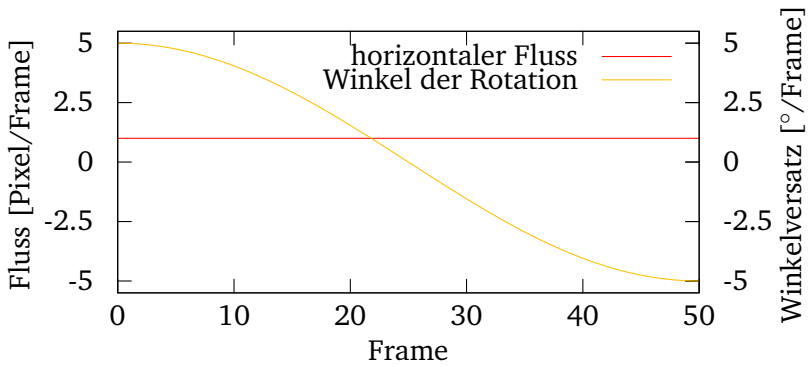

Abbildung 5.9.: *Erzeugter optischer Fluss in horizontaler Richtung und ein störender Rotationsfluss*

Bildausschnitts mit einem Winkel α gedreht wird:

$$b_{\alpha dc} = b_{fe} \ ,$$

wobei e und f durch

$$\begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \left(\begin{bmatrix} c \\ d \end{bmatrix} - \begin{bmatrix} x_0 + \frac{N}{2} \\ y_0 + \frac{M}{2} \end{bmatrix} \right) + \begin{bmatrix} x_0 + \frac{N}{2} \\ y_0 + \frac{M}{2} \end{bmatrix}$$

erzeugt werden. Abbildung 5.9 zeigt die Flusskonstellation bei zusätzlich auftretender Rotation.

5.3. Untersuchungsergebnisse

Dieses Kapitel beschreibt detailliert die Ergebnisse der Untersuchungen für jeden Algorithmus. Die für die Analyse notwendigen ausführlichen Diagramme sind im Anhang zu finden. Die charakteristische Ausgabe jedes Algorithmus für die gewählten Parameter ist in den einzelnen Abschnitten dargestellt (Abbildungen 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.17, 5.18).

5.3.1. Zeilen-Block-Matching

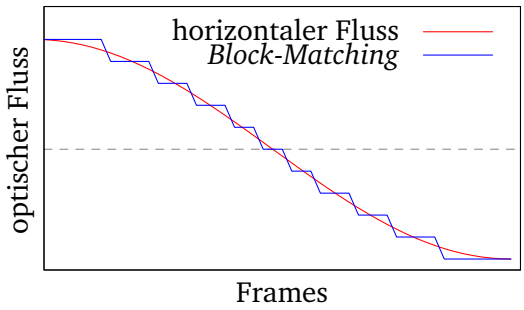

Abbildung 5.10: *Charakteristische Ausgabe des* Block-Matching-*Algorithmus*

Das *Zeilen-Block-Matching*-Verfahren funktioniert grundsätzlich, wie der Test mit synthetischem Bildmaterial in A.1 zeigt. Die Periodizität der Bildmerkmale sollte jedoch größer sein als der optische Fluss, der durch den Algorithmus mit der Suchumgebung abgedeckt ist, da es sonst zu falschen Ausgaben kommt, wie bei $T = 5$ und $T = 8$ zu sehen ist. Dieser Test wurde mit einer Suchumgebung durchgeführt, die jeden auftretenden optischen Fluss umfasst. Interessant ist die Frage, was ausgegeben wird, wenn der optische Fluss größer als der Suchradius ist. Abbildung A.2 zeigt ein entsprechendes Experiment. Es zeigt sich ein gutes Verhalten insofern als kein Fluss mit entgegengesetztem Vorzeichen detektiert wird.

Das gleiche Experiment mit realem Bildmaterial zeigt Abbildung A.3. Zusätzlich wird das Bildmaterial, wenn mit VD gekennzeichnet, mit einer vertikalen Durchschnittsbildung vorverarbeitet. Das Verhalten bezüglich größerer optischer Flüsse als dem Suchradius zeigt sich hier sogar besser, als mit synthetischen Daten, da stets die Geschwindigkeit ausgegeben wird, die der zu detektierenden am nächsten liegt. Das Verhalten des Algorithmus bei Rauschen zeigt Abbildung A.4. Selbst bei starkem Rauschen mit SNR $= 3,2$ beträgt die Abweichung der Ausgabe maximal 1 *Pixel*/*Frame*.

Durch die Anwendung eines Weichzeichnungsfilters auf die Zeile der vertikalen Durchschnitte des Bildes lässt sich die Robustheit gegenüber Rauschen verbessern,

wie Abbildung A.5 zeigt. Gegenüber Helligkeits- und Kontrastreduzierung ist er weitestgehend unanfällig (A.6). Keinen Einfluss auf die Ausgabe des Algorithmus hat die Anzahl kontrastreicher Kanten, wie Abbildung A.7 zeigt.

Die weiteren Vergleiche werden mit einer Suchumgebung von $K = 11$, vertikaler Durchschnittsbildung und einem Weichzeichner mit $R = 10$ durchgeführt.

5.3.2. First-Order

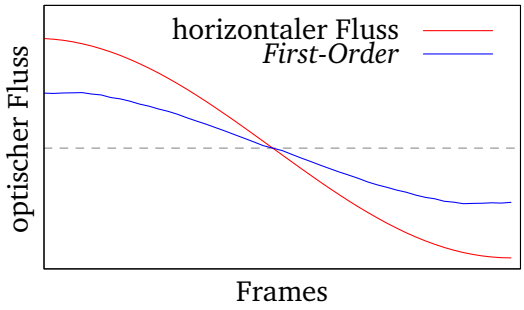

Abbildung 5.11: *Charakteristische Ausgabe des* First-Order-*Algorithmus*

In Abbildung A.8 ist die Ausgabe des *First-Order*-Algorithmus für das synthetische Bild bei verschiedenen Perioden T dargestellt. Es zeigt sich, dass die Qualität der Ausgabe mit der Länge der Periode T des Sinusoiden steigt. Insbesondere gilt das für größere optische Flüsse. Das ist klar, da bei $T = 10$ ein optischer Fluss von $u = 5$ bereits ein gesamtes Bildmerkmal (Intensitätsmaximum, bzw. -minimum) überspringt, wodurch nicht einmal mehr die Richtung des optischen Flusses ermittelt werden kann. Das lässt die Schlussfolgerung zu, dass der Algorithmus bei wenigen, dafür stark weichgezeichneten Bildmerkmalen am besten funktioniert. Für das natürliche Bildmaterial ist daher zu erwarten, dass der Radius einer Weichzeichnung die Qualität der Ausgabe ebenfalls stark beeinflusst.

Abbildung A.9 zeigt zunächst den Vergleich zwischen Differenzierung mit und ohne $2 \times 2 \times 2$-Durchschnittsbildung nach [HS80]. Schon dabei ist zu sehen, dass die Ausgabe mit Glättung der Ableitungen verbessert ist. Abbildung A.10 zeigt den Einfluss einer Weichzeichnung auf den Algorithmus. Wie erwartet wird die Ausgabequalität durch eine größere Weichzeichnungsumgebung besser. Allerdings scheint für dieses Bildmaterial ein Optimum bereits bei $R = S = 5$ erreicht. Für die folgenden Untersuchungen wird daher die Weichzeichnung mit $R = S = 5$ genutzt.

Abbildung A.11 zeigt, dass sich der Algorithmus gegenüber Rauschen und Kontrastreduzierung recht stabil erweist. Jedoch wird die Anfälligkeit in Bezug auf Helligkeitsschwankungen sichtbar. Durch Anwendung des Helligkeitsdurchschnitts als

Schwellwert auf das Bild mit anschließender Weichzeichnung lassen sich die Helligkeitseinflüsse verringern, wie Abbildung A.12 zeigt. Allerdings wird diese Helligkeitstoleranz durch einen hohen Berechnungsaufwand erkauft. Die Weichzeichnung, die zum Beispiel zu den Ergebnissen aus Diagramm A.10 führt, kann durch Unschärfe im Objektiv erzeugt werden und muss nicht berechnet werden. Da bei A.12 die Weichzeichnung erst nach der Schwellwertbildung ausgeführt wird, muss diese durch den implementierten Algorithmus selbst durchgeführt werden, was bei großen Umgebungen ($R = S = 25$) stark die Berechnungszeit erhöht.

Für den Vergleich mit den anderen Algorithmen werden Weichzeichnung mit Gleichverteilung und $R = S = 5$ und Glättung der Ableitung genutzt.

5.3.3. Zeilen-First-Order

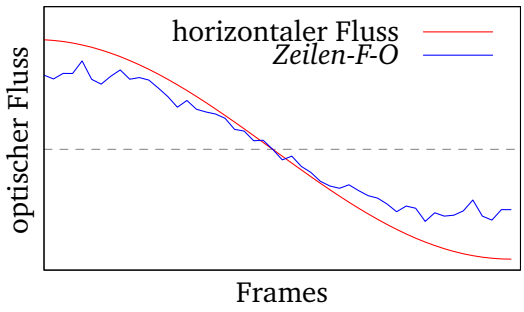

Abbildung 5.12: *Charakteristische Ausgabe des* Zeilen-First-Order-*Algorithmus.*

Abbildung A.14 zeigt zunächst wieder die Ausgabe des Algorithmus mit künstlichem Bildmaterial. Im Vergleich zur zweidimensionalen Variante wird die Ausgabe erst bei erheblich größeren Perioden des Sinusoiden befriedigend. In Kapitel 5.4.1 ist ersichtlich, dass Zeilen-Algorithmen ohne vertikale Durchschnittsbildung kaum optische Flüsse quer zu ihrer Detektionsrichtung verkraften. Daher wird dieser Algorithmus von vornherein mit entsprechender Vorverarbeitung untersucht. Abbildung A.15 zeigt den Einfluss einer Weichzeichnung im Anschluss an die vertikale Durchschnittsbildung. Anders als das Experiment mit synthetischem Bildmaterial vermuten lässt, scheint ein Optimum der Ausgabe bei einer Weichzeichnungsumgebung von $R = 10$ schon erreicht und wird daher für die folgenden Untersuchungen genutzt. Auch dieser Algorithmus ist besonders empfindlich bei Helligkeits- und darüber hinaus auch bei Kontrastschwankungen, wie Abbildung A.16 zeigt. Die Datenreihe „*Zeilen-F-O* SW $p_h = -50\,\%$“ zeigt, dass auch hier die Anwendung eines Schwellwertes erheblich zur Verringerung der Auswirkungen von Helligkeitsschwankungen führt. Im Vergleich zur zweidimensionalen Variante braucht die Weich-

zeichnung, da sie erst nach der vertikalen Durchschnittsbildung durchgeführt wird, lediglich in einer Dimension erfolgen. Daher ist der Berechnungsaufwand dafür geringer als bei zwei Dimensionen.

Für folgende Vergleiche wird der Algorithmus mit Schwellwert-, vertikaler Durchschnittsbildung und Weichzeichnung mit Gleichverteilung und $R = 10$ genutzt.

5.3.4. Lucas-Kanade

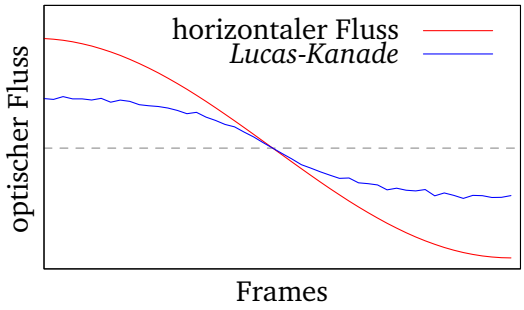

Abbildung 5.13: *Charakteristische Ausgabe des* Lucas-Kanade-*Algorithmus*

Analog zum *First-Order*-Algorithmus wird die Ausgabe des Algorthmus bei synthetischem Bildmaterial mit größerer Periode T besser (A.18).

Für reale Bilddaten zeigt sich bei der Ermittlung der Ergebnisse des überbestimmten Gleichungssystems anhand eines 2×2-Umfeldes in Abbildung A.19 ein Optimum der Weichzeichnung bei $R = S = 10$. Die Umgebungsgröße, die zum überbestimmten Gleichungssystem führt, scheint kaum Einfluss auf die Ausgabe zu haben, wie Abbildung A.20 verdeutlicht. Daher genügt, zu gunsten der Laufzeit, die Wahl einer kleinen 2×2 Umgebung. Auch bei diesem Algorithmus zeigt sich in A.21 die größte Anfälligkeit bei Helligkeitsänderungen.

Wie bei dem *First-Order*-Algorithmus wird auch hier auf eine Schwellwertbildung vorher verzichtet und für den weiteren Vergleich die Parameter $n = 2 \times 2$ und $R = S = 10$ gewählt.

5.3.5. Uras et al.

Der Uras-Algorithmus zeigt bei dem ersten Test an synthetischem Bildmaterial (A.23) erst einmal keine richtige Funktionsweise. Das könnte den Schluss zulassen, dass ein Fehler in der Implementierung oder ähnliches vorliegt. Allerdings zeigt sich bei dem Test an realem Bildmaterial (A.24) doch die erwartete Funktionsweise, wie die Kurve „*Uras* $R = S = 20$“ zeigt. Dieser Test wurde ohne die von Urase et al.

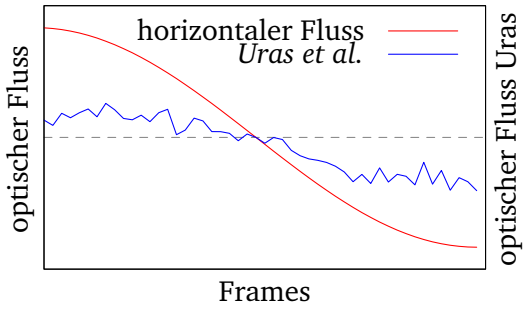

Abbildung 5.14:
Charakteristische Ausgabe des Uras-et-al-*Algorithmus*

vorgeschlagene Selektierung der Messwerte anhand des Kriteriums $\frac{|\mathbf{M}\nabla I|}{|\nabla I_t|} \ll 1$ (M-Selektierung) und der Condition-Number (C-Selektierung) bzw. Determinaten (D-Selektierung) durchgeführt. Abbildung A.25 zeigt die Einflüsse der verschiedenen Selektierungen. Die Selektierung anhand des M-Kriteriums scheint zu den besten Ergebnissen zu führen. Änderungen an der Umgebungsgröße (8×8-Teilbildbereich) oder der Anzahl der ausgewählten Werte (8 aus dem 8×8-Bereich) haben keine Verbesserung gebracht.

Die Untersuchung des Algorithmus auf Störungen des Bildmaterials zeigt in Abbildung A.26 und A.27 den starken Einfluss von Rauschen und Kontrastschwankungen. Gegenüber Helligkeitsänderungen (A.28) ist er dafür realtiv stabil.

Für die weiteren Untersuchungen werden die Parameter: 8 Werte aus einem 8×8-Bildbereich durch M-Selektierung mit einer $R = S = 20$-Weichzeichnung als Vorverarbeitung gewählt.

5.3.6. Fleet und Jepson

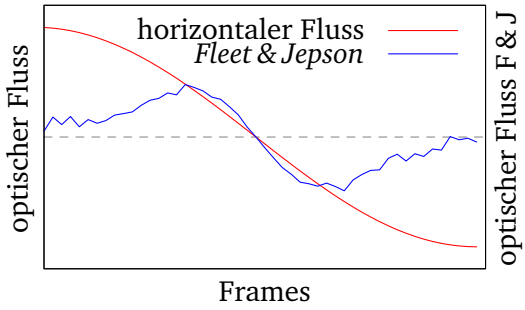

Abbildung 5.15:
Charakteristische Ausgabe des Fleet & Jepson-*Algorithmus*

Die Untersuchung und Abstimmung der Parameter des *Fleet-und-Jepson*-Algorithmus gestaltet sich schwerer, als bei den bisherigen Verfahren. Es müssen die drei Parameter θ, T und σ aufeinander abgestimmt werden. Alle Kombinationen dieser drei Werte aufzustellen und den besten zu wählen, ist aufgrund der dabei entstehenden

hohen Anzahl von Messkurven kaum möglich. Deshalb werden die besten Belegungen für θ, T und σ in dieser Reihenfolge ermittelt, wobei die noch nicht bestimmten mit einem sich im Laufe der Untersuchungen als gut erwiesenen Wert konstant gehalten werden. Abbildung A.30 zeigt wieder die gundsätzliche Funktionsfähigkeit mit den Parametern $T = 10$, $\theta = 0$ und $\sigma = 3$.

Zunächst soll untersucht werden, wie sich Orientierung θ der Basisfunktion auf die Ausgabe des Algorithmus auswirkt. Die beiden anderen Parameter werden mit $T = 5$ und $\sigma = 3$ konstant gehalten. Abbildung A.31 zeigt dieses Experiment. Die weitaus besten Ergebnisse lassen sich offenbar erzielen, wenn die Basisfunktion bei einem horizontalen optischen Fluss ebenfalls horizontal oder vertikal ausgerichtet ist. Die Winkel dazwischen führen zum schlechtesten Ergebnis. Da der auftretende optische Fluss in seiner Richtung jedoch nicht kontrollierbar ist, muss vom schlechtesten Fall ausgegangen werden. Aus diesem Grund wird die Untersuchung mit $\theta = 2\pi/3$ fortgesetzt. Als nächstes wird in Abbildung A.32 die beste Belegung für T gesucht. $\sigma = 3$ bleibt wieder konstant. $T = 10$ führt hier zu den besten Ausgaben. Zuletzt ist in Abbildung A.33 zu beobachten, dass die Veränderung von σ die Ausgabe nicht stark beeinflusst. Dennoch scheint $\sigma = 3$ zu dem besten Ergebnis zu führen.

Wie in [FJ90] angestrebt, zeigt der Algorithmus tatsächlich eine sehr gute Stabilität gegenüber Rauschen und Kontrastschwankungen, wie Abbildung A.34 zeigt. Auch Helligkeitsänderungen, führen zunächst zu immer noch guten Ergebnissen. Lediglich bei $p_h = -50\,\%$ werden die Abweichungen größer.

5.3.7. Reichardt-Detektor

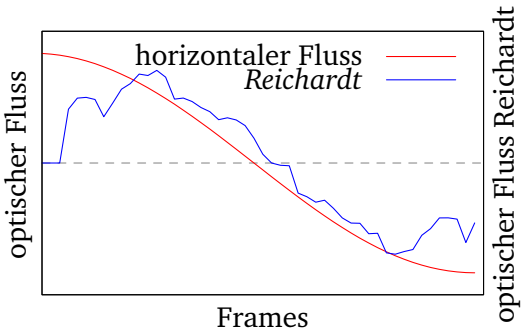

Abbildung 5.16: *Charakteristische Ausgabe des* Reichardt-*Algorithmus*

Der Test des Reichardt-Detektors an synthetischem Bildmaterial in A.36 zeigt dessen Abhängigkeit von Anzahl und Steilheit der Kanten. Je mehr im Bildmaterial vor-

handen sind, desto ausgeprägter ist die Ausgabe. Die Änderung des Parameters α führt zu keiner qualitativen, sondern nur zu einer quantitativen Änderung der Ausgabe, wie in Abbildung A.37 demonstriert. Für die weiteren Untersuchungen wird $\alpha = 0,5$ gewählt.

A.38 zeigt den Einfluss unterschiedlicher Vorverarbeitungsschritte auf die Ausgabe des Detektors. Die Auswahl fällt auch im Hinblick auf die Toleranz gegenüber störendem vertikalen optischen Fluss (siehe 5.4.1) auf Schwellwertbildung mit dem Helligkeitsdurchschnitt als Schwellwert und anschließender vertikaler Durchschnittsbildung.

In Abbildung A.39 ist zu sehen, dass Kontraständerungen nahezu keine Auswirkungen auf die Ausgabe haben, Rauschen und Helligkeitsveränderungen, trotz der gewählten Vorverarbeitungsschritte, diese jedoch sehr stark beeinflussen.

5.3.8. Neuronenzeile

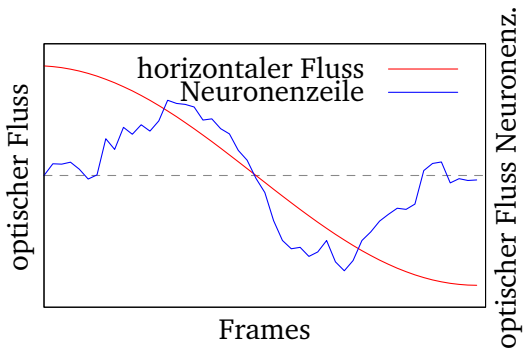

Abbildung 5.17:
Charakteristische Ausgabe der zweilagigen Neuronenzeile

Zunächst werden sowohl die einlagige als auch die zweilagige Neuronenzeile mit synthetischem Bildmaterial getestet. Dieser Test wird mit den Gewichten für komplexe künstliche Reize und für natürliche Reize, wie sie in [Wol07] evolviert wurden, durchgeführt. Die Abbildungen A.41 und A.42 zeigen die Ergebnisse für komplexe künstliche Reize. Es ist zu erkennen, dass beide neuronalen Netze wohl auf kurze steile Kanten, ähnlich dem Reichardt-Detektor, zu reagieren scheinen. Auffällig ist weiterhin, dass die Netze nicht tolerant gegenüber größeren optischen Flüssen sind, sich die detektierte Richtung also umkehrt.

Betrachtet man die Ergebnisse mit den Gewichten für natürliche Reize in A.43 und A.44, stellt sich heraus, dass die zweilagige Neuronenzeile bei einem breiteren Spektrum an Eingangsdaten bessere Ausgaben liefert, als die einlagige Neuronenzeile.

Wie bei den anderen Zeilen-Algorithmen besteht auch bei diesem die Notwendigkeit einer vertikalen Durchschnittsbildung. Die Einflüsse der einzelnen Vorverarbeitungsschritte bei realem Bildmaterial zeigen die Abbildungen A.45 und A.46. Für beide Netze scheint die vertikale Durchschnittsbildung mit anschließender Anwendung eines Schwellwerts die beste Wahl zu sein. Grundsätzlich zeigt die zweilagige Neuronenzeile gegenüber der einlagigen die deutlich besseren Ergebnisse. Die Auswahl fällt daher auf die zweilagige Neuronenzeile mit Gewichten für natürliche Reize und die genannten Vorverarbeitungsschritte.

Bei Störungen des Bildmaterials zeigt die zweilagige Neuronenzeile in A.47 wieder Parallelen zum Reichardt-Detektor. Kontraständerungen wirken sich kaum aus, Rauschen und Helligkeitsänderungen dagegen stark.

5.3.9. Slow-Feature-Analysis

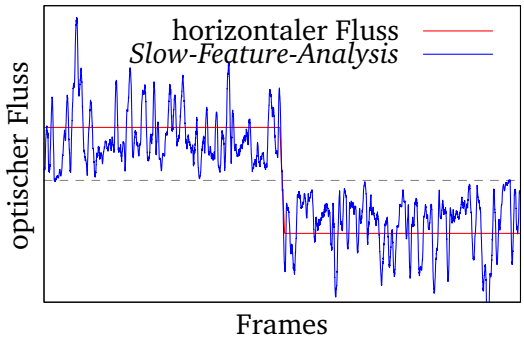

Abbildung 5.18: *Charakteristische Ausgabe der* Slow-Feature-Analysis

Die *Slow-Feature-Analysis* kann, wie in Kapitel 4.5.3 beschrieben, sich langsam ändernde Informationen aus einem Strom von Eingangsdaten herausfiltern. Als Eingangsdaten werden hier, wie auch bei den anderen Zeilenalgorithmen, eine bestimmte Anzahl an Pixeln einer Zeile des Bildes genutzt. Jedes Bild der Bildfolge erzeugt dabei einen neuen Datensatz für die Analyse. Alle Eingangsdaten werden nach den Kriterien in Kapitel 4.5.3 normiert. Ziel der Testdatengenerierung muss es sein, dass der auftretende optische Fluss die Informationskomponente mit der geringsten Änderung ist. Daher wird er lediglich einmal bei der Hälfte der Bildfolge geändert:

$$u(\mathbf{x},t) = \begin{cases} 1 & t < \frac{l}{2} \\ -1 & sonst \end{cases}$$

und $v(\mathbf{x},t) = 0$. Die dadurch entstehenden Daten werden mit Hilfe der *Slow-Feature-Analysis* während der Lernphase ausgewertet und es muss sich zeigen, ob der Aus-

gang y_0 als langsamste Komponente mit dem optischen Fluss korrespondiert. Tut er das, besteht eine große Wahrscheinlichkeit, dass dieser Ausgang tatsächlich den optischen Fluss widerspiegelt. Zur Überprüfung dessen wird die bestimmte Funktion g auf Testdaten angewendet, die sich in ihren Eigenschaften lediglich in dem optischen Fluss unterscheiden. Hier werden dazu aus dem gleichen Bildmaterial Daten mit entgegengesetztem Fluss erzeugt:

$$u(\mathbf{x}, t) = \begin{cases} -1 & t < \frac{l}{2} \\ 1 & sonst \end{cases} .$$

Abbildung A.49 zeigt die Ausgabe der *Slow-Feature-Analysis* bei zwei Sensoreingängen und ihrer Erweiterung um einen Zeitschritt. Das hinzugefügte Rauschen ist nötig, da die Koeffizienten der Funktionen g_j sonst nicht berechnet werden können. Es zeigt sich zwar, dass der optische Fluss den Ausgang in seiner Charakteristik ändert, er jedoch in seinem Auftreten nicht richtig angezeigt wird. Durch eine quadratische Erweiterung der Sensorsignale nach der zeitlichen Erweiterung erhält man die Ausgabe, die Abbildung A.50 zeigt. Der optische Fluss wird hier schon deutlich besser repräsentiert. Der Versuch mit Erweiterung der Sensorsignale um zwei Zeitschritte brachte keine wesentliche Änderung der Ausgabe, wie Abbildung A.51 zeigt. Auch führten Änderungen bei den Bilddaten, beziehungsweise bei der Anzahl der Sensoreingänge, zu keinen wesentlich anderen Ausgaben oder Erkenntnissen. Ein weiteres Experiment ist, inwiefern die Ausgabe der gelernte Funktion g_0 robust gegenüber Bilddaten ist, die nicht Teil der Lernphase sind. Dafür wurden in der Lernphase ein Sinosiod mit der Periode $T = 10$ und in der Ausführungsphase mit $T = 11$ genutzt. Abbildung A.52 zeigt, dass selbst diese verhältnismäßig kleine Änderung bereits dazu führt, dass der optische Fluss nahezu nicht mehr erkannt werden kann. Die Extraktion des optischen Flusses aus den Sensordaten gestaltet sich bei realen Bilddaten schwerer. Mit einem Umfang von 50 Datensätzen können hier keine brauchbaren Ergebnisse erzielt werden. Mehr als 6000 Datensätze führen zu der Ausgabe, die in Abbildung A.53 zu sehen ist. Wieder scheint es möglich den optischen Fluss in seiner Richtung, nicht jedoch im Betrag, richtig zu bestimmen. Abbildung A.54 zeigt das Experiment, inwieweit die trainierte Funktion g_0 aus Bildmaterial, welches im Vergleich zu dem Bildmaterial aus der Lernphase leicht verändert wurde, dennoch in der Lage ist, den optischen Fluss zu extrahieren. Im vorliegenden Fall wurde eine Kontrastreduzierung von $p_k = 30\,\%$ vorgenommen. Wie bereits bei den synthetischen Experimenten abzusehen war, zeigen die gefundenen Funktionen g_j

kaum Toleranz gegenüber Eingangsdaten, die nicht in der Lernphase aufgetreten sind. Es ist also nötig die Kontrastschwankungen gegenüber denen der Algorithmus sich als stabil erweisen soll, in die Lernphase mit einfließen zu lassen. Im folgenden Experiment werden die Bilddaten mit einer sinusförmigen Kontrastreduzierung zwischen $p_k = 0$ und $p_k = 30\,\%$ erzeugt. Als Periode wird zunächst 200 Pixel gewählt. Abbildung A.54 zeigt, dass unter diesen Vorraussetzungen der optische Fluss nicht bestimmt werden kann. Auch eine große Anzahl verschiedener Perioden der Kontraständerung führte zu keinem besseren Ergebnis. Die *Slow-Feature-Analysis* zur Analyse des optischen Flusses gegenüber Schwankungen in den Eigenschaften des Bildmaterials stabil zu gestalten, erfordert offenbar mehr Aufwand, als im Rahmen dieser Arbeit erbracht werden kann. Für die weiteren Vergleiche wird dieser Algorithmus daher nicht berücksichtigt.

5.4. Einfluss der Flussarten

Verschiedene Arten des optischen Flusses dürfen sich gegenseitig nicht beeinflussen. Beispielsweise darf ein auftretender Fluss in vertikaler Richtung nicht die Ausgabe des horizontalen Flusses beeinflussen. Gleiches gilt für Flussfelder, die eine Rotation um das Zentrum des Bildausschnitts darstellen. Diese beiden Fälle werden in diesem Kapitel untersucht.

5.4.1. Vertikaler optischer Fluss

Ein vertikaler optischer Fluss darf keinen Einfluss auf die Detektion des horizontalen haben. Das folgende Experiment zeigt die Ausgabe der Algorithmen für einen konstanten optischen Fluss von 1 Pixel/Frame und einem überlagterten vertikalen Fluss zwischen einem und 5 Pixel/Frame. Die Diagramme A.55 und A.56 zeigen die Ergebnisse. Bis auf die Algorithmen *Zeilen-First-Order* und *Fleet & Jepson* sind die Algorithmen weitestgehend stabil gegenüber einem optischen Fluss in vertikaler Richtung, die beiden genannten mit ihren Einstellungen jedoch nicht.

Die Notwendigkeit der vertikalen Durchschnittsbildung soll ein weiteres Experiment verdeutlichen. Dazu werden alle Zeilenalgorithmen mit den bestimmten Parametern, jedoch ohne diesen Vorverarbeitungsschritt, ebenfalls auf den Einfluss eines vertikalen Flusses untersucht. Wie Diagramm A.57 zeigt, ist keiner der Zeilen-Algorithmen bei dieser Störung des Flusses mehr in der Lage, den horizontalen Fluss zu detektieren.

5.4.2. Rotation

Die Betrachtungen in Kapitel 2.3.4 zeigen, dass sich diese bezüglich des horizontalen und vertikalen optischen Flusses auslöschen. Inwiefern das für die hier vorgestellten Algorithmen mit ihren Vorverarbeitungsschritten gilt, soll an folgendem Experiment untersucht werden. Dafür wird ein konstanter horizontaler optischer Fluss mit einer Stärke von 1 *Pixel/Frame* mit einer variablen Rotation um den Bildausschnittsmittelpunkt zwischen einem und fünf Grad pro Teilbild überlagert. Der Idealfall ist, dass diese Rotation die Ausgabe der Algorithmen nicht beeinflusst. Abbildungen A.58 und A.59 zeigen die Ergebnisse des Experiments. Der *Zeilen-First-Order*-Algorithmus und die Neuronenzeile reagieren am stärksten auf die Rotation, alle anderen Algorithmen reagieren nicht sehr stark auf diese.

6. Mikrocontroller-Implementierung

6.1. Hardwareaufbau

In diesem Kapitel wird die hardwarenahe Implementierung einer der beschriebenen Algorithmen erläutert. Für die Umsetzung steht der Zeilenlichtsensor TSL3301 der Firma TAOS zur Verfügung. Angesteuert wird dieser von dem ARM-basierten Prozessor STM32 mit einer Taktrate von 72 MHz. Das genutzte Prozessorboard ist ein *AccelBoard3D* der humanoiden Roboterplattform Myon. Näher beschrieben ist es in der Arbeit [Ben10]. Der Mikrocontroller kommuniziert mit dem Zeilensensor über eine synchrone serielle Schnittstelle. Der detektierte optische Fluss wird über eine asynchrone serielle Schnittstelle zum PC gesendet. Den schematischen Aufbau der Hardware zeigt Abbildung 6.1.

Die serielle Schnittstelle des TSL3301 beruht nicht auf der Übertragung ganzer, beispielsweise 8 bit langer, Datenwörter. Weiterhin muss, über die Übertragung der eigentlichen Kommandos an den Sensor hinaus, auf eine zusätzliche Generierung von Takten geachtet werden, da die interne Logik des TSL3301 auf diese angewiesen ist. Diese Faktoren erfordern die Umsetzung der Datenübertragung per softwaregesteuerten universellen I/O-Leitungen (*Bit-Banging*).

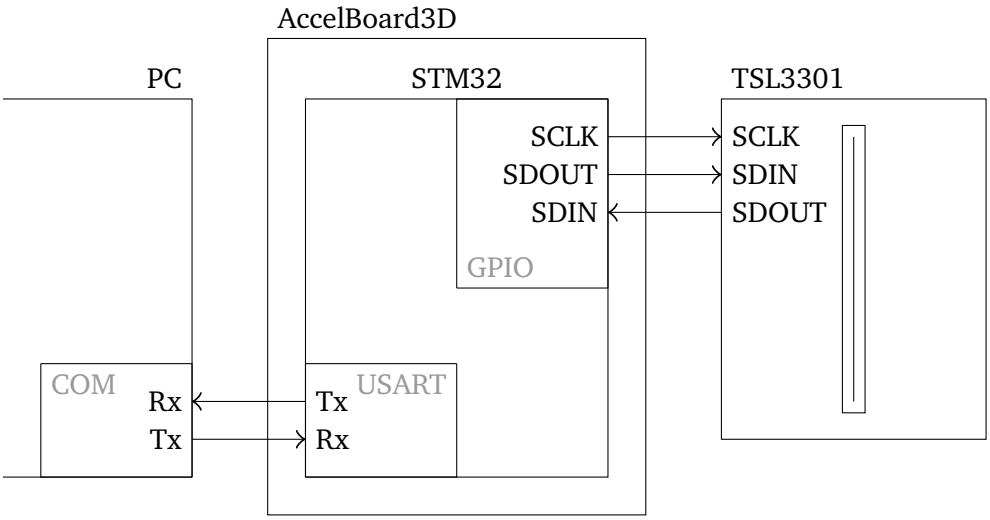

Abbildung 6.1.: *Anordnung der elektronischen Komponenten im Testaufbau*

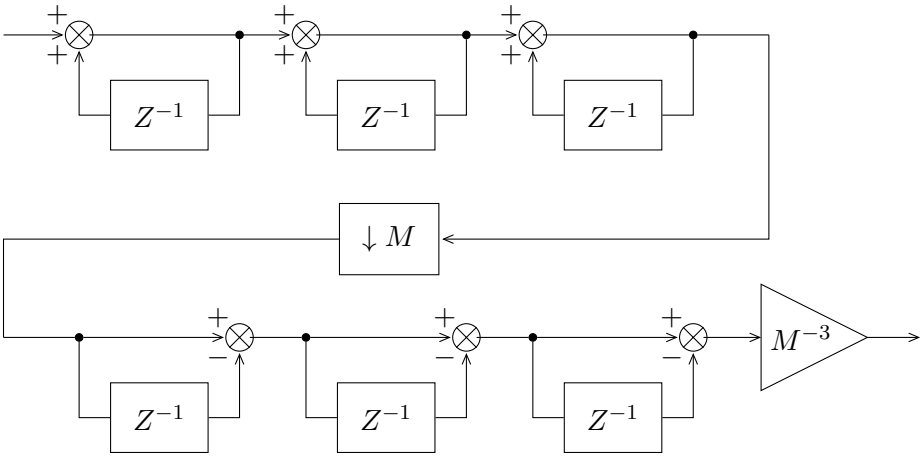

Abbildung 6.2.: *Dreistufiger Sinc-Filter*

6.2. Software

Die Auswahl der Verfahren für eine Implementierung auf dieser Hardware beschränkt sich auf die vorgestellten Zeilen-Algorithmen. Dazu zählen das *Zeilen-First-Order*-Verfahren, eindimensionales *Block-Matching*, der *Reichardt*-Detektor und die beiden Neuronenzeilen. Das *Block-Matching*-Verfahren, wie es hier vorgestellt wurde, gibt den detektierten optischen Fluss nur diskret aus. Es eignet sich daher nicht für die Einkopplung in analoge Rechenstrukturen, wie es beim Myon der Fall ist. Die anderen genannten Algorithmen geben den Fluss als kontinuierliche Größe aus. Die Ergebnisse in Kapitel 5 haben gezeigt, dass der *Zeilen-First-Order*-Algorithmus den optischen Fluss besser detektiert, als der *Reichardt*-Detektor und die Neuronenzeile. Implementiert wurde also der *Zeilen-First-Order*-Algorithmus. Dem vorausgeschaltet ist ein Weichzeichner mit Gleichverteilung und einer Umgebung von $R = 5$ (siehe Kapitel 3.4). Die Durchlaufzeit des implementierten Algorithmus beträgt ca. 400 μs, gibt also 2500 mal pro Sekunde einen Wert aus. Die Ausgabe ist jedoch zum Teil mit deutlichem Rauschen behaftet. Zur Glättung des Signals und Reduzierung der Ausgabefrequenz wird ein Sinc^3-Filter eingesetzt ([OH03], [GIT95], [Pre00]). Die Ausgabefrequenz wird um den Faktor $M = 25$ geteilt, beträgt nach der Filterung also 100 Hz. Aufgebaut ist der Filter aus drei sequenziellen Integratoren, einem Schalter zur Herabsetzung der Ausgabefrequenz, gefolgt von drei Differentiatoren und einer Divison. Den vereinfachten Aufbau des Filters zeigt Abbildung 6.2. $H(z)$ ist die

Übergangsfunktion des Sinc^3-Filters:

$$H_M(z) = \left(\frac{1}{M} \frac{1 - z^{-M}}{1 - z^{-1}} \right)^3 .$$

Den Quelltext des Programms in Pseudocode zeigt Ansicht B.1.

6.3. Experimentalaufbau

Die Ausgabe des implementierten Algorithmus soll mit den Daten aus der Simulation in Kapitel 5.3.3 verglichen werden. Dazu ist es notwendig, einen sinusförmigen optischen Fluss eines Bildmusters vor der Linse der Kamera zu erzeugen. Für diesen Zweck wurde ein Pendel mit einem halbkreisförmigen Schirm gebaut. Abbildung 6.3 zeigt den Aufbau im nicht ausgelenkten, als auch im ausgelenkten Zustand. Die Kamera filmt von unten aus etwa 10 cm Entfernung den Schirm ab, welcher mit unterschiedlichen Bildmustern bestückt werden kann. Für eine definierte Beleuchtung wird der abgefilmte Bildausschnitt mit einer Lichtquelle beleuchtet. Bei der Pendelbewegung handelt es sich um eine gedämpfte Schwingung. Der Dämpfungseffekt des Pendels ist bei der Beurteilung einer Schwingungsperiode, wie auch in Kapitel 5 geschehen, vernachlässigbar gering. Natürlich wird auch stets die gleiche Schwingung für die Auswertung genutzt. Damit sind die Amplituden der Ausgaben miteinander vergleichbar.

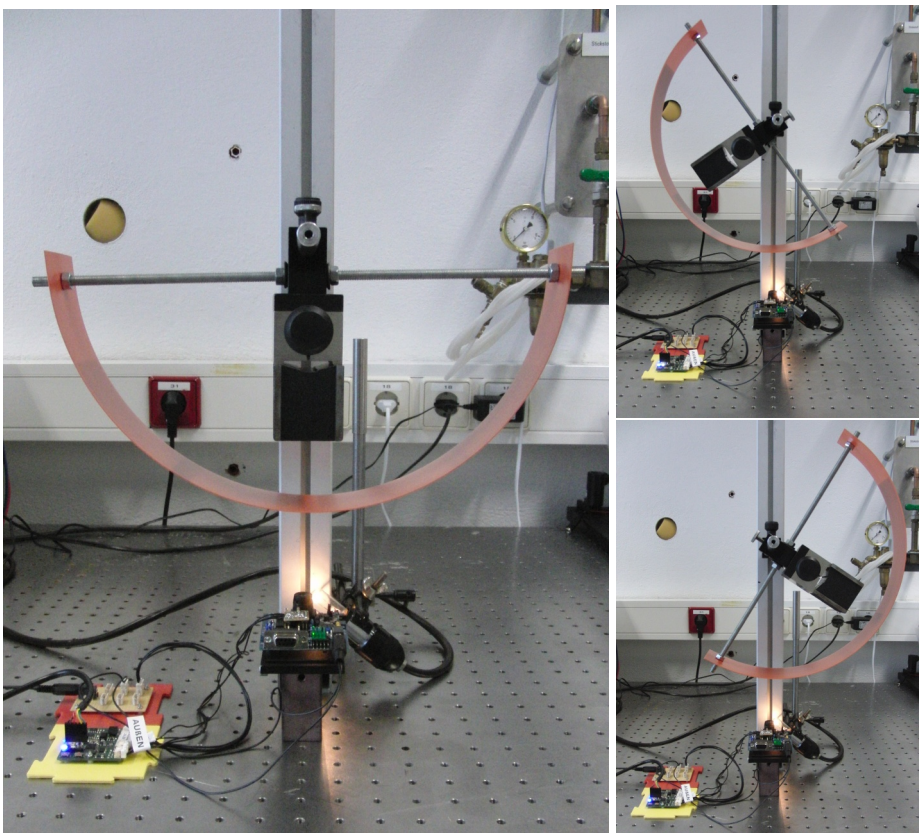

Abbildung 6.3.: *Experimentalaufbau für die Untersuchung des Implementierten Algorithmus*

6.4. Experimentalergebnisse

Das erste Experiment wird wieder mit synthetischen Bildmustern durchgeführt. Dafür wurden sinusoide Muster mit unterschiedlicher Periodenlänge auf den Pendelbildschirm aufgebracht. Ein Muster hat eine Periodenlänge von 60 Pixel, das andere 200 Pixel bei einer Druckauflösung von $300\,Pixel/Inch$. Durch die genutzte Optik und den Abstand der Kamera vom Bildmuster erzeugt das sinusiode Muster mit 5 und 15 Pixel Periodenlänge auf dem Sensor. Diagramm 6.4 zeigt die Ausgaben für beide Periodenlängen. In rot ist eine Referenzkurve eingezeichnet, die den Vergleich der Kurven erleichtern soll. Beide Kurven liegen in ihrer Repräsentation des tatsächlichen optischen Flusses sehr nah beieinander, mit leichten Vorteilen bei der größeren Periodenlänge des Bildmusters. Das legt wieder die Vermutung nahe, das ein unscharfes Bild die Ausgabe positiv beeinflussen kann.

Bei dem nächsten Bildmuster, was untersucht wurde, handelt es sich wieder um das Foto, welches schon in Kapitel 5.1.2 Grundlage der Untersuchung an realen Bilddaten wurde. Abbildung A.11 zeigt die Ausgaben des Algorithmus für dieses Bildmaterial. Die beiden Beleuchtungszustände h_1 und h_2 wurden durch Dimmen der Lichtquelle am Experiment erzeugt. Mangels einer genauen Lichtstärkemessung, können diese nur subjektiv in hell ($p_h = 0$), dunkler ($p_h = h_1 < 0$) und deutlich dunkler ($p_h = h_2 \ll h_1$) eingeteilt werden. Das kontrastreduzierte Bildmuster wurde vor dem Ausdruck digital um $p_k = 60$ reduziert. Die Kurven zeigen, dass der optische Fluss auch bei realen Bildaten gut detektiert werden kann. Die Ausgabe bei moderaten Helligkeitsänderungen ist weitestgehend stabil, bei stärkeren ändert sich die Amplitude der Ausgabe deutlich. Gleiches gilt für stärkere Kontraständerungen.

Das nächste Experiment zeigt den Einfluss von senkrecht zur Detektionsrichtung verlaufendem optischen Fluss auf die Ausgabe. Dazu wurde die Kamera quer zur Pendelrichtung gedreht. Diagramm 6.6 zeigt die Ergebnisse des Experiments für verschiedene Objektivjustierungen. Die Ausgabe müsste idealerweise stets null sein. Da es bei einem Zeilensensor jedoch nicht möglich ist, wie in Kapitel 5.4.1 vorgeschlagen, einen vertikalen Durchschnitt zu bilden, wäre eine Ausgabe wie in Abbildung A.57 zu erwarten gewesen. Dass sich die hier vorliegende Kurve wesentlich stabiler gegenüber vertikalem Fluss zeigt, liegt ganz offensichtlich an der sehr hohen Abtastfrequenz. Damit scheint der auftretende vertikale optische Fluss in diesem Experiment nicht über $1\,\text{Pixel/Bild}$ zu liegen. Eine unschärfe im Objektiv scheint die Toleranz gegenüber vertikalem Fluss etwas zu verbessern. Die Reproduzierbarkeit der Ergebnisse ist mit dem Versuchsaufbau sehr gut. Allerdings schließt das keine

systematischen Fehler aus. Ist die Achse des optischen Sensors nicht genau senkrecht oder parallel zur Pendelachse des Musters, hat das bereits deutliche Einflüsse auf die Ausgabe dieses und die anderen Experimente.

Bei der Wahl der Beleuchtung hat sich gezeigt, dass Kunstlichtquellen, deren Leuchtcharakteristik durch die 50 Hz Netzfrequenz beeinflusst ist, periodische Störungen der Ausgabe hervorrufen. Aus diesem Grund ist eine Anpassung des Teilungsfaktors M ratsam. Er sollte so gewählt werden, dass die Frequenz von 50 Hz herausgefiltert wird. In diesem Fall ist $M = 50$ also eine bessere Wahl.

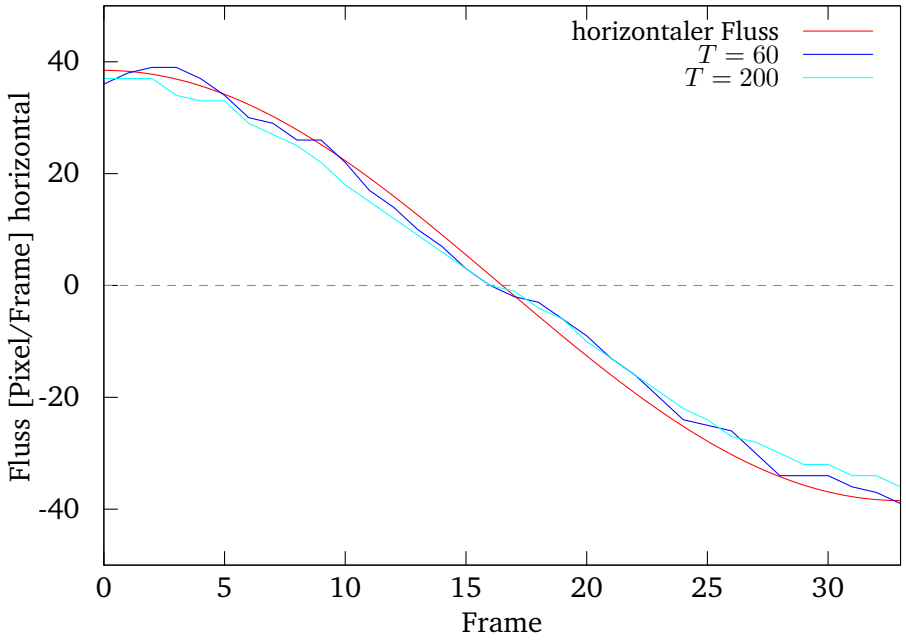

Abbildung 6.4.: *Optischer Fluss des implementierten Algorithmus mit synthetischem Bildmaterial für unterschiedliche Perioden T des Sinusoiden*

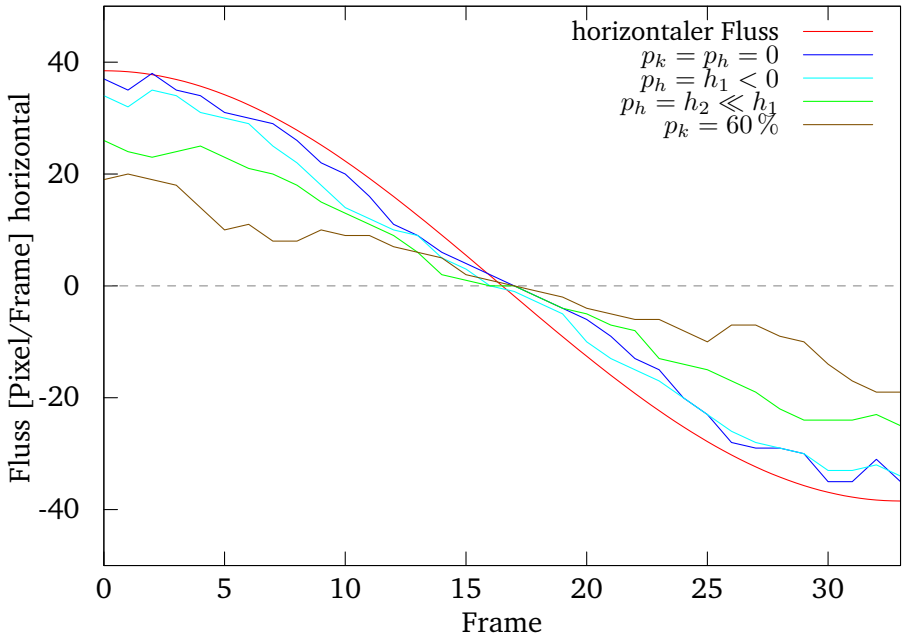

Abbildung 6.5.: *Einfluss von Kontrast- und Helligkeitsreduzierung auf die Ausgabe des implementierten* Zeilen-First-Order-*Algorithmus*

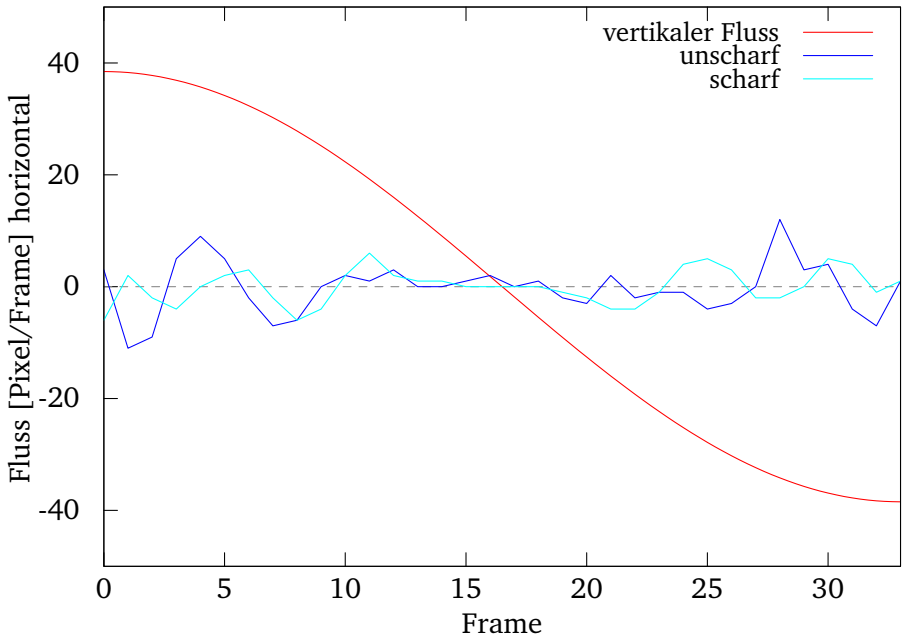

Abbildung 6.6.: *Einfluss von optischem Fluss senkrecht zur Detektionsrichtung des Sensors*

7. Vergleich

7.1. Vergleichskriterien

Will man die vorgestellten Algorithmen miteinander vergleichen, muss dies anhand bestimmter Anforderungen erfolgen. Je nach Verwendungszweck und Einsatzszenario der Algorithmen ergeben sich die verschiedensten Vergleichskriterien. Eine mögliche Auswahl stellen die folgenden dar:

- Genauigkeit der Ausgabe des optischen Flusses, das heißt eine möglichst lineare Abbildung der tatsächlich herrschenden Bewegung,
- Anfälligkeit gegenüber Störungen der Qualität des vorliegenden Bildmaterials, wie zum Beispiel Rauschen, Helligkeits- und Kontrastschwankungen,
- Einfluss von optischen Fluss aus anderen Richtungen als der eigentlichen Detektionsrichtung auf die Ausgabe,
- Reaktivität, Berechnungsgeschwindigkeit,
- Adaptionfähigkeit auf verschiedene Umwelt- beziehungsweise Bildcharakteristika.

7.2. Bewertung

Genauigkeit der Ausgabe

Unterliegt das Bildmaterial keinen Störungen, wie sie in Kapitel 5.1.2 beschreiben sind, sollte ein Algorithmus den tatsächlich auftretenden optischen Fluss möglichst linear abbilden. Dieser Anforderung kommen die gradientenbasierten Algorithmen erster Ordnung am nähesten. Dazu gehören das *First-Order-*, *Zeilen-First-Order-* und *Lucas & Kanade*-Verfahren. Falls es nicht auf eine kontinuierliche Ausgabe der Werte ankommt, bildet das *Block-Matching*-Verfahren den optischen Fluss fast stets ideal ab. Aufgrund des Amplitudeneinbruchs bei größeren optischen Flüssen, ordnet sich

danach erst der Algorithmus von *Fleet & Jepson* sowie der *Reichardt-Detektor* und die Neuronenzeilen an. Das Verfahren von *Uras-et-al.* und die *Slow-Feature-Analysis* erfüllen dieses Kriterium nur mangelhaft.

Anfälligkeit gegenüber Störungen

Wenn das Bildmaterial Störungen, wie sie in Kapitel 5.1.2 eingeführt werden, unterliegt, ergibt sich eine andere Bewertung. Als sehr tolerant hat sich der *Block-Matching*-Algorithmus erwiesen. Mit den gewählten Vorverarbeitungsschritten zeigen sich die gradientenbasierten Algorithmen (*First-Order*, *Zeilen-First-Order* und *Lucas & Kanade*) ebenfalls als recht robust, sieht man von Helligkeitsänderungen ab. Dieses scheint eine grundlegende Schwäche der Gradientenverfahren zu sein. In dieser Beurteilung folgen *Reichardt-Detektor* und Neuronenzeile. Keine zufriedenstellende Toleranz gegenüber den gewählten Störungen zeigt der *Uras-et-al.*-Algorithmus.

Insgesamt zeigt sich bei diesem Kriterium der große Einfluss der Vorverarbeitungsschritte, die zum Teil zu einer deutlichen Verbesserung in diesem Bezug führen. Gesondert hervorzuheben ist das Verfahren von *Fleet & Jepson*, welches, wie auch in [FJ90] beschrieben, ohne Vorverarbeitungsschritte eine gute Toleranz gegenüber Störungen des Bildmaterials zeigt.

Anfälligkeit gegenüber verschiedenen Flussarten

Ist ein Verfahren auf die Bestimmung einer translatorischen Komponente des optischen Flusses ausgelegt, sollten andere Flussarten idealerweise keinen Einfluss auf die Ausgabe haben. Bei störender Bewegung senkrecht zur Detektionsrichtung erfüllen diese Anforderung die gradientenbasierten Verfahren *First-Order* und *Lucas & Kanade*, sowie der hier vorgestellte *Block-Matching*-Algorithmus und *Reichardt-Detektor* am besten. Die anderen Verfahren zeigen eine deutlich größere Anfälligkeit gegenüber dieser Störung. Die gleiche Bewertung ergibt sich bei ungewünschten Rotationsbewegungen des Bildes.

Reaktivität

Dieses Kriterium zielt insbesondere auf die Berechnungsgeschwindigkeit der Algorithmen. Eine schnellere Berechnung führt zu einer höheren Ausgaberate und damit schnelleren Reaktion auf den herrschenden optischen Fluss. Obwohl die Laufzeiten

nicht genauer untersucht wurden (siehe 4.6), lässt sich die Aussage treffen, dass die Zeilenalgorithmen aufgrund der geringeren Datenmengen, die zu verarbeiten sind, potenziell deutlich schneller sind, als jene, die auf zweidimensionalen Daten arbeiten. Die vorgestellten neuronalen Verfahren (*Reichardt-Detektor*, Neuronenzeile) unterliegen, wegen der zugrunde liegenden Netzstruktur, stets einer festen Latenz in der Ausgabe des optischen Flusses, die von der Anzahl der Ebenen des neuronalen Netzes abhängt. Deutlich größeren Berechnungsaufwand erfordern die Verfahren von *Uras-et-al.* und *Fleet & Jepson*.

Gradientenbasierte Algorithmen sind grundsätzlich auf eine hohe Berechnungsgeschwindigkeit angewiesen, da in diesem Fall ihre Vorraussetzung, die *Helligkeitsbeständigkeits*-Bedingung (2.3.2), am besten erfüllt werden kann. Prinzipbedingt können *Block-Matching*-Verfahren auch aus Bildern mit größerem Zeitabstand den optischen Fluss bestimmen. Für die Genauigkeit der Ausgabe sind sie daher nicht auf eine schnelle Berechnung angewiesen.

Adaptionsfähigkeit

Stellt sich die Frage, inwiefern die Algorithmen Potenzial bieten, sich auf unterschiedliche Bildeigenschaften einzustellen, sind die neuronalen Verfahren hervorzuheben. Sie bieten viele Ansatzpunkte für Lernverfahren [Roj96] und damit Anpassbarkeit auf unterschiedliche Reize. Bei der *Slow-Feature-Analysis* ist die Lernphase Bestandteil des Prinzips. Alle anderen Verfahren bieten kaum Anpassungsmöglichkeiten auf unterschiedliche Bildeigenschaften. Diese Adaptionsfähigkeit ist jedoch nicht notwendig, wenn ein Algorithmus bereits stabil gegenüber den unterschiedlichsten Bildeigenschaften ist. Dieses trifft auf den *Block-Matching*-Algorithmus weitestgehend zu.

8. Diskussion

In dieser Arbeit wurde die Möglichkeit untersucht, Eigenbewegungen einer Kamera durch optischen Fluss zu bestimmen. Theoretische Vorbetrachtungen dazu wurden in Kapitel 2 angestellt. Kapitel 2.3.4 hat gezeigt, dass Translations-Bewegungskomponenten entlang der drei Kamera-Achsen und Rotation um die Achse in Blickrichtung voneinander unterschieden werden können. Die vorgestellte einfache Detektoranordnung ist jedoch nicht in der Lage Rotationen senkrecht zur Blickrichtungsachse von kombinierten Translationsbewegungen zu unterscheiden (2.2). Die ausgewählte Anordnung von Detektoren beruht auf Einzeldetektoren, die lediglich auf Erkennung des optischen Flusses zweier senkrecht zueinander stehender translatorischer Bewegungskomponenten beruht. Daher wurde im Weiteren eine Auswahl an Algorithmen zusammengestellt, die eine genaue Bestimmung einer dieser Flusskomponenten zulassen. Eine Vorarbeit dazu wurde in der Arbeit [BFB94] geleistet. Daraus wurden die besten Algorithmen für die Erkennung tanslationaler optischer Flüsse, erweitert um eine Auswahl an Algorithmen auf neuronaler Basis, für die Untersuchungen ausgewählt.

Im Weiteren wurden diese Verfahren auf ihre Fähigkeiten, den optischen Fluss zu bestimmen, getestet. Bewertungskriterien sind Qualität der Abbildung des tatsächlich auftretenden optischen Flusses einer realen Filmszene, wie sie bei Eigenbewegung entsteht, die Toleranz gegenüber Qualitätsänderungen des vorliegenen Bildmaterials (5.1.2) und Unanfälligkeit bei Störbewegungen der Kamera (5.2). Die Auswirkungen verschiedener Verfahren zur Vorverarbeitung des Bildmaterials wurden ebenfalls untersucht (3).

Kapitel 5 schildert detailliert die Ergebnisse der Experimente jedes einzelnen Algorithmus. Ein Vergleich und eine Bewertung der Algorithmen anhand dieser Ergebnisse wurde in Bezug auf einige Kriterien in 7 angestellt. In Abhängigkeit des Einsatzzwecks und der gestellten Anforderungen sind unterschiedliche Verfahren zu präferieren. Eine wesentliche Erkenntnis ist jedoch, dass die Auswahl geeigneter Schritte zur Vorverarbeitung des Bildmaterials großen Einfluss auf die Qualität der Ausgabe hat. So ist die vertikale Durchschnittsbildung, im Hinblick auf Toleranz ge-

genüber vertikalem optischen Fluss (5.4.1), nahezu unerlässlich bei Verfahren, die auf einer Bildzeile arbeiten. Ebenso verbessert eine Weichzeichnung die Ausgabe gradientenbasierter Algorithmen deutlich. Die Anwendung von Schwellwertverfahren steigert die Toleranz gegenüber Helligkeits- und Kontrastschwankungen. Vergleicht man die Leistungsfähigkeit der Algorithmen in Bezug auf die Bestimmung des optischen Flusses, sind die nicht neuronalen Verfahren den biologisch inspirierten überlegen. Das heißt, sie bilden den zu bestimmenden optischen Fluss in seiner Charakteristik besser ab.

Aufgrund der Detektionsgenauigkeit und Berechnungsgeschwindigkeit wurde das *Zeilen-First-Order*-Verfahren für die Mikrocontrollerimplementierung und den optischen Zeilensensor TSL3301 ausgewählt. Die Experimente dieser Implementierung können die Ergebnisse der Simulation bestätigen. Es zeigt sich auch, dass eine hohe Abtastrate von 2,5 kHz die Anfälligkeit eines Zeilen-Algorithmus bezüglich störendem optischen Fluss quer zur Detektionsrichtung kompensieren kann. Aufgrund der hohen Abtastrate stellen Kunstlichtquellen mit Leuchtschwankungen, die auf die 50 Hz Netzfrequenz zurückgehen, ein Problem dar. Diese Einflüsse lassen sich durch eine geeignete Filterung minimieren.

8.1. Zukünftige Arbeit

Die im Rahmen der vorliegenden Arbeit gewonnenen Erkenntnisse lassen weiterführende Tätigkeiten zu. Dazu gehört die Erweiterung der untersuchten Algorithmen. Insbesondere lässt sich das *Block-Matching*-Verfahren in der Form ändern, dass es den optischen Fluss nicht nur in ganzzahligen Größen, sondern auch in Bruchteilen davon, ausgeben kann. Damit wäre die gute Toleranz gegenüber Qualitätsschwankungen des Bildmaterials mit der besseren Einkoppelbarkeit in analoge Rechenstrukturen kombiniert. Grundsätzlich könnten die Erkenntnisse der Untersuchungen besser aufgeteilt werden in Effekte, die auf Vorverarbeitungsschritte zurückzuführen sind, und jene, die auf den eingesetzten Algorithmus zur Bestimmung des optischen Flusses basieren.

Die Mikrocontrollerimplementierung des *Zeilen-First-Order*-Algorithmus hat gezeigt, dass eine Filterung der Ausgabe unbedingt nötig ist. Dafür wäre die Analyse und der Vergleich verschiedener Filterstrategien nötig. Mögliche Auswirkungen der vorgestellten Vorverarbeitungsschritte auf die Detektion des optischen Flusses in der Mikrocontrollerimplementierung sollte ebenfalls untersucht werden. Auch könnte weitere Hardware, wie zum Beispiel ein FPGA, als Basis einer Implementierung dienen.

Zur Verbesserung der Erkennung von Eigenbewegungen, insbesondere Rotation um alle Kameraachsen, kann eine verbesserte Detektoranordnung entwickelt werden. Die Bestimmung des optischen Flusses auf dem genutzten Zeilensensor ließe sich in Teilbereiche der Zeile partitionieren, was eine Bestimmung von Translationen entlang der Blickrichtung des Sensors zuließe. Die Ergebnisse der Simulation könnten aufschlussreicher sein, wenn spezielle Eigenschaften der Kamera, wie Rauschverhalten und Verzeichnung, mit einfließen.

A. Diagramme

A.1. Zeilen-Block-Matching

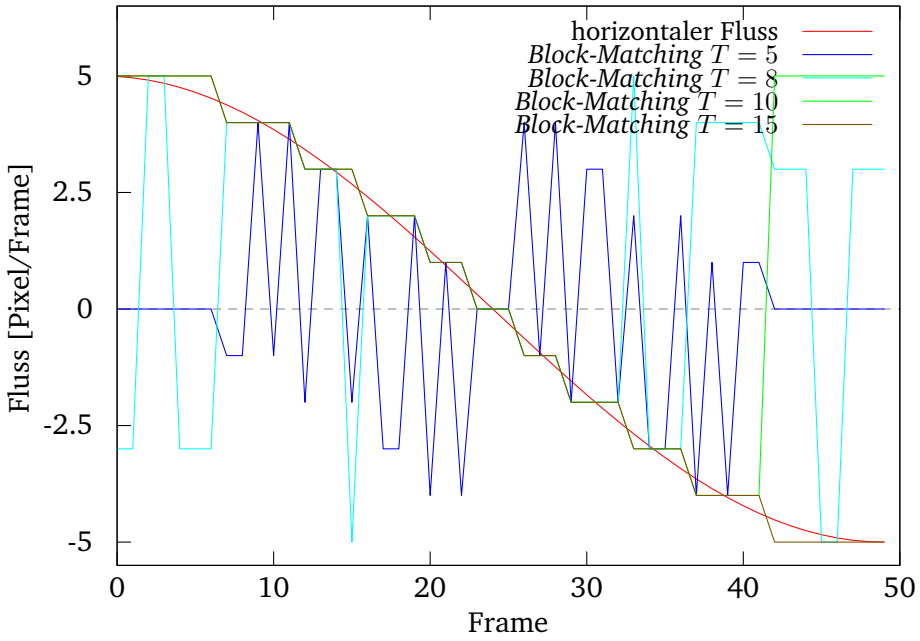

Abbildung A.1.: *Optischer Fluss des* Block-Matching.*-Algorithmus mit synthetischem Bildmaterial und einer Suchumgebung mit $K = 11$ für unterschiedliche Perioden T des Sinusoiden*

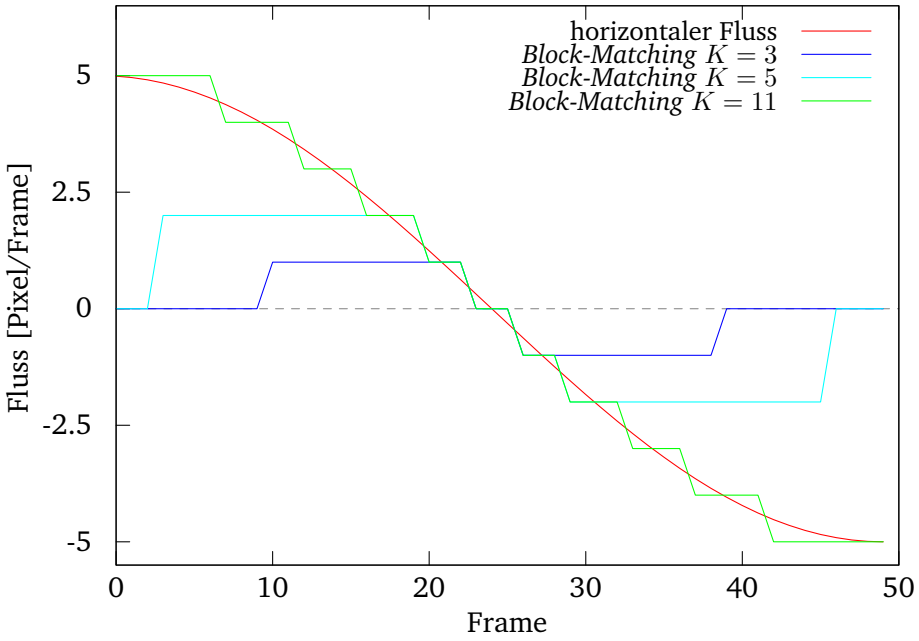

Abbildung A.2.: *Optischer Fluss des* Block-Matching-*Algorithmus mit synthetischem Bildmaterial für unterschiedliche Suchumgebungen K*

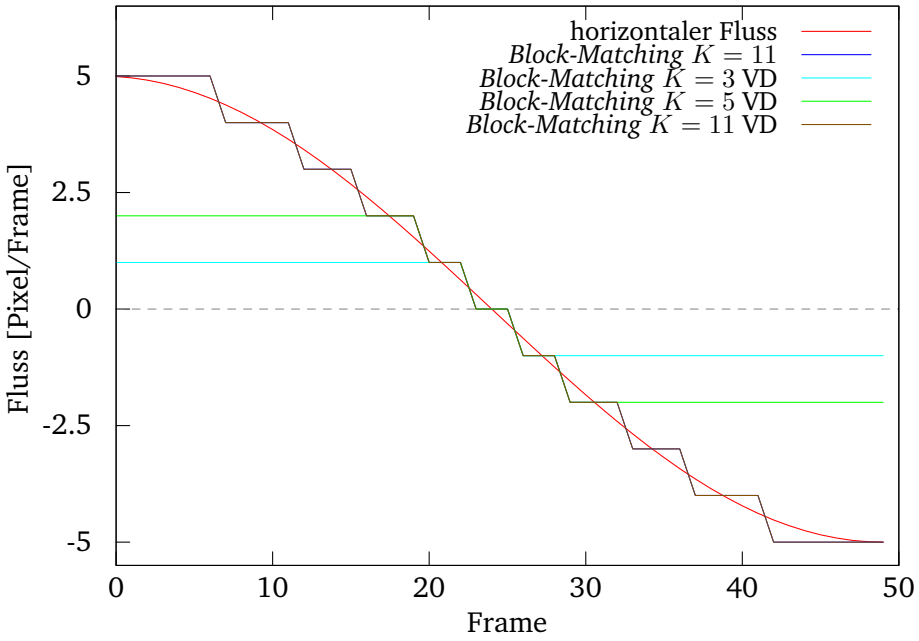

Abbildung A.3.: *Optischer Fluss des* Block-Matching-*Algorithmus mit realem Bildmaterial unterschiedlichen Suchumgebungen und Vorverarbeitungsschritten (VD meint vertikale Durchschnittsbildung)*

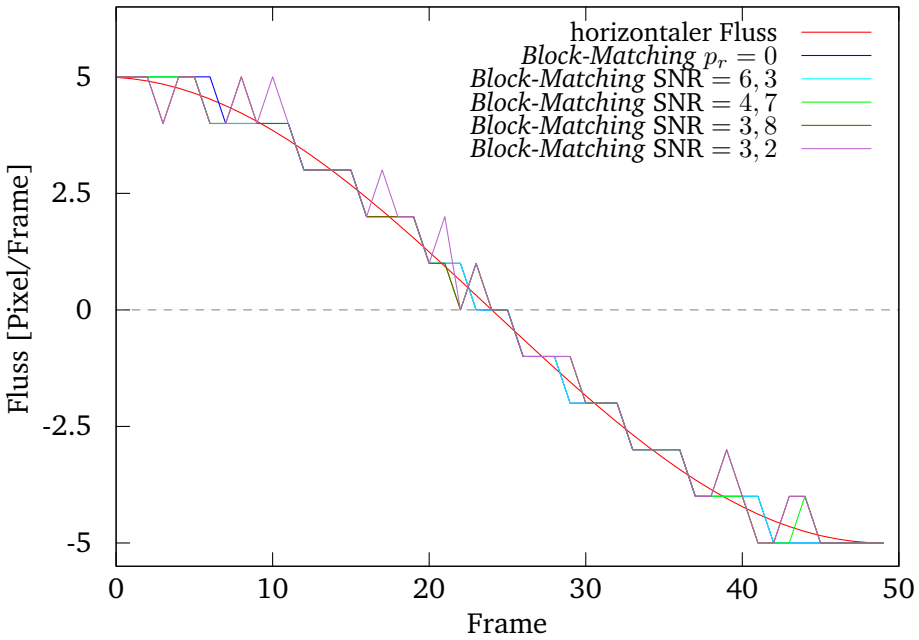

Abbildung A.4.: *Einfluss von Rauschen auf die Ausgabe des* Block-Matching-*Algorithmus mit realem Bildmaterial, einer Suchumgebung von* $K = 11$ *und vertikaler Durchschnittsbildung als Vorverarbeitung*

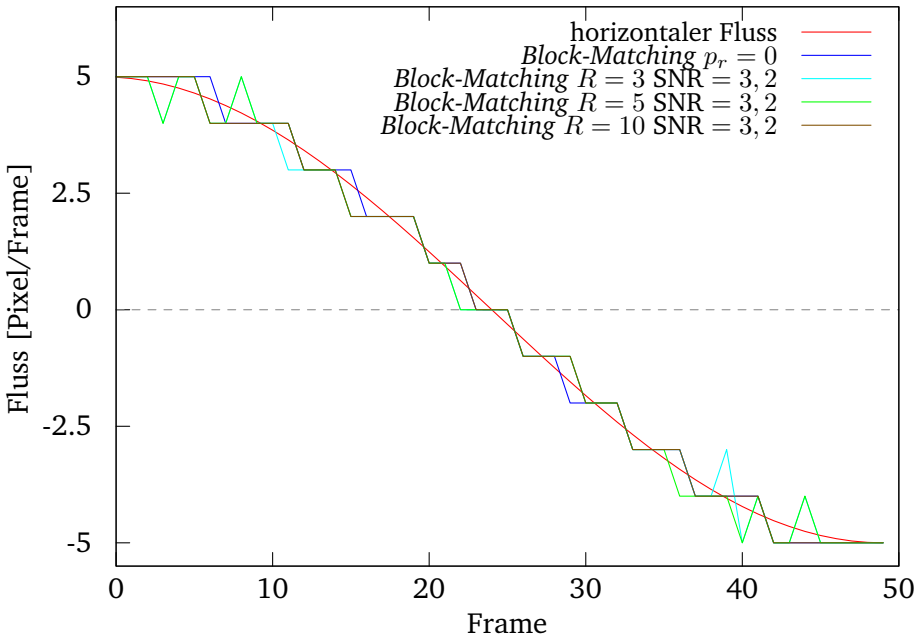

Abbildung A.5.: *Einfluss von Rauschen auf die Ausgabe des* Block-Matching-*Algorithmus mit realem Bildmaterial, einer Suchumgebung von* $K = 11$ *und vertikaler Durchschnittsbildung mit anschließender gleichverteilter Weichzeichnung für unterschiedliche Umgebungen* R *als Vorverarbeitung*

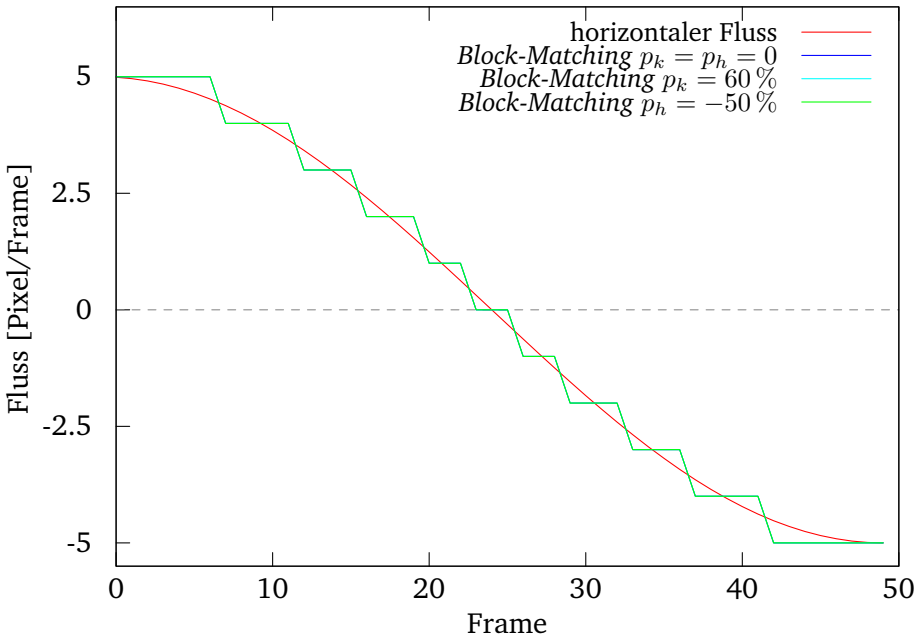

Abbildung A.6.: *Einfluss einer Kontrast- und Helligkeitsreduzierung auf die Ausgabe des* Block-Matching*-Algorithmus mit realem Bildmaterial, einer Suchumgebung von $K = 11$ und vertikaler Durchschnittsbildung als Vorverarbeitung*

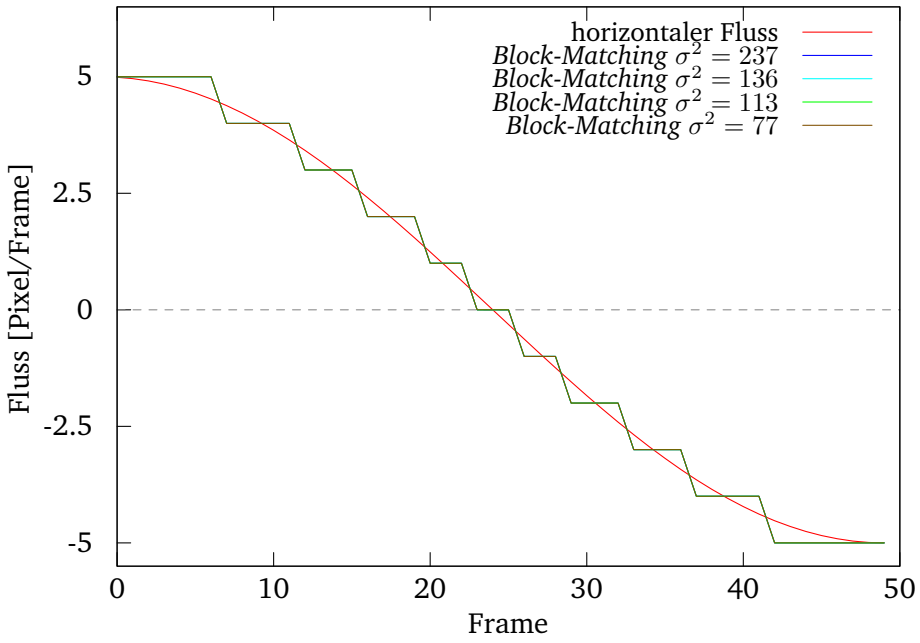

Abbildung A.7.: *Einfluss der Anzahl Kontrastreicher Kanten auf die Ausgabe des* Block-Matching-*Algorithmus mit realem Bildmaterial, einer Suchumgebung von* $K = 11$ *und vertikaler Durchschnittsbildung und Weichzeichnung mit* $R = 10$ *als Vorverarbeitung*

A.2. First-Order

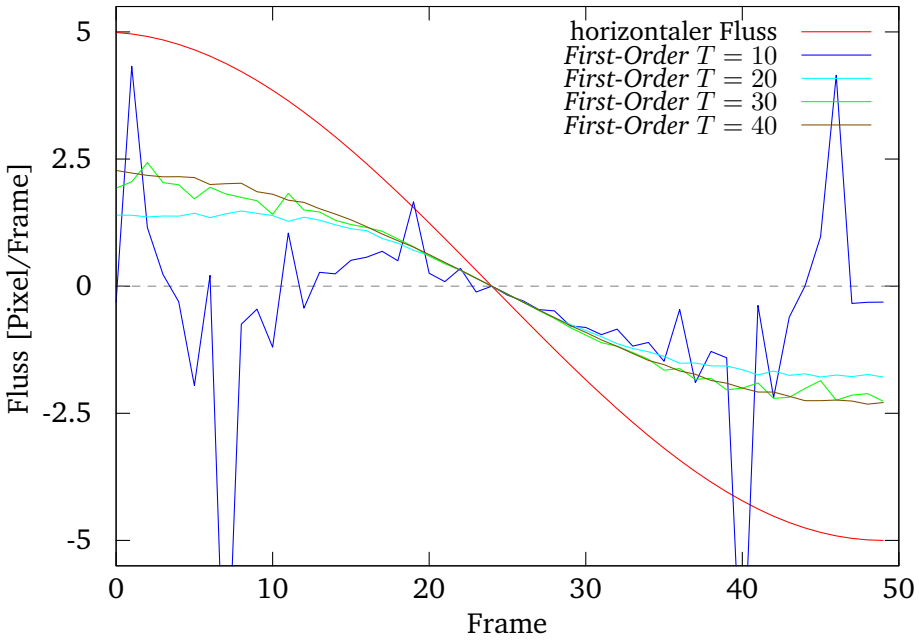

Abbildung A.8.: *Optischer Fluss des* First-Order*-Algorithmus mit synthetischem Bildmaterial und unterschiedlichen Perioden T des Sinusoiden*

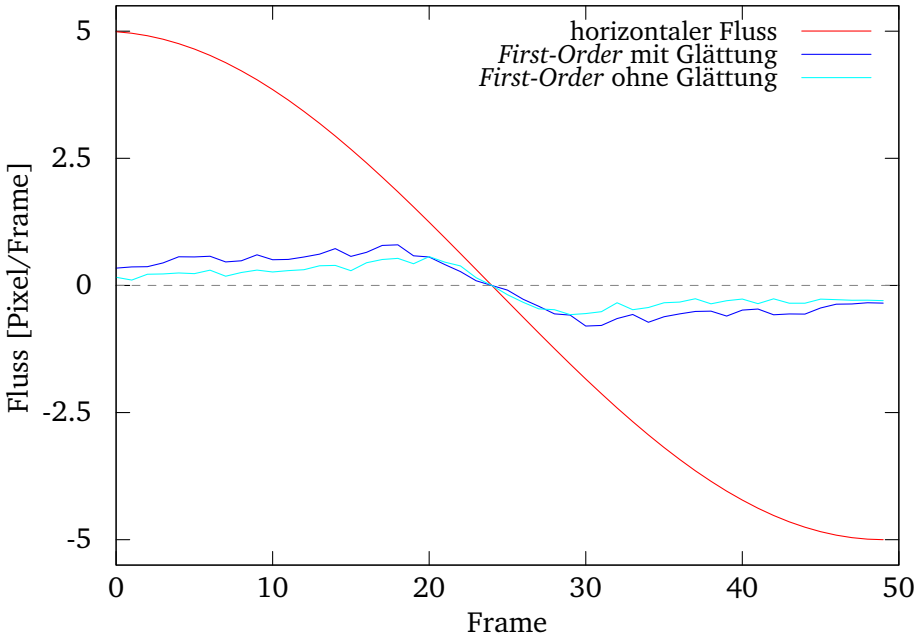

Abbildung A.9.: *Optischer Fluss des* First-Order-*Algorithmus mit natürlichem Bildmaterial und mit und ohne Glättung der Ableitung*

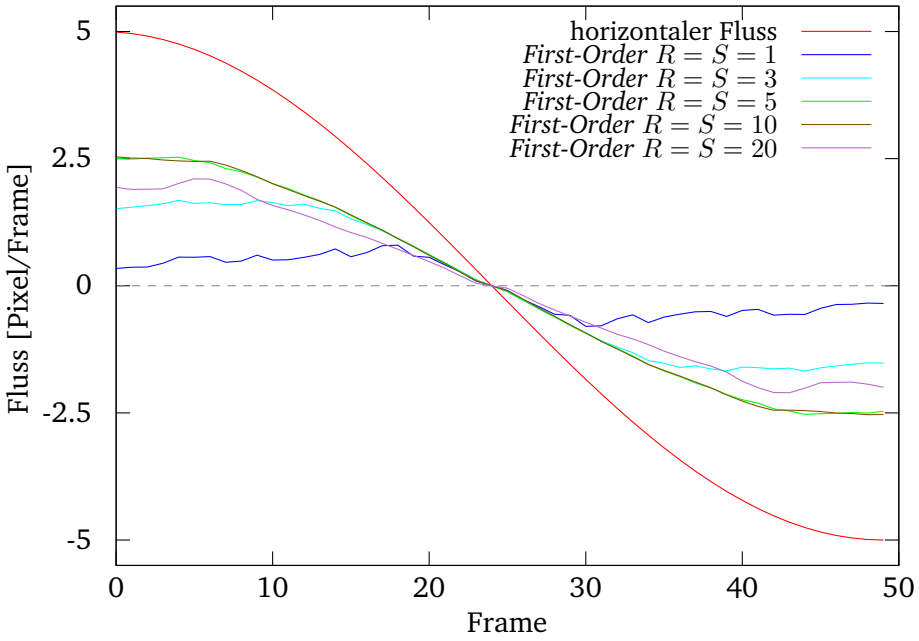

Abbildung A.10.: *Optischer Fluss des* First-Order-*Algorithmus mit natürlichem Bildmaterial, Glättung der Ableitung und unterschiedlichen Umgebungsgrößen R, S des Weichzeichners mit Gleichverteilung*

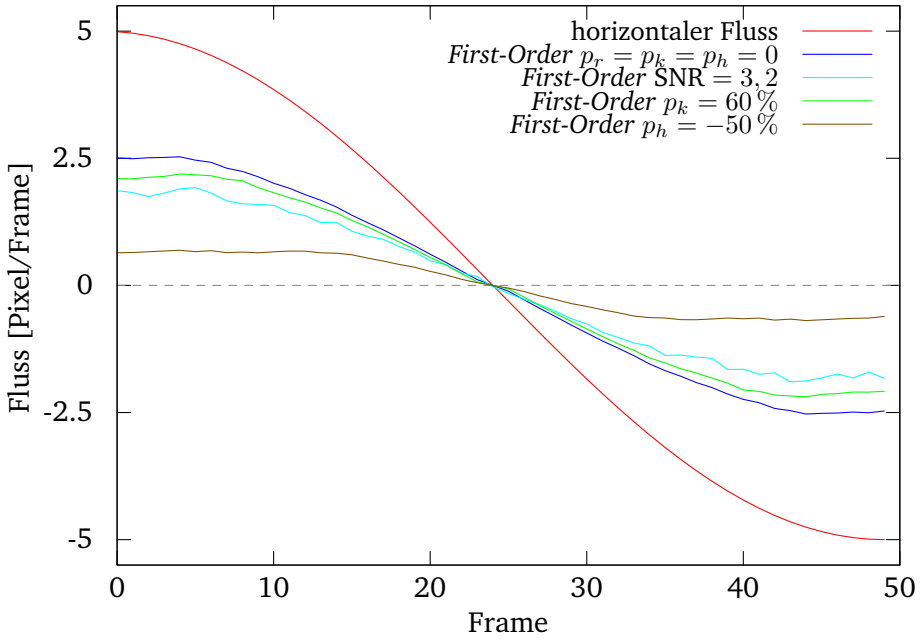

Abbildung A.11.: *Einfluss von Rauschen, Kontrast- und Helligkeitsreduzierung auf die Ausgabe des* First-Order-*Algorithmus mit Glättung der Differenzierung und Weichzeichnung mit* $R = S = 5$

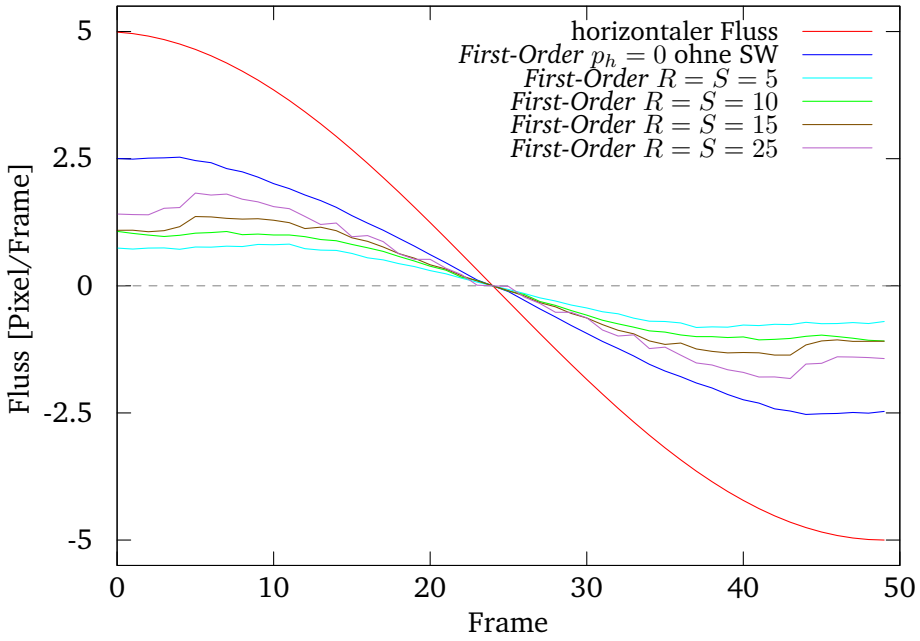

Abbildung A.12.: *Ausgabe des* First-Order-*Algorithmus mit Glättung der Differenzierung nach Schwellwertbildung (SW) und Weichzeichnung mit unterschiedlichen* R, S *bei einer Helligkeitsänderung von* $p_h = -50$

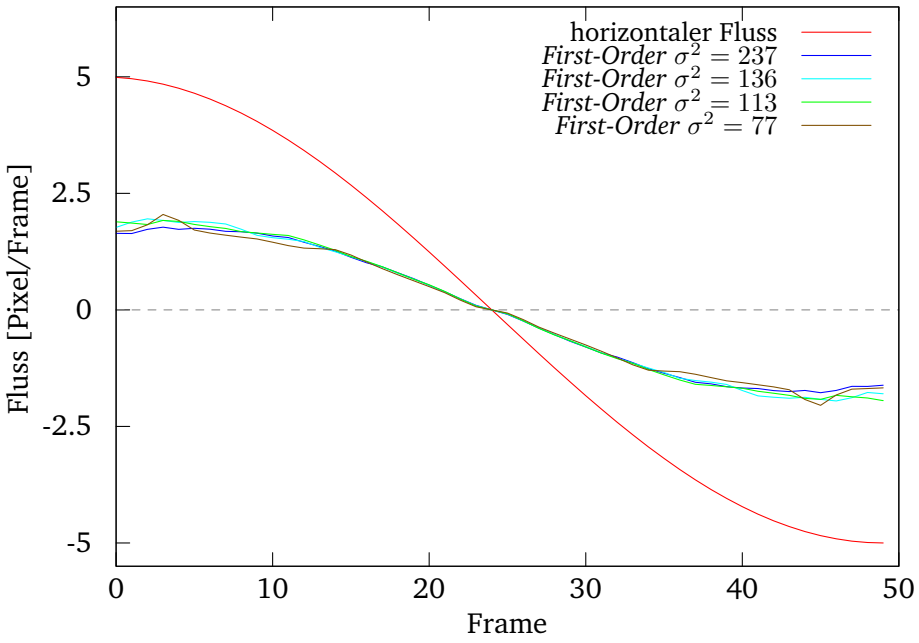

Abbildung A.13.: *Einfluss der Anzahl Kontrastreicher Kanten auf die Ausgabe des* First-Order-*Algorithmus mit Glättung der Differenzierung und Weichzeichnung mit* $R = S = 5$

A.3. Zeilen-First-Order

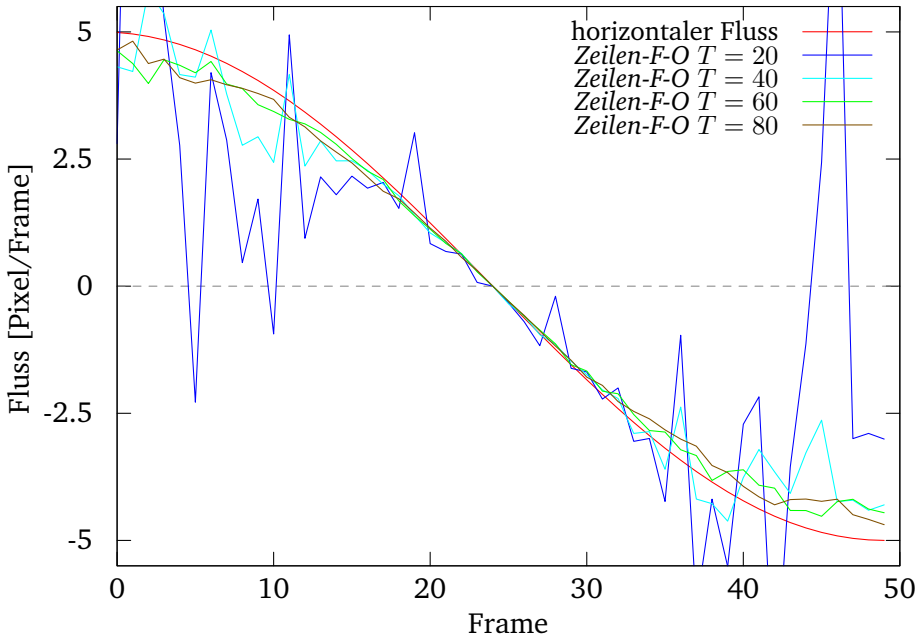

Abbildung A.14.: *Optischer Fluss des* Zeilen-First-Order-*Algorithmus mit synthetischem Bildmaterial und unterschiedlichen Perioden T des Sinusoiden*

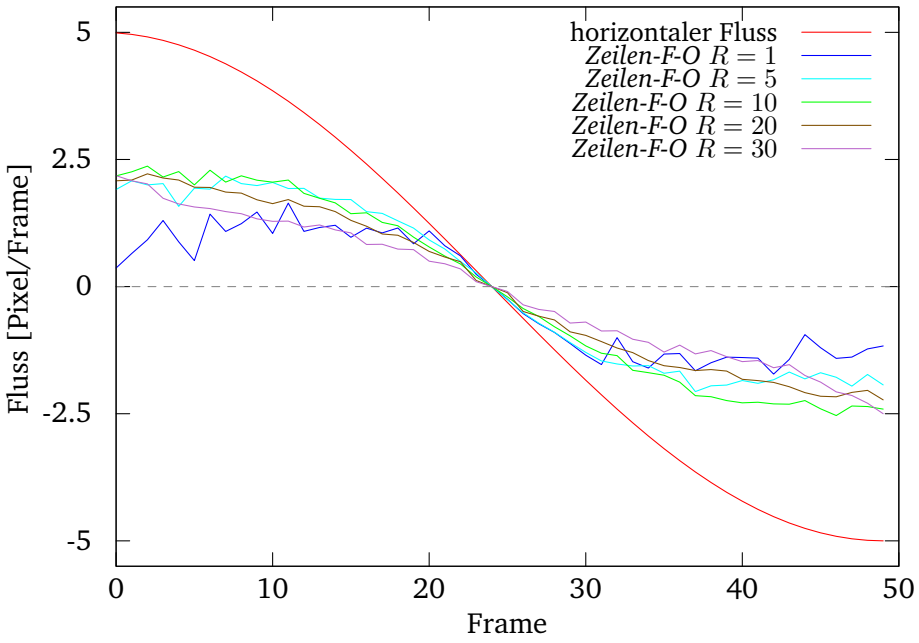

Abbildung A.15.: *Optischer Fluss des* Zeilen-First-Order-*Algorithmus mit natürlichem Bildmaterial, vertikaler Durchschnittsbildung und unterschiedlichen Umgebungsgrößen R des Weichzeichners mit Gleichverteilung*

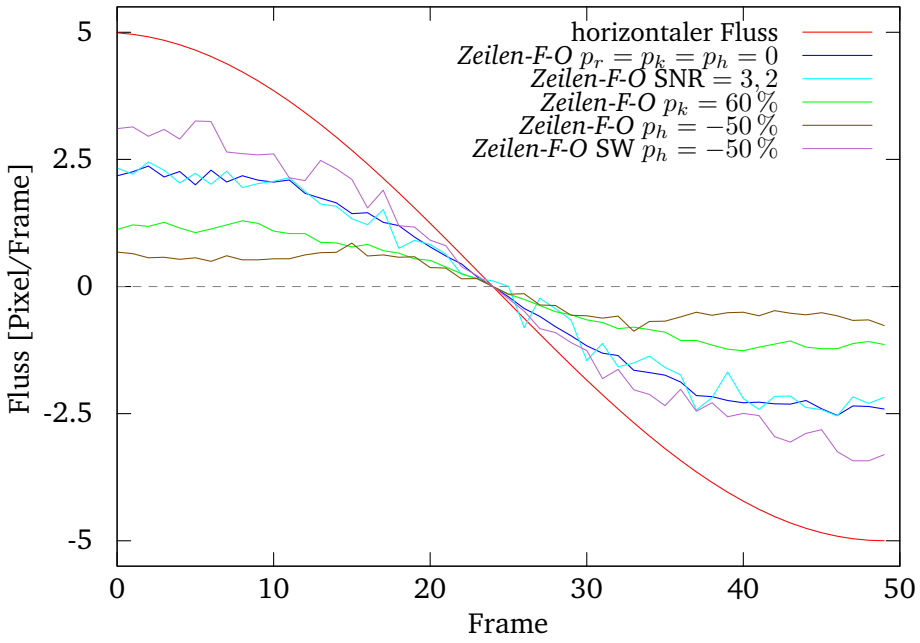

Abbildung A.16.: *Einfluss von Rauschen, Kontrast- und Helligkeitsreduzierung auf die Ausgabe des* Zeilen-First-Order-*Algorithmus mit vertikaler Durchschnittsbildung und Weichzeichnung mit $R = 10$ und, wenn mit SW gekennzeichnet, einer vorhergehenden Schwellwertbildung*

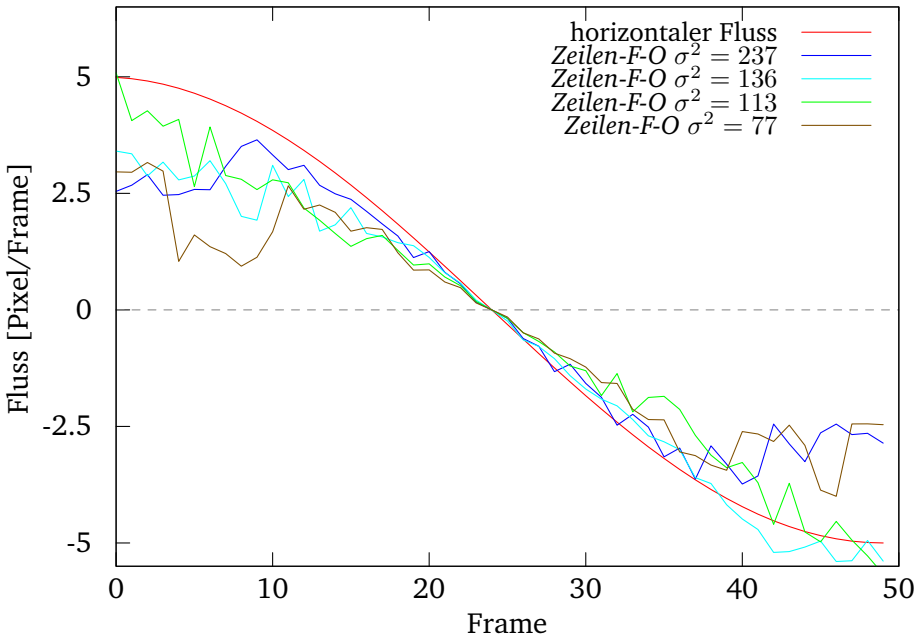

Abbildung A.17.: *Einfluss der Anzahl Kontrastreicher Kanten auf die Ausgabe des* First-Order-*Algorithmus mit Glättung der Differenzierung und Weichzeichnung mit* $R = S = 5$

A.4. Lucas-Kanade

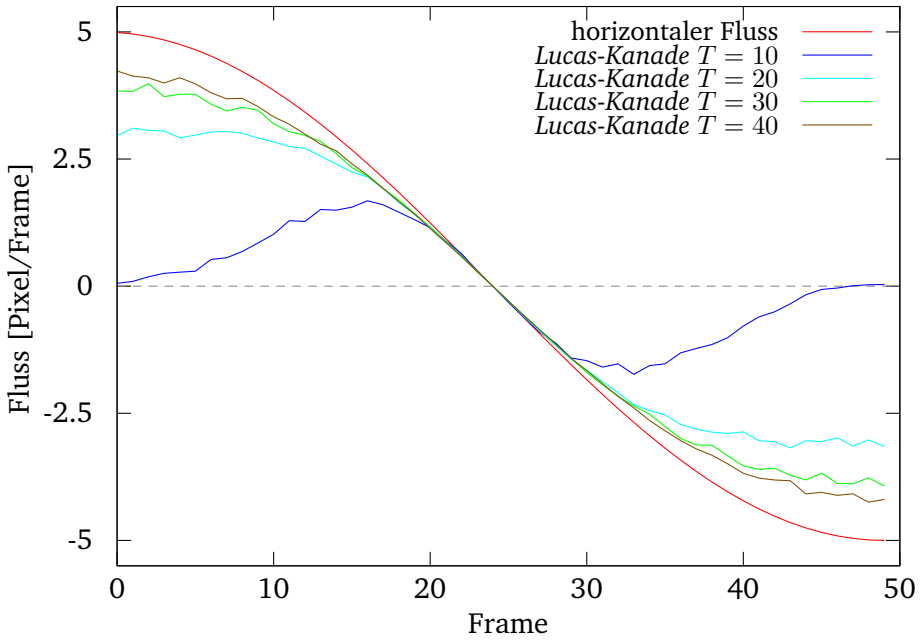

Abbildung A.18.: *Optischer Fluss des* Lucas-Kanade*-Algorithmus mit $n = 2 \times 2$ für synthetisches Bildmaterial und unterschiedlichen Perioden T des Sinusoiden*

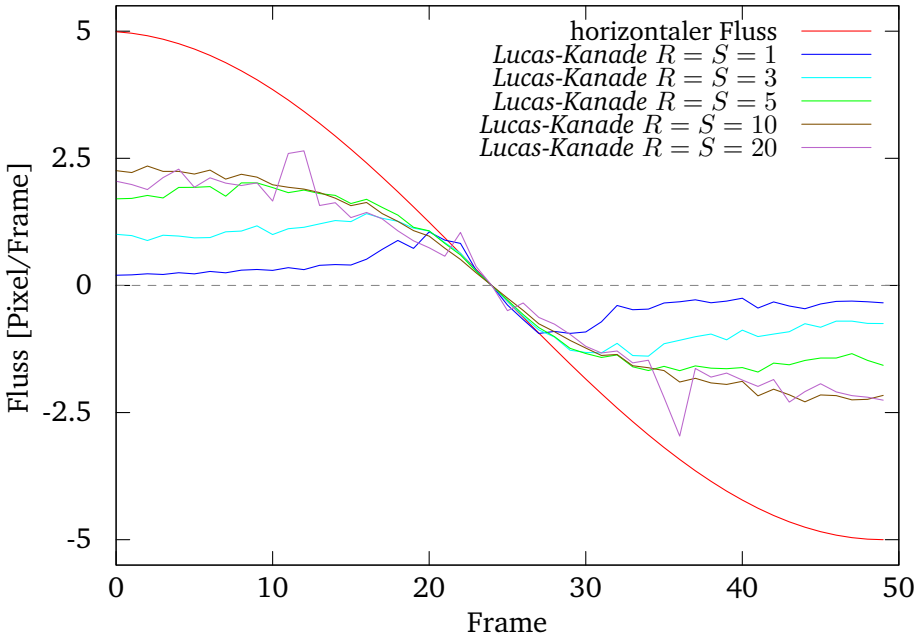

Abbildung A.19.: *Optischer Fluss des* Lucas-Kanade-*Algorithmus mit einer Umgebungsgröße von $n = 2 \times 2$ und realem Bildmaterial für unterschiedliche Umgebungsgrößen R, S des Weichzeichners mit Gleichverteilung*

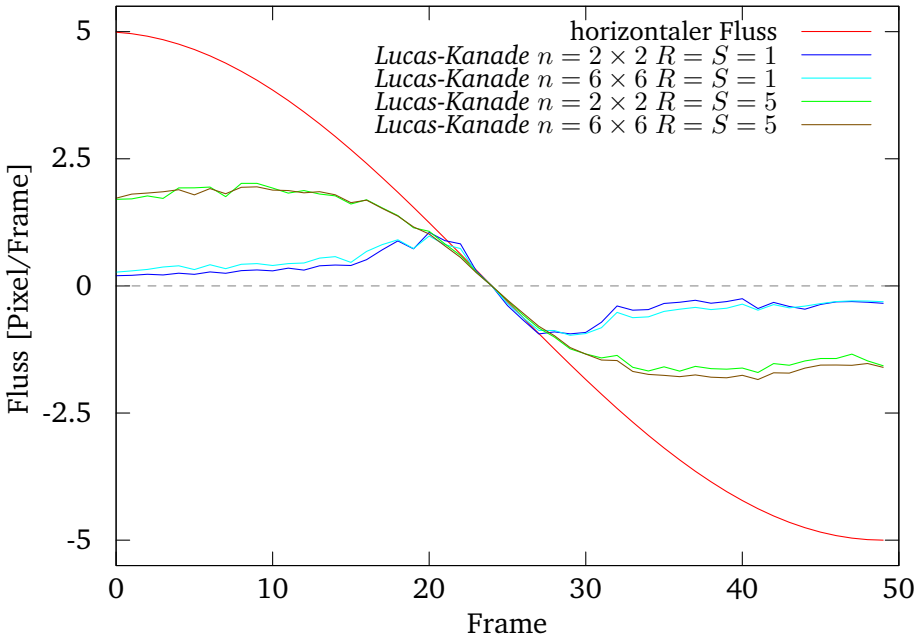

Abbildung A.20.: *Optischer Fluss des* Lucas-Kanade-*Algorithmus mit unterschiedlichen Umgebungsgrößen n, R, S und realem Bildmaterial*

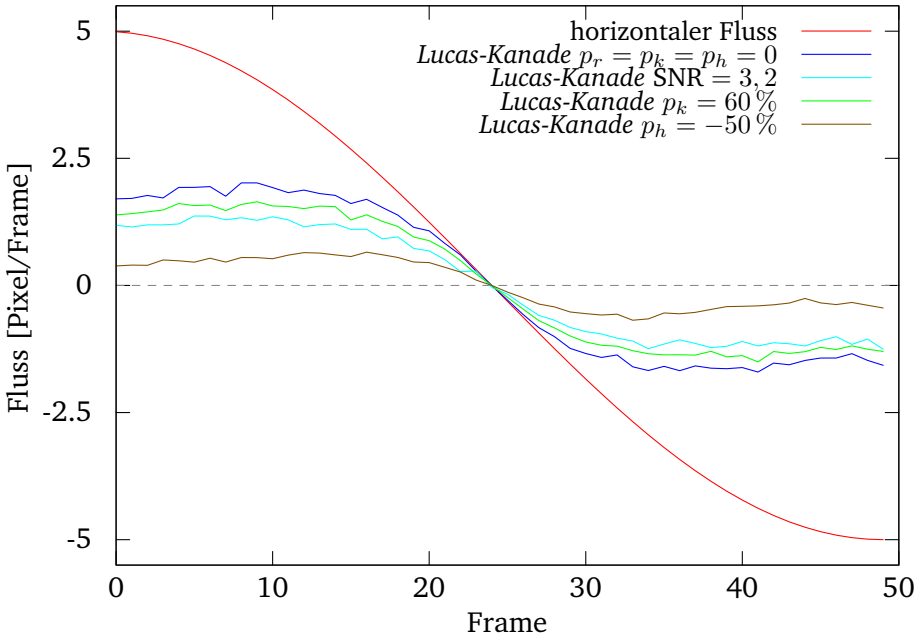

Abbildung A.21.: *Einfluss von Rauschen, Kontrast- und Helligkeitsreduzierung auf die Ausgabe des* Lucas-Kanade-*Algorithmus mit* $n = 2 \times 2$ *und* $R = S = 5$

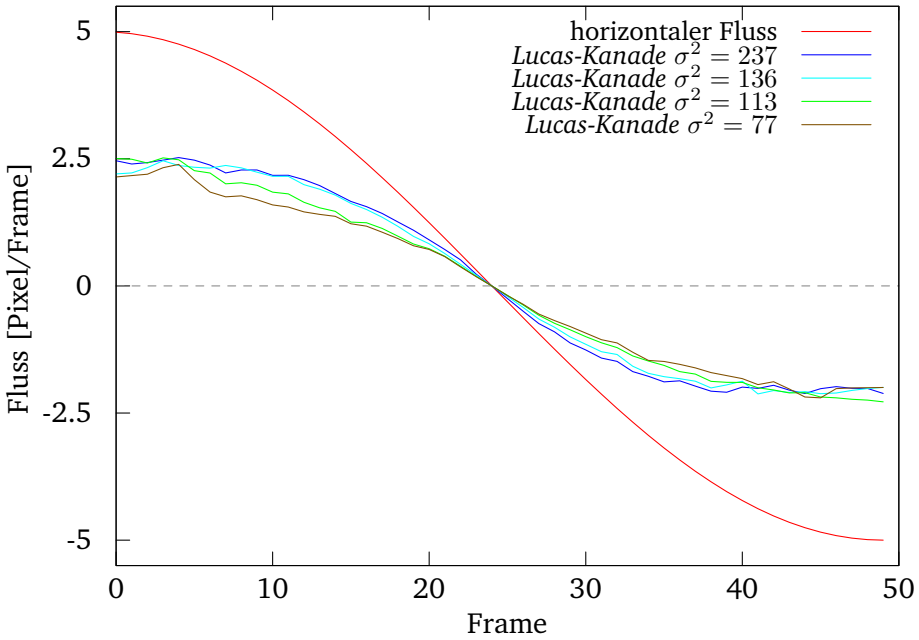

Abbildung A.22.: *Einfluss der Anzahl Kontrastreicher Kanten auf die Ausgabe des* Lucas-Kanade-*Algorithmus mit* $n = 2 \times 2$ *und* $R = S = 5$

A.5. Uras et al.

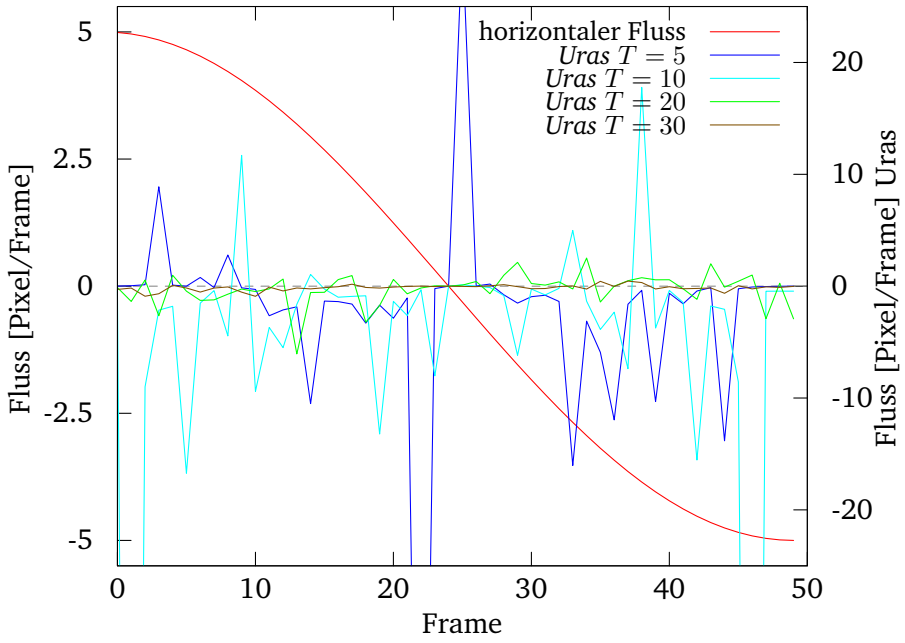

Abbildung A.23.: *Optischer Fluss des* Uras-et-al.*-Algorithmus mit synthetischem Bildmaterial und Selektion der Geschwindigkeitsvektoren für unterschiedliche Perioden T des Sinusoiden*

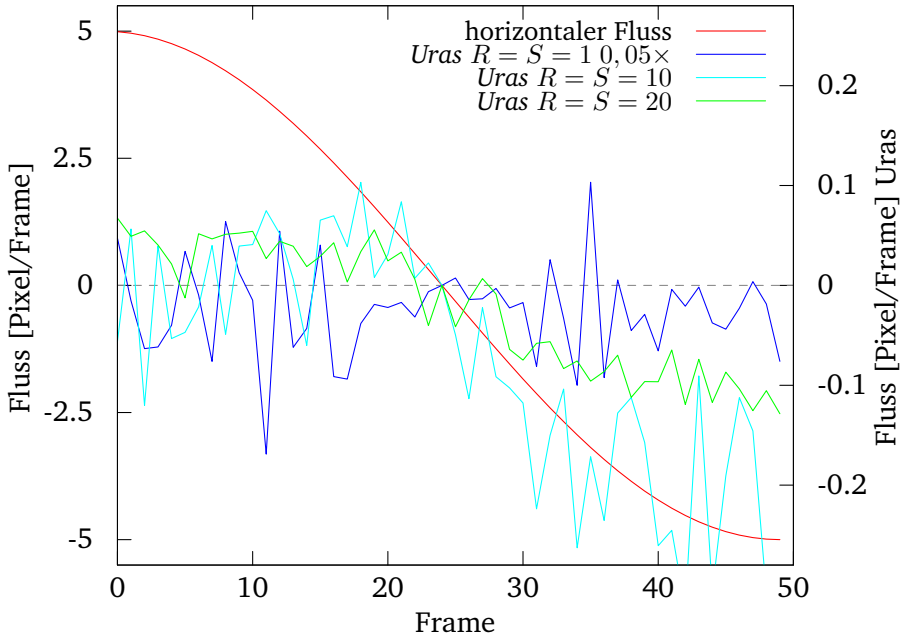

Abbildung A.24.: *Optischer Fluss des* Uras-et-al-*Algorithmus ohne Selektierung der Geschwindigkeitsvektoren mit realem Bildmaterial für unterschiedliche Umgebungsgrößen R, S des Weichzeichners mit Gleichverteilung*

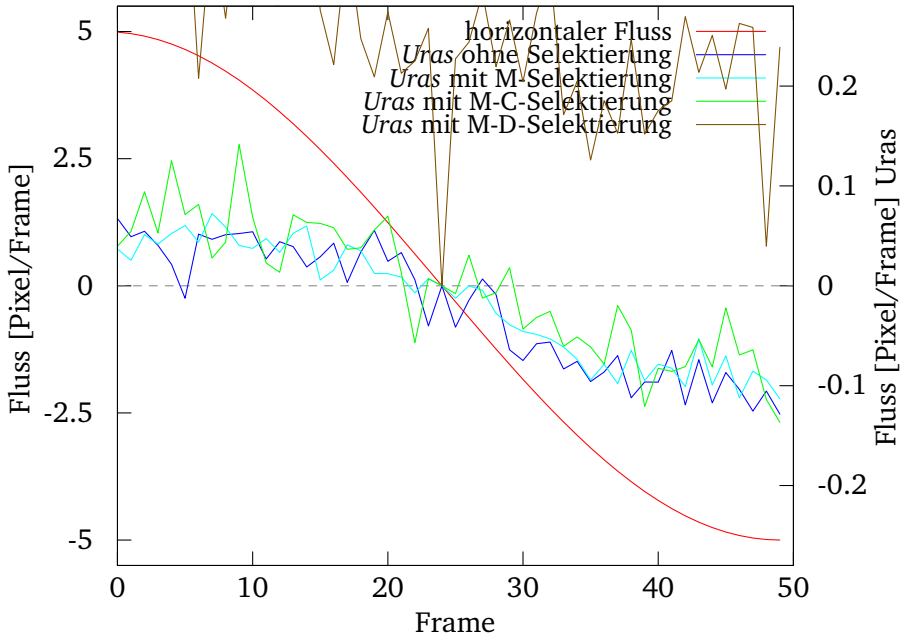

Abbildung A.25.: *Optischer Fluss des* Uras-et-al-*Algorithmus mit $R = S = 20$ des Weichzeichners mit Gleichverteilung und realem Bildmaterial für unterschiedliche Selektierungen der Geschwindigkeitsvektoren*

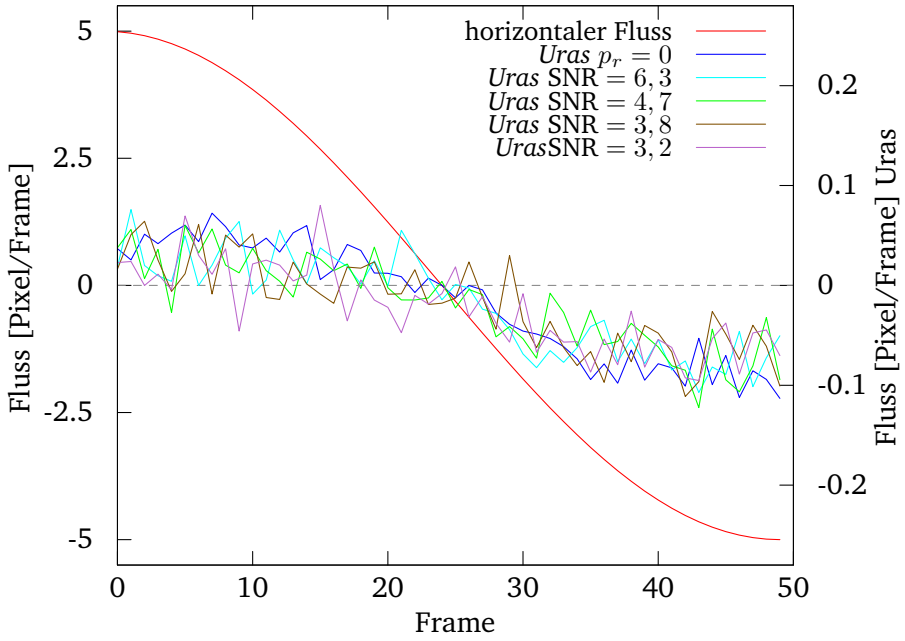

Abbildung A.26.: *Einfluss von Rauschen auf die Ausgabe des* Uras-et-al.*-Algorithmus mit $R = S = 20$ des Weichzeichners mit Gleichverteilung und Selektion der Geschwindigkeitsvektoren anhand* $\mathbf{M}$

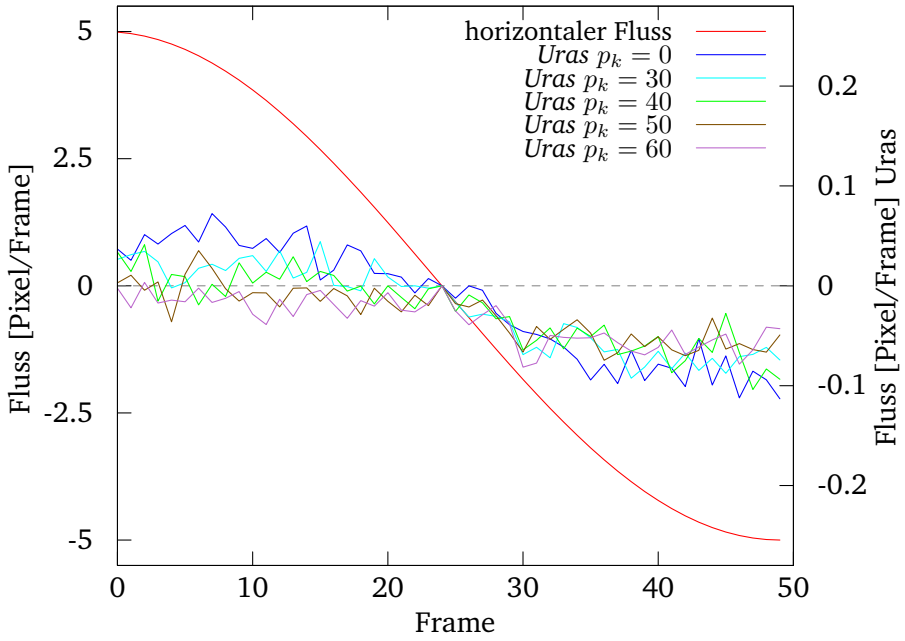

Abbildung A.27.: *Einfluss einer Kontrastreduzierung auf die Ausgabe des* Uras-et-al.*-Algorithmus mit $R = S = 20$ des Weichzeichners mit Gleichverteilung und Selektion der Geschwindigkeitsvektoren anhand* $\mathbf{M}$

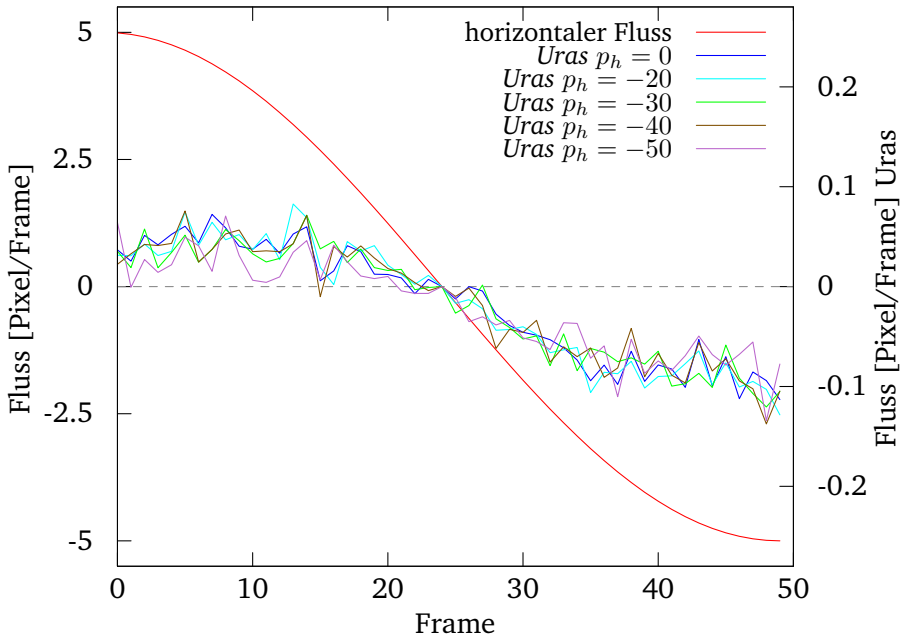

Abbildung A.28.: *Einfluss einer Helligkeitsreduzierung auf die Ausgabe des* Uras-et-al.*-Algorithmus mit $R = S = 20$ des Weichzeichners mit Gleichverteilung und Selektion der Geschwindigkeitsvektoren anhand* $\mathbf{M}$

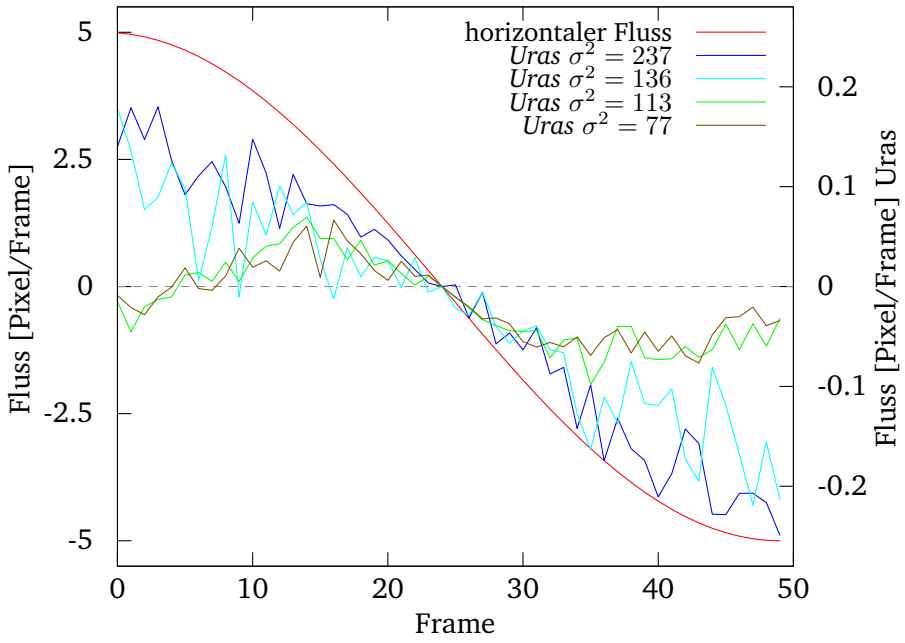

Abbildung A.29.: *Einfluss der Anzahl Kontrastreicher Kanten auf die Ausgabe des* Uras-et-al.*-Algorithmus mit $R = S = 20$ des Weichzeichners mit Gleichverteilung und Selektion der Geschwindigkeitsvektoren anhand* $\mathbf{M}$

A.6. Fleet und Jepson

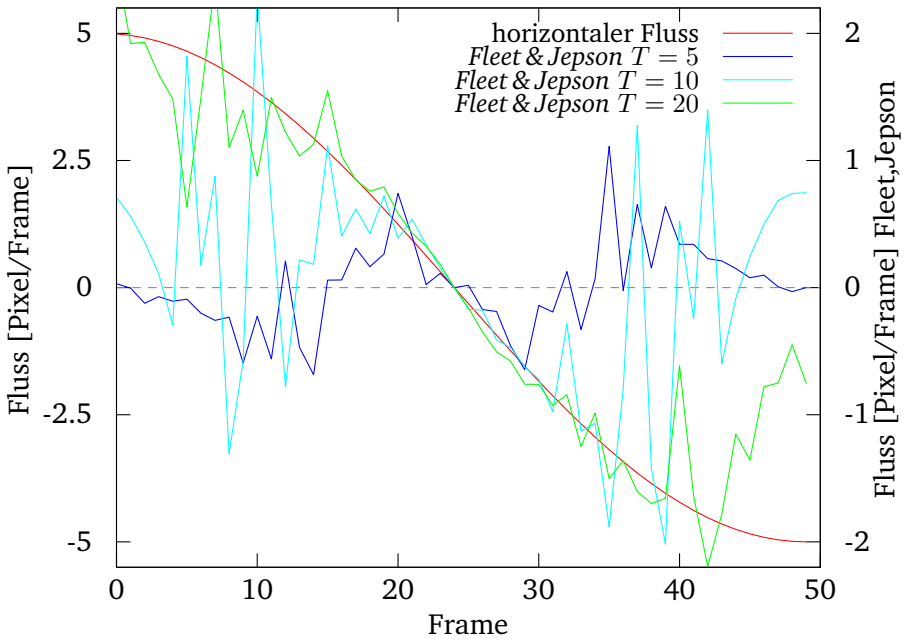

Abbildung A.30.: *Optischer Fluss des* Fleet & Jepson*-Algorithmus mit synthetischem Bildmaterial und den Parametern $T = 10$, $\theta = 0$ und $\sigma = 3$ für unterschiedliche Perioden T des Sinusoiden*

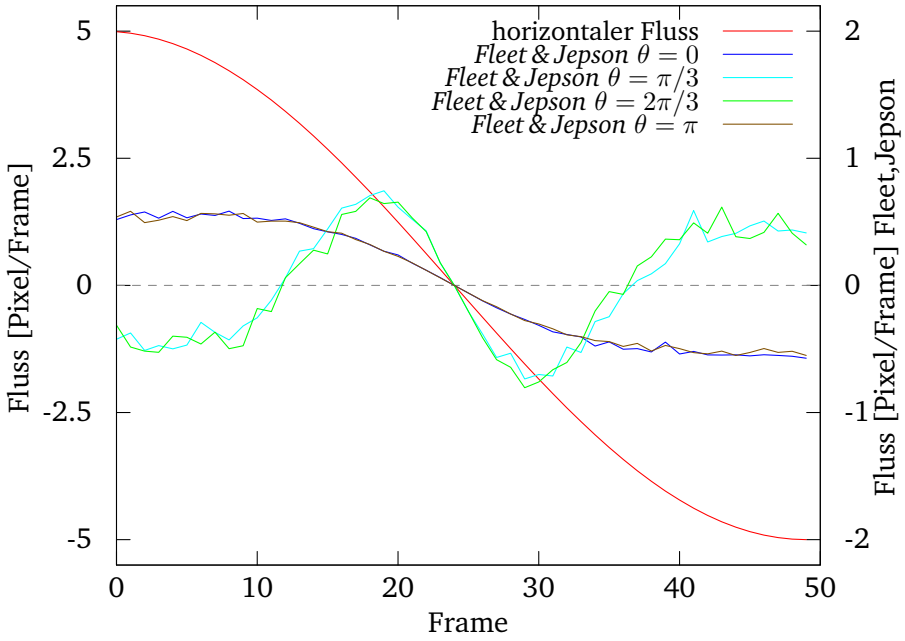

Abbildung A.31.: *Optischer Fluss des* Fleet & Jepson*-Algorithmus mit realem Bildmaterial, $T = 5$, $\sigma = 3$ und unterschiedlichen Ausrichtungen θ der Basisfunktion*

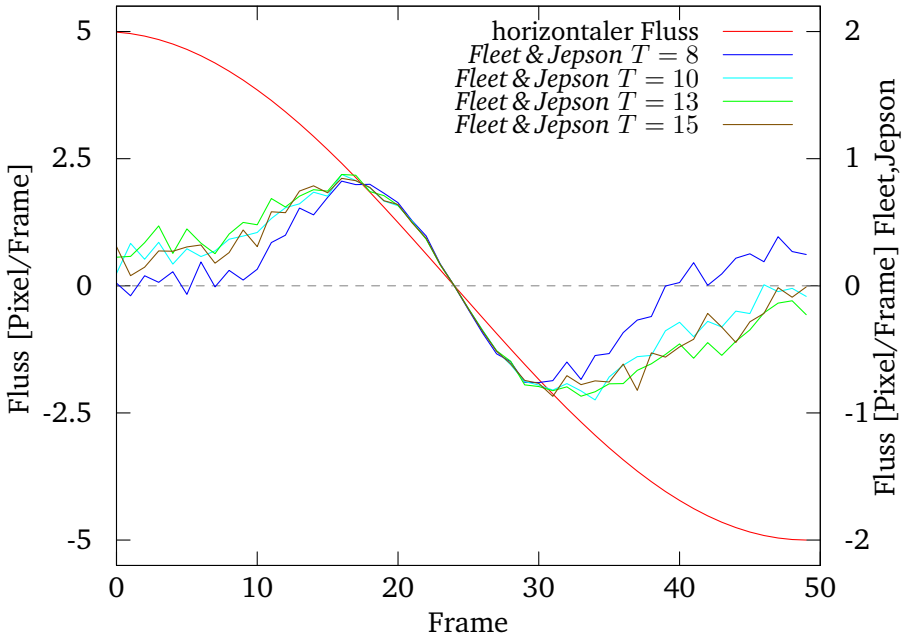

Abbildung A.32.: *Optischer Fluss des* Fleet & Jepson*-Algorithmus mit realem Bildmaterial, $\theta = 2\pi/3$, $\sigma = 3$ und unterschiedlichen Perioden T der Basisfunktion*

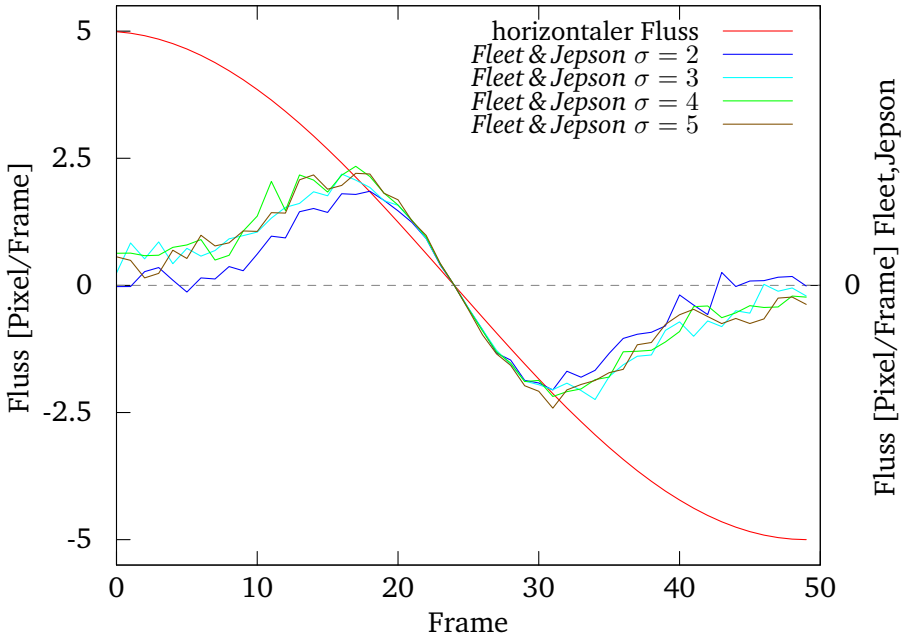

Abbildung A.33.: *Optischer Fluss des* Fleet & Jepson*-Algorithmus mit realem Bildmaterial, $\theta = 2\pi/3$, $T = 10$ und unterschiedlichen Breiten σ der Fensterfunktion*

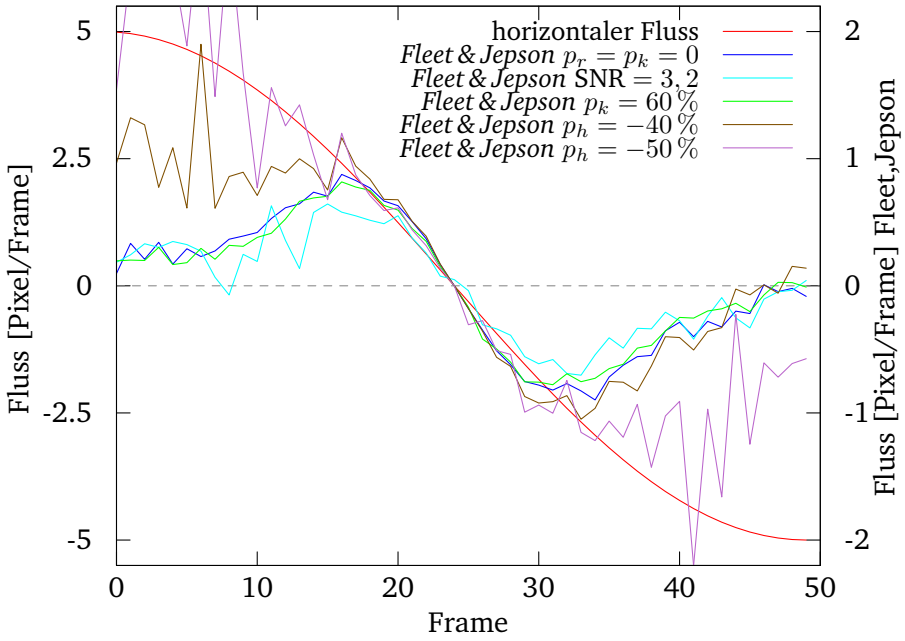

Abbildung A.34.: *Einfluss von Rauschen, Helligkeits- und Kontrastreduzierung auf die Ausgabe des* Fleet & Jepson*-Algorithmus mit $\theta = 2\pi/3$, $T = 10$ und $\sigma = 3$*

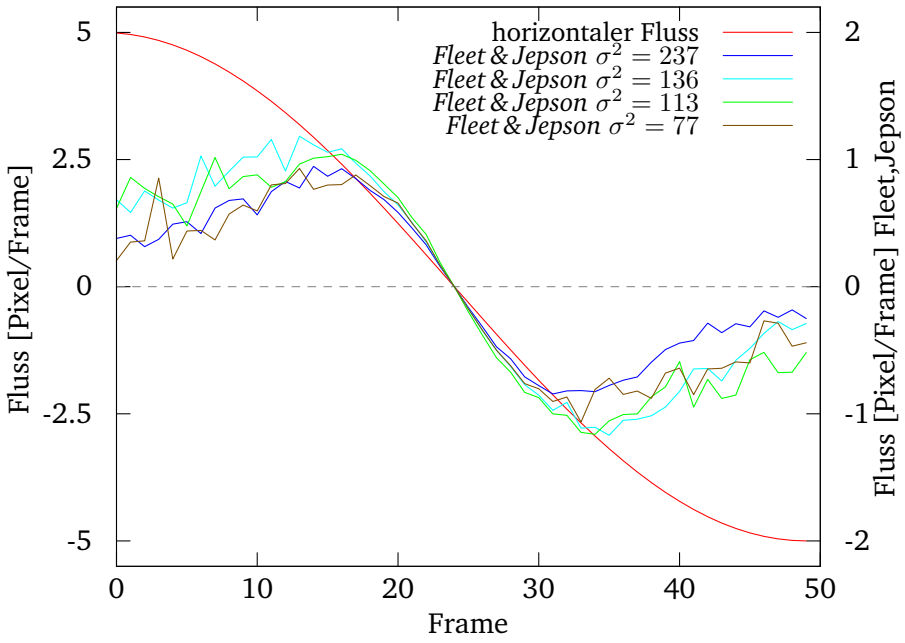

Abbildung A.35.: *Einfluss der Anzahl Kontrastreicher Kanten auf die Ausgabe des* Fleet & Jepson-*Algorithmus mit* $\theta = 2\pi/3$, $T = 10$ *und* $\sigma = 3$

A.7. Reichardt-Detektor

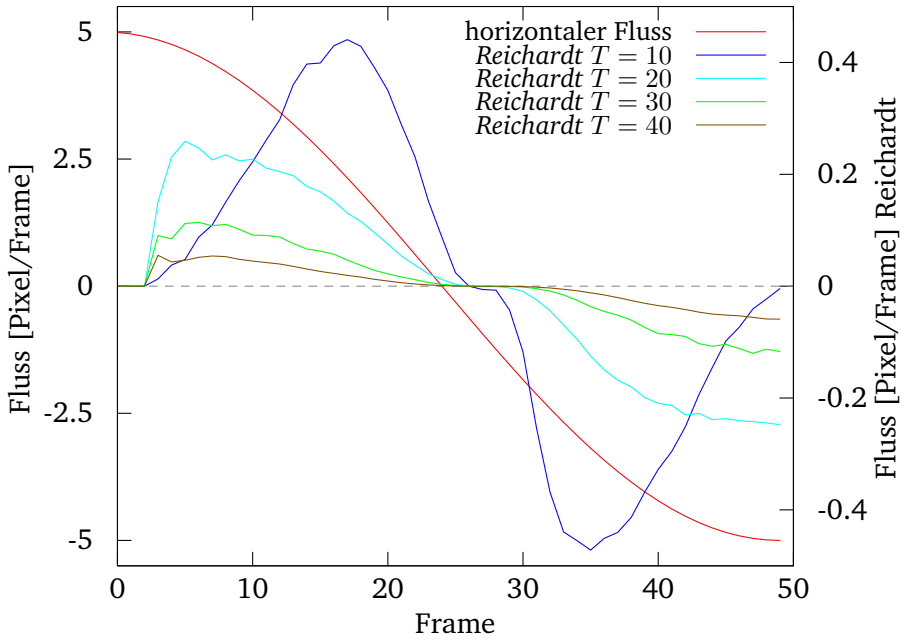

Abbildung A.36.: *Optischer Fluss des* Reichardt*-Algorithmus mit $\alpha = 0.5$ für synthetisches Bildmaterial und unterschiedliche Perioden T des Sinusoiden*

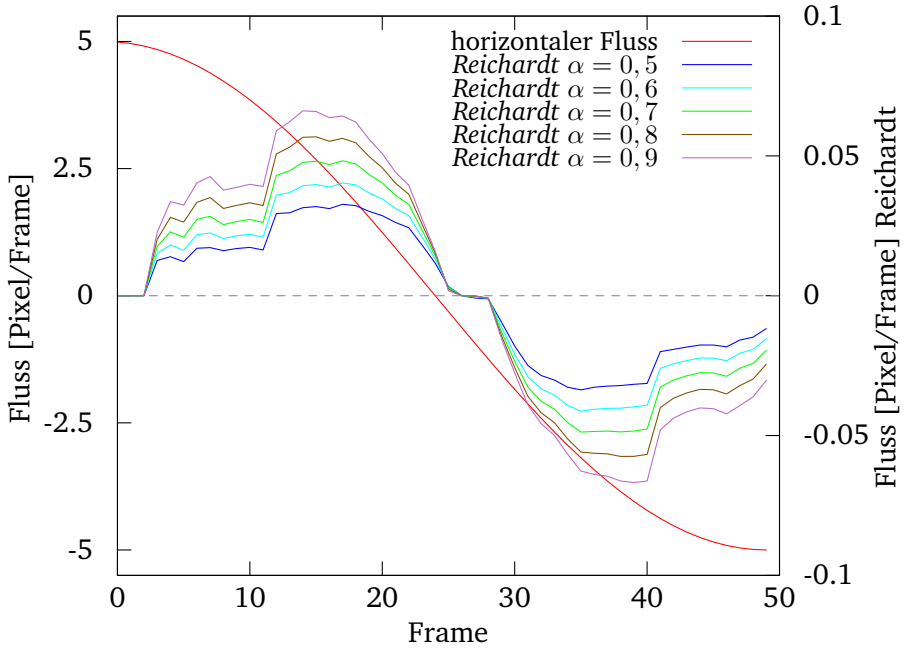

Abbildung A.37.: *Optischer Fluss des* Reichardt-*Algorithmus mit realen Bilddaten und unterschiedlichen* α

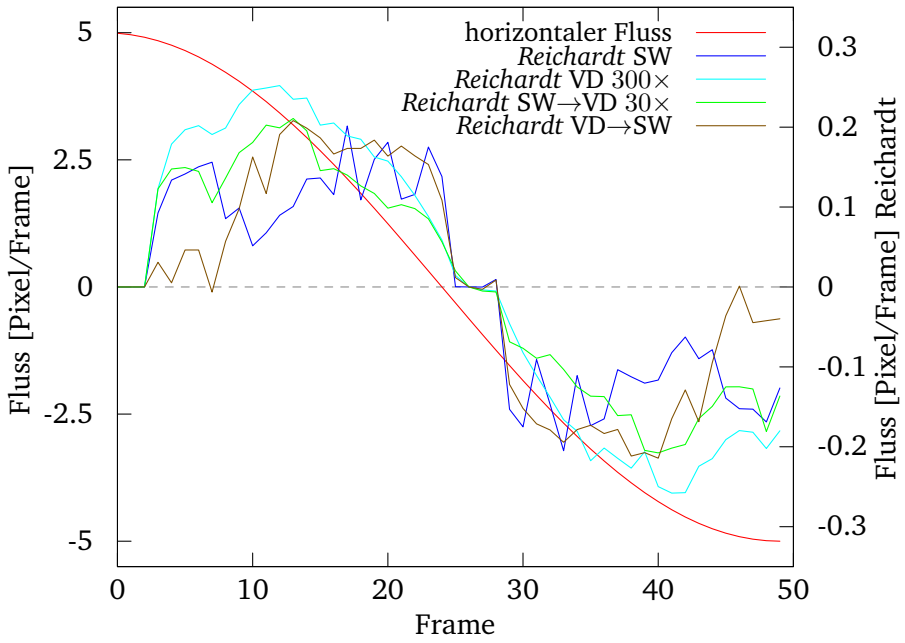

Abbildung A.38.: *Optischer Fluss des* Reichardt-*Algorithmus mit realen Bilddaten und unterschiedlichen Vorverarbeitungsschritten*

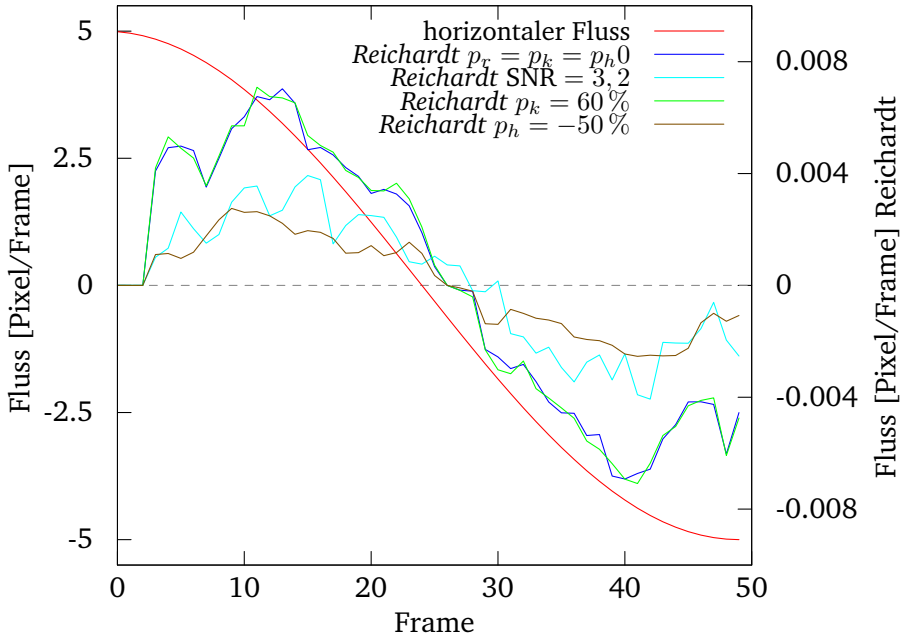

Abbildung A.39.: *Einfluss von Rauschen, Kontrast- und Helligkeitsreduzierung auf die Ausgabe des* Reichardt*-Algorithmus mit $\alpha = 0,5$ und Schwellwertbildung mit anschließender vertikaler Durchschnittsbildung*

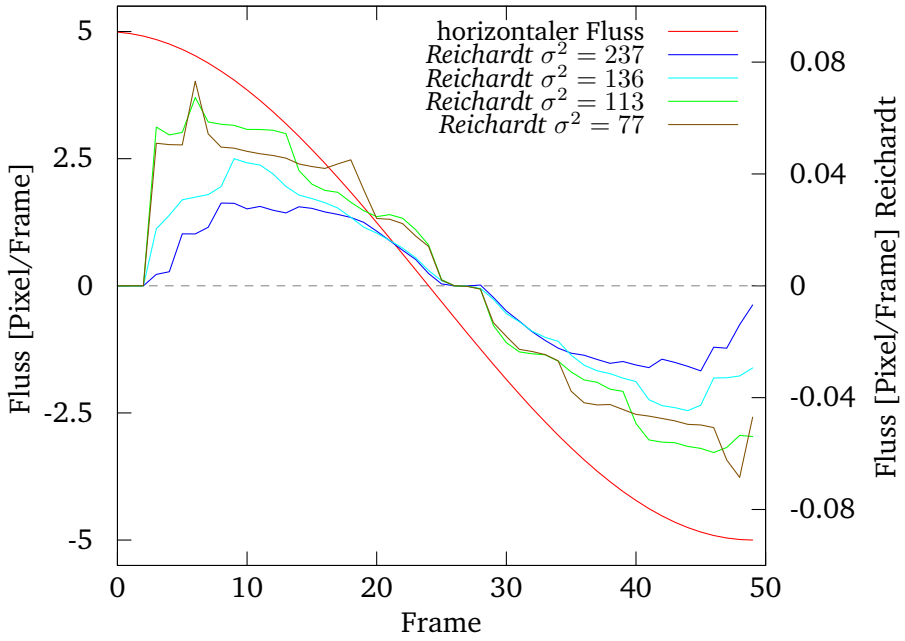

Abbildung A.40.: *Einfluss der Anzahl Kontrastreicher Kanten auf die Ausgabe des* Reichardt*-Algorithmus mit $\alpha = 0,5$ und Schwellwertbildung mit anschließender vertikaler Durchschnittsbildung*

A.8. Neuronenzeile

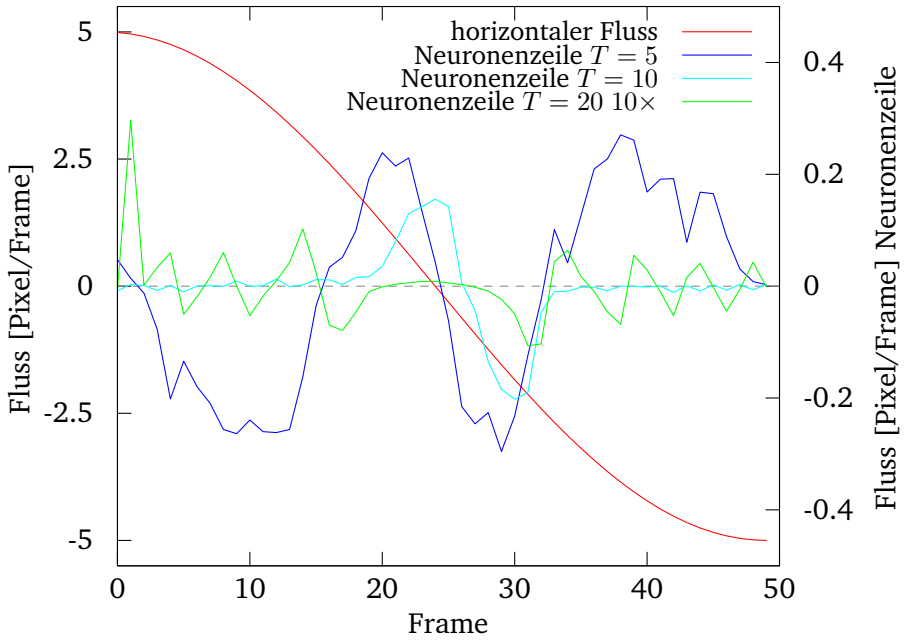

Abbildung A.41.: *Optischer Fluss der einlagigen Neuronenzeile mit Gewichten für komplexe künstliche Reize für synthetisches Bildmaterial und unterschiedliche Perioden T des Sinusoiden*

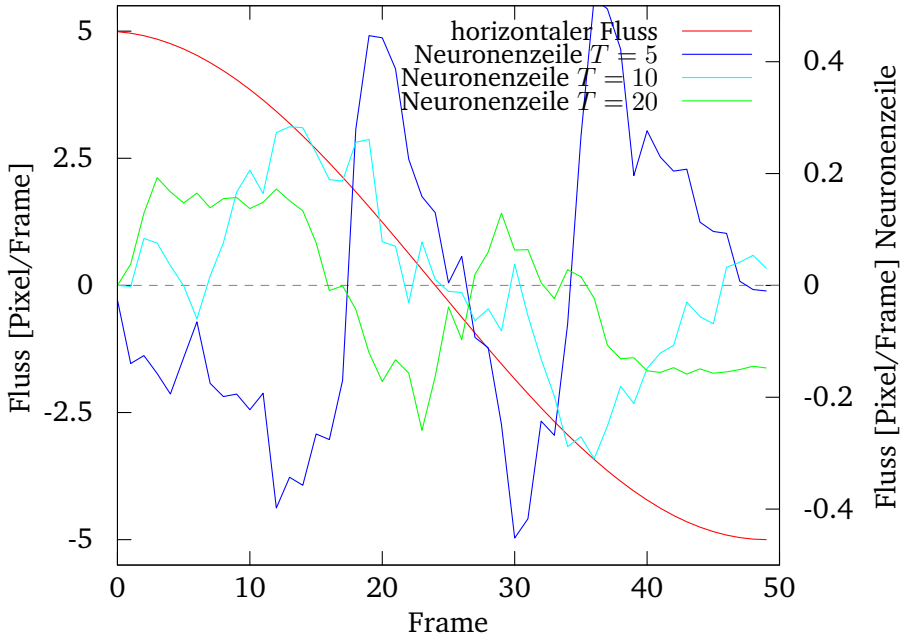

Abbildung A.42.: *Optischer Fluss der zweilagigen Neuronenzeile mit Gewichten für komplexe künstliche Reize für synthetisches Bildmaterial und unterschiedliche Perioden T des Sinusoiden*

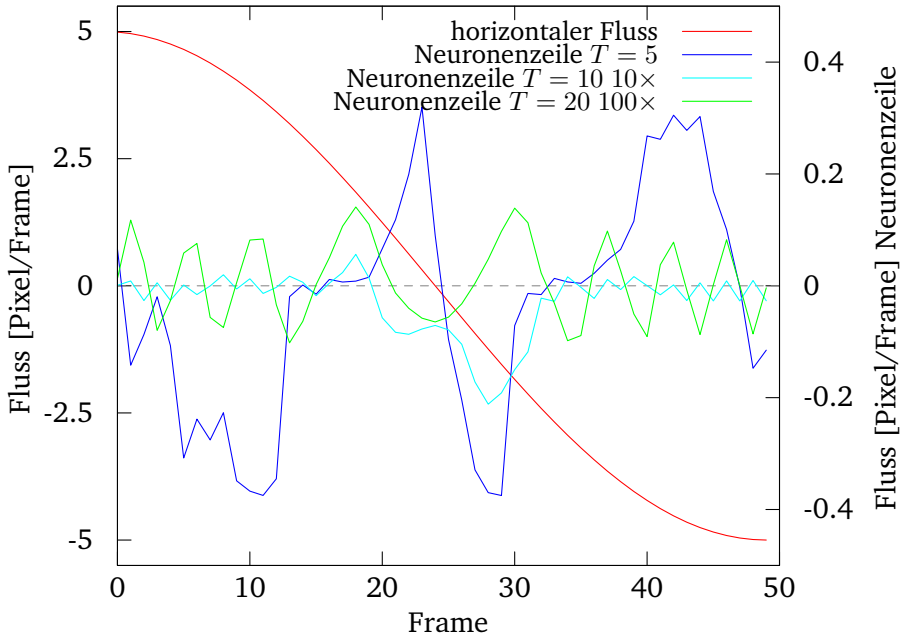

Abbildung A.43.: *Optischer Fluss der einlagigen Neuronenzeile mit Gewichten für natürliche Reize für synthetisches Bildmaterial und unterschiedliche Perioden T des Sinusoiden*

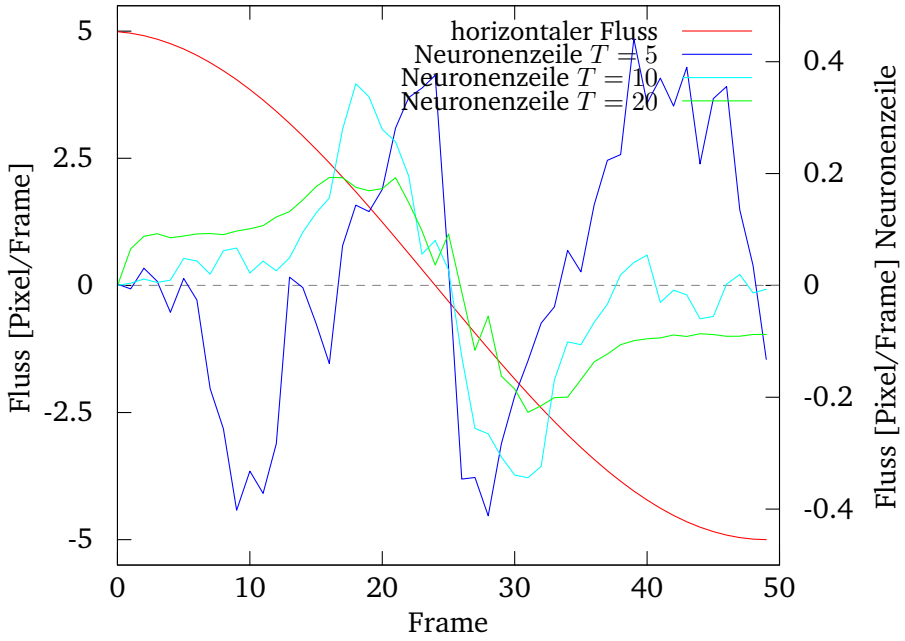

Abbildung A.44.: *Optischer Fluss der zweilagigen Neuronenzeile mit Gewichten für natürliche Reize für synthetisches Bildmaterial und unterschiedliche Perioden T des Sinusoiden*

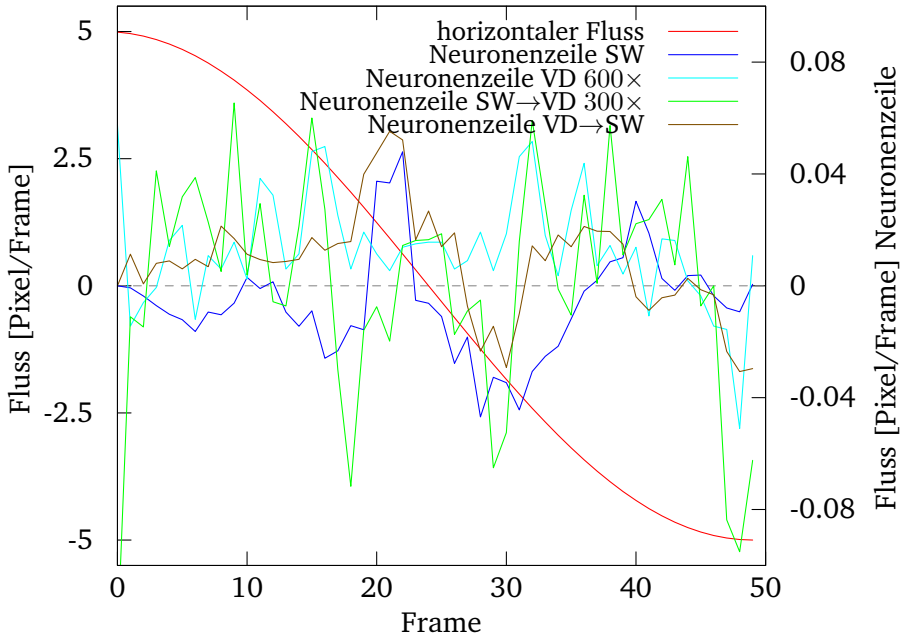

Abbildung A.45.: *Optischer Fluss der einlagigen Neuronenzeile mit Gewichten für natürliche Reize für reale Bilddaten und unterschiedliche Vorverarbeitungsschritte*

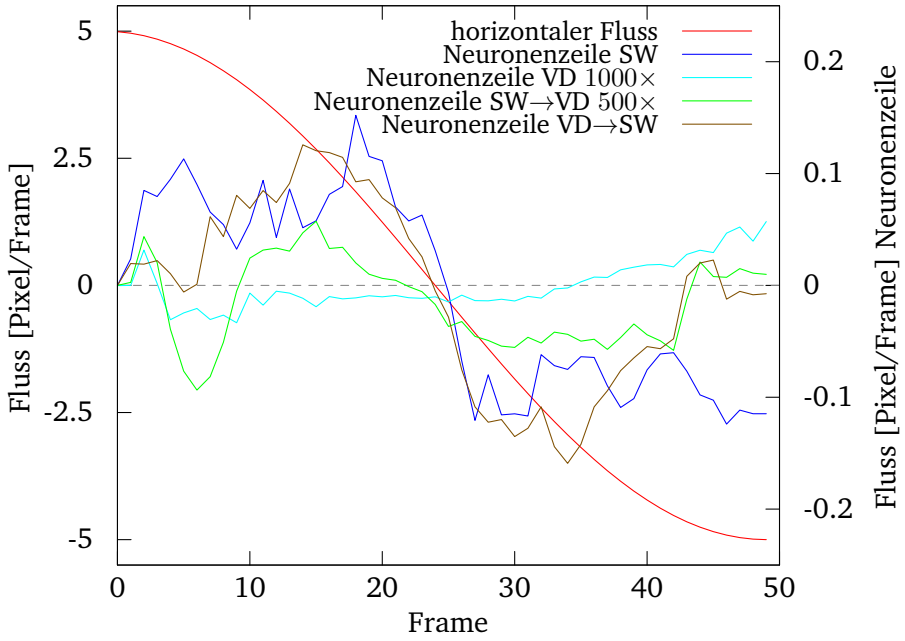

Abbildung A.46.: *Optischer Fluss der zweilagigen Neuronenzeile mit Gewichten für natürliche Reize für reale Bilddaten und unterschiedliche Vorverarbeitungsschritte*

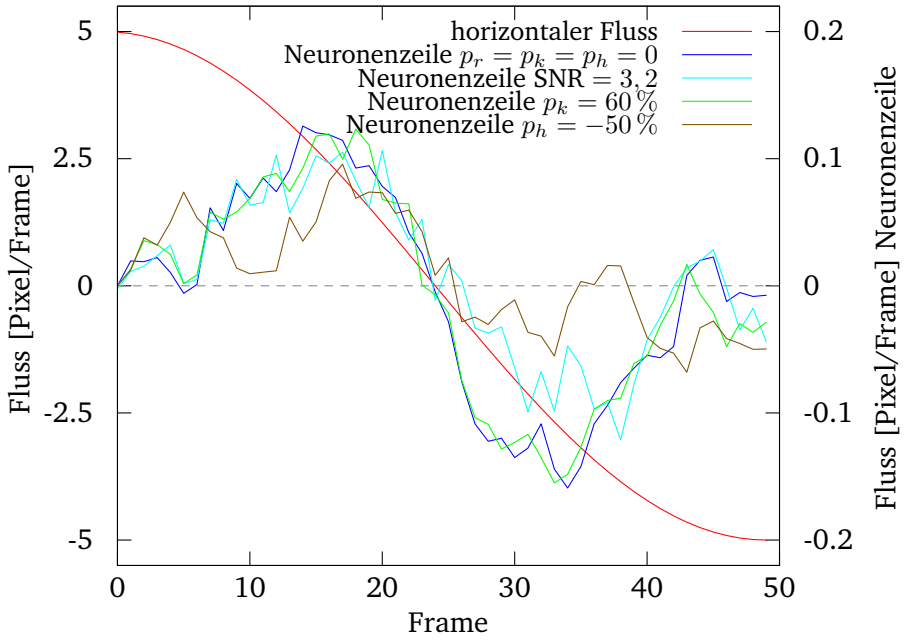

Abbildung A.47.: *Einfluss von Rauschen, Kontrast- und Helligkeitsreduzierung auf die Ausgabe der zweilagigen Neuronenzeile mit vertikaler Durchschnittsbildung und anschließender Schwellwertbildung*

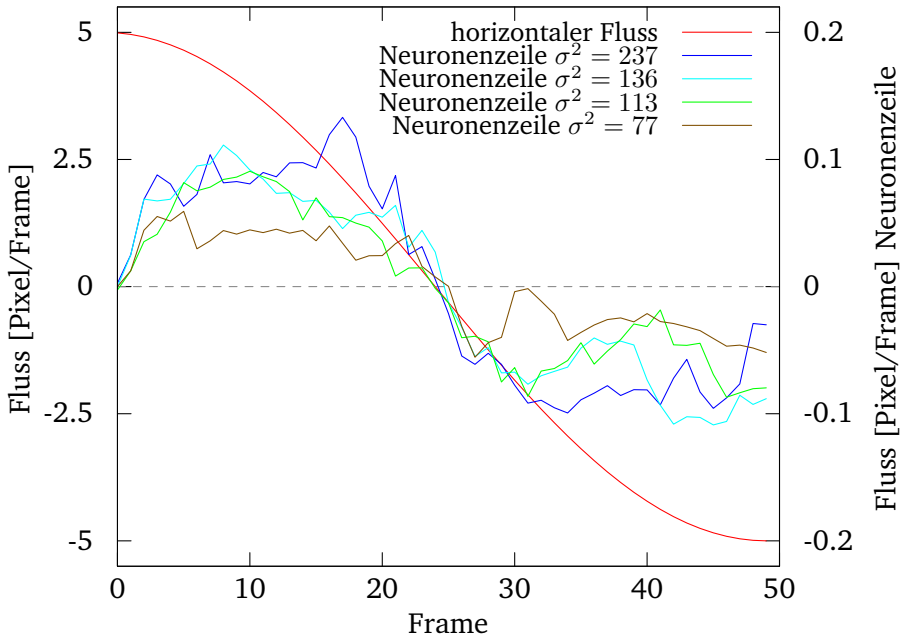

Abbildung A.48.: *Einfluss der Anzahl Kontrastreicher Kanten auf die Ausgabe der zweilagigen Neuronenzeile mit vertikaler Durchschnittsbildung und anschließender Schwellwertbildung*

A.9. Slow-Feature-Analysis

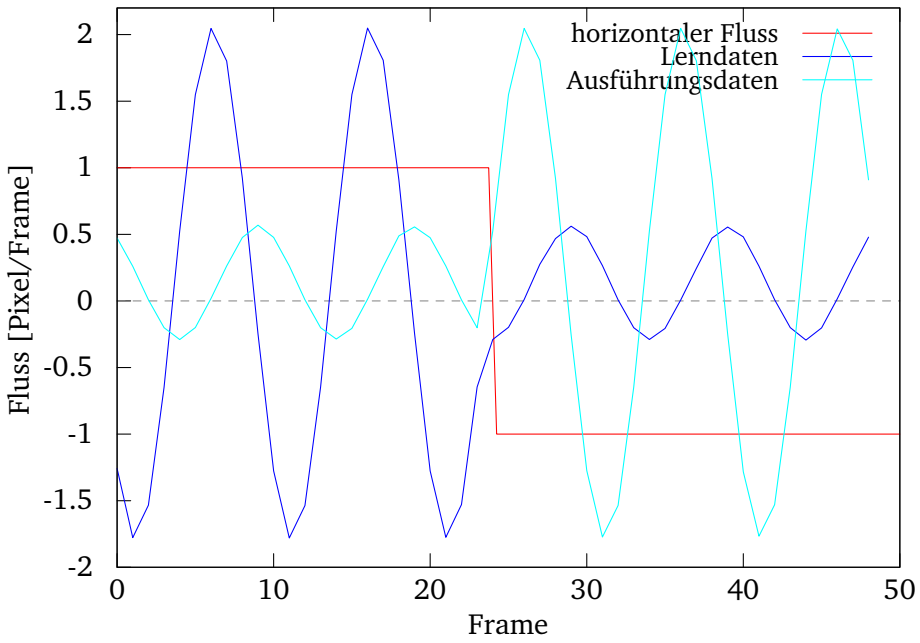

Abbildung A.49.: *Erste Ausgabe y_0 der SFA mit zwei Sensoreingängen, Erweiterung um einen Zeitschritt in die Zukunft und einem Rauschen von $p_r = 0,001$ für synthetisches Bildmaterial mit einer Periode $T = 10$ des Sinusoiden. Die Ausführungsphase wurde mit dem gleichen Bildmaterial erzeugt, jedoch entgegengesetztem Vorzeichen des optischen Flusses.*

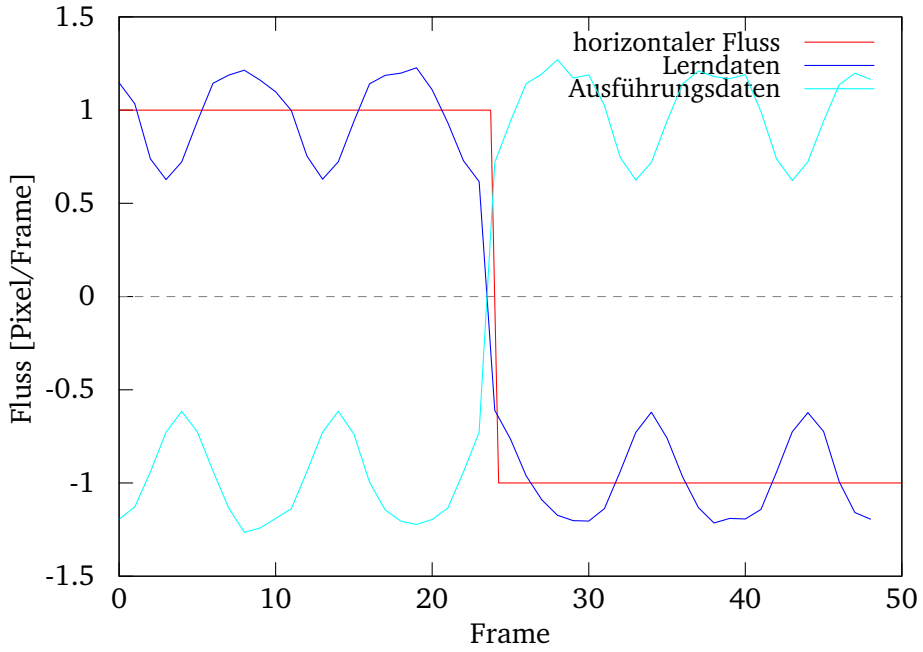

Abbildung A.50.: *Erste Ausgabe y_0 der SFA mit zwei Sensoreingängen, Erweiterung um einen Zeitschritt in die Zukunft, quadratischer Erweiterung und einem Rauschen von $p_r = 0,001$ für synthetisches Bildmaterial mit einer Periode $T = 10$ des Sinusoiden. Die Ausführungsphase wurde mit dem gleichen Bildmaterial erzeugt, jedoch entgegengesetztem Vorzeichen des optischen Flusses.*

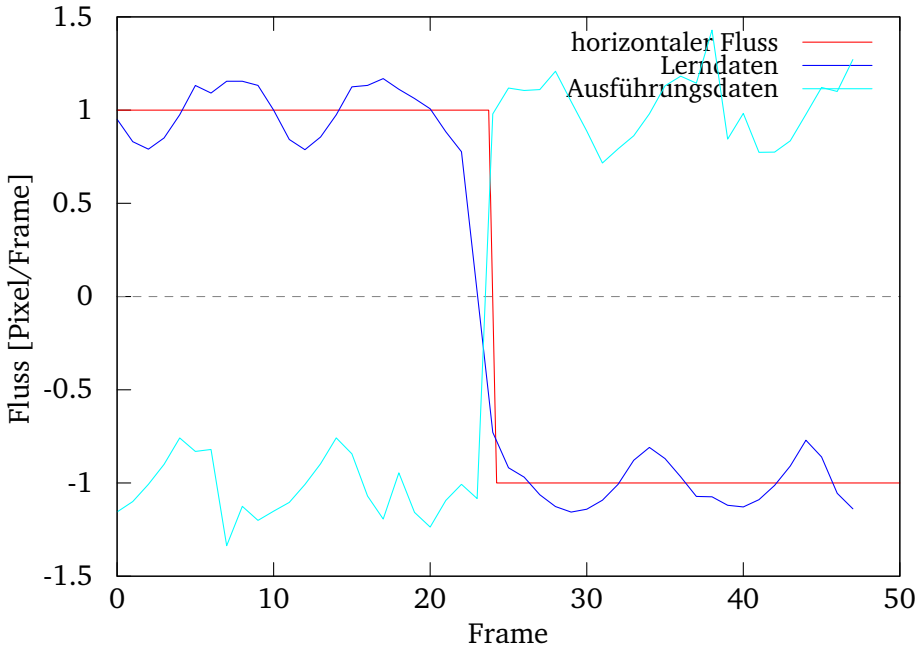

Abbildung A.51.: *Erste Ausgabe y_0 der SFA mit zwei Sensoreingängen, Erweiterung um einen Zeitschritt in die Zukunft, quadratischer Erweiterung und einem Rauschen von $p_r = 0,001$ für synthetisches Bildmaterial mit einer Periode $T = 10$ des Sinusoiden. Die Ausführungsphase wurde mit dem gleichen Bildmaterial erzeugt, jedoch entgegengesetztem Vorzeichen des optischen Flusses.*

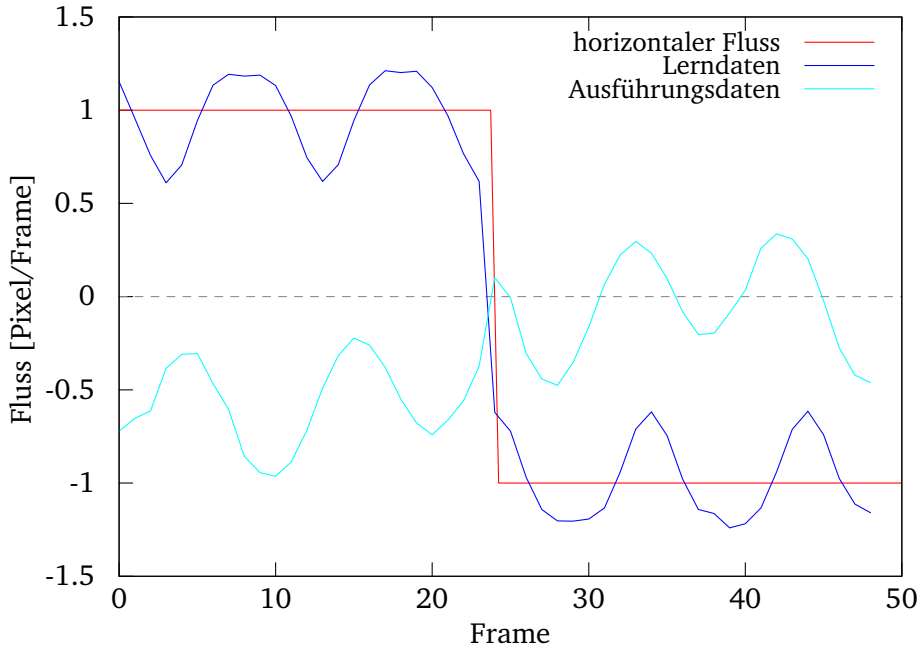

Abbildung A.52.: *Erste Ausgabe y_0 der SFA mit zwei Sensoreingängen, Erweiterung um einen Zeitschritte in die Zukunft, quadratischer Erweiterung und einem Rauschen von $p_r = 0,001$ für synthetisches Bildmaterial mit einer Periode $T = 10$ des Sinusoiden. Die Ausführungsphase wurde mit negativem optischen Fluss und $T = 11$ erzeugt.*

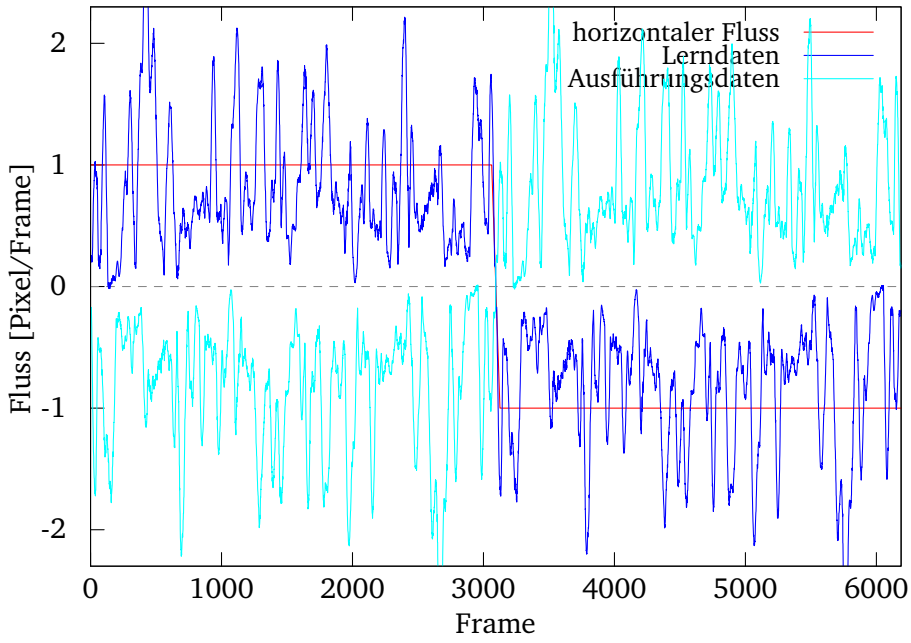

Abbildung A.53.: *Erste Ausgabe y_0 der SFA mit 40 Sensoreingängen, Erweiterung um einen Zeitschritt in die Zukunft, quadratischer Erweiterung und einem Rauschen von $p_r = 0,0001$ für reales Bildmaterial. Die Ausführungsphase wurde mit negativem optischen Fluss erzeugt.*

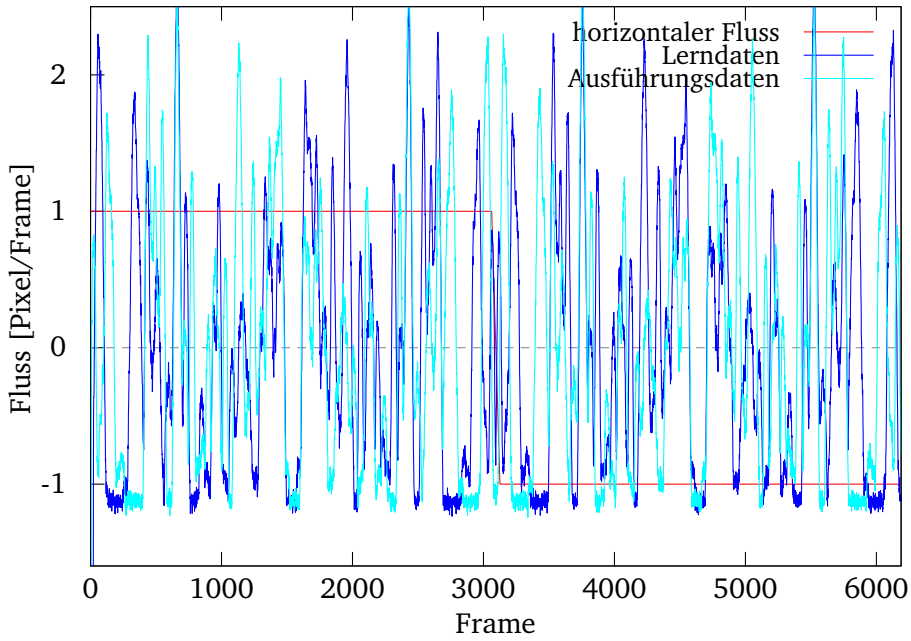

Abbildung A.54.: *Erste Ausgabe y_0 der SFA mit 40 Sensoreingängen, Erweiterung um einen Zeitschritt in die Zukunft, quadratischer Erweiterung und einem Rauschen von $p_r = 0,0001$ für reales Bildmaterial und periodischer Kontrastreduzierung zwischen $p_k = 0\%$ und $p_k = 30\%$ mit einer Periode von 200. Die Ausführungsphase wurde mit negativem optischen Fluss und und einer Kontrastreduzierung von $p_k = 30\%$ erzeugt.*

A.10. Vertikaler Fluss

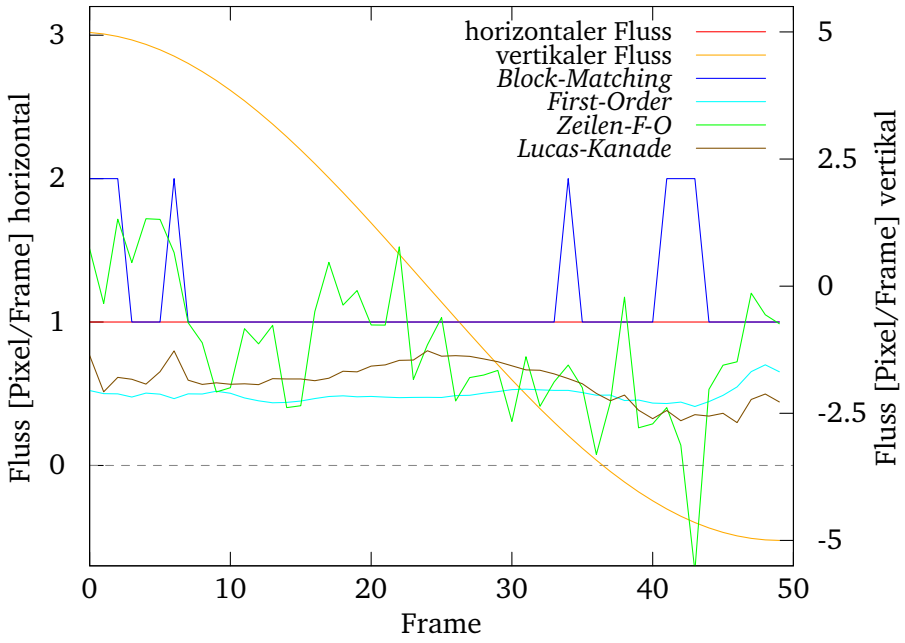

Abbildung A.55.: *Einfluss von störendem vertikalen optischen Fluss auf die Ausgabe der Algorithmen*

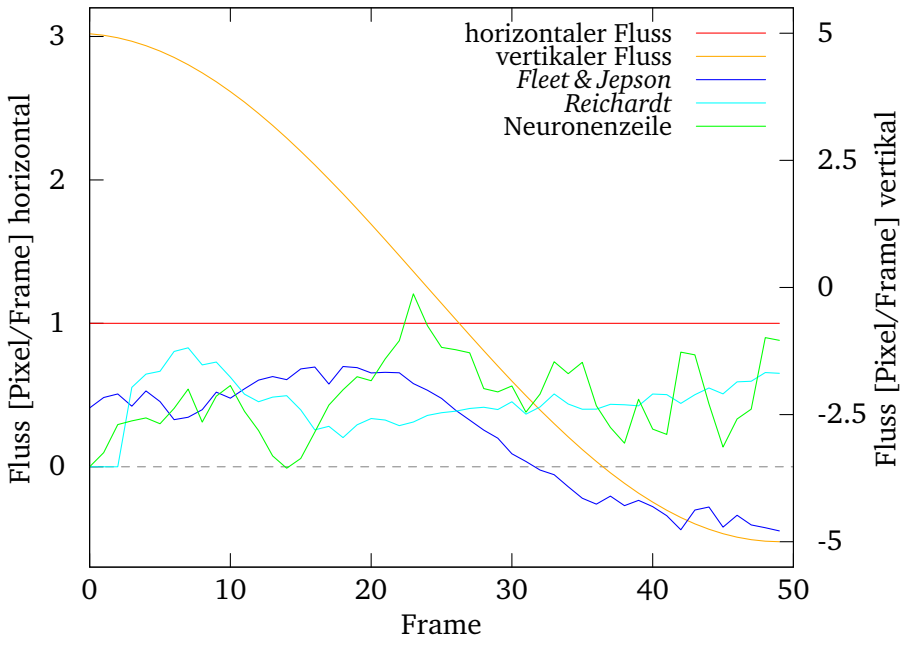

Abbildung A.56.: *Einfluss von störendem vertikalen optischen Fluss auf die Ausgabe der Algorithmen*

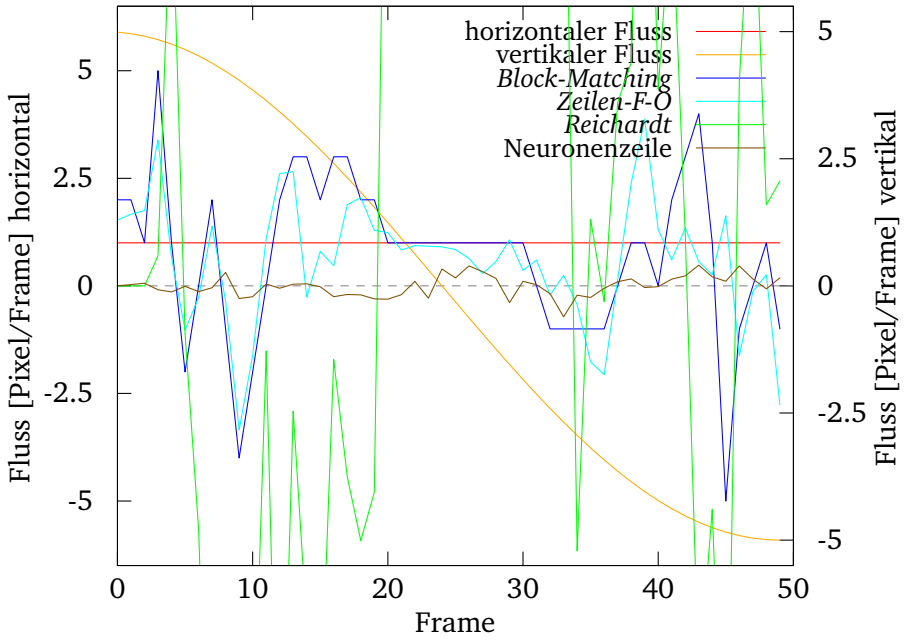

Abbildung A.57.: *Einfluss von störendem vertikalen optischen Fluss auf die Ausgabe der Zeilen-Algorithmen ohne vertikale Durchschnittsbildung*

A.11. Rotation

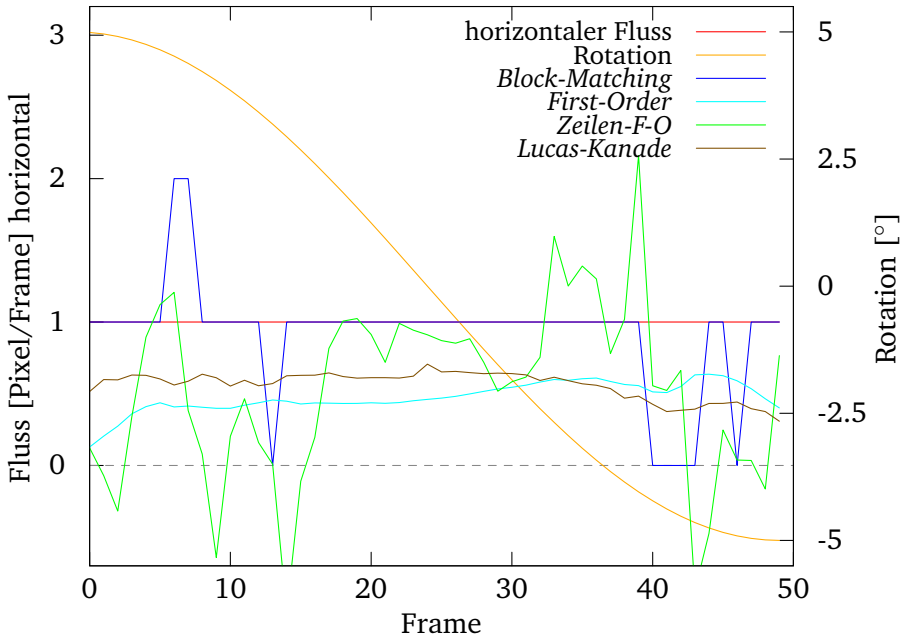

Abbildung A.58.: *Einfluss von störender Rotation auf die Ausgabe der Algorithmen*

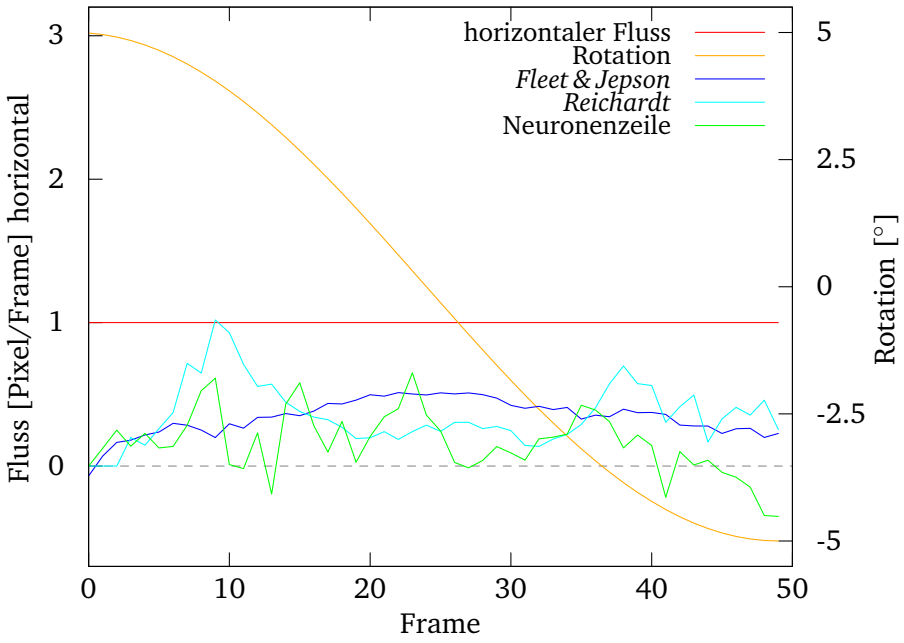

Abbildung A.59.: *Einfluss von störender Rotation auf die Ausgabe der Algorithmen*

B. Quelltexte

Listing B.1: *Programm in Pseudocode für die Bestimmung des optischen Flusses mit Hilfe des optischen Zeilensensors TSL3301*

```
main()
{
  Variable
    //Zeilen–Puffer
    Pixel[102], Pixel_D[102], Pixel_dt[102], Pixel_dx[102],
    akt_D, // Durchschnitt der aktuellen Umgebung
    D_Größe, // Umgebungsgröße für Durchschnittsberechnung
    of, // optischer Fluss der aktuellen Zeile
    gültig, // gültig == 0 ≙ zwei Zyklen ohne lange Unterbrechung
    Zähler, // Schleifenzähler
    df, // Dezimierungsfaktor
    // Variablen für sinc^3–Filter
    int1, int2, int3, diff1, diff2, diff3;

  Variablen auf 0 Initialisieren;
  df = 25; D_Größe = 5;

  System Initialisieren;

  TSL3301 Initialisieren; // ireset, Offset– und Gain–Register = 0
  TSL3301 Integration starten;

  while(1)
  {
    // atomares Kommando an TSL3301
    TSL3301 Integration stoppen,
      Pixelauslesen, Integration starten;

    for (int i = 0; i < 102; i++)
    {
```

```
  of = 0;
  Pixel[i] = TSL3301 Pixel einlesen;
  of += Pixel[i];

  if (i >= D_Größe) akt_D −= Pixel[i − D_Größe];

  // zeitliche Differenzierung
  if (i >= D_Größe − 1)
  {
    Pixel_dt[i] = akt_D − Pixel_D[i];
    Pixel_D[i] = akt_D;
  }
  // räumliche Differenzierung
  if (i >= D_Größe + 1)
  {
    Pixel_dx[i − 1] = Pixel_D[i] − Pixel_D[i − 2];
    if (Pixel_dx[i − 1] != 0)
      of += −100 * Pixel_dt[i − 1] / Pixel_dx[i − 1];
    else
    {
      // Blindrechnung zur Konstanthaltung
      // der Schleifendurchlaufzeit.
      −100 * Piel_dt[i − 1] / 3; of += 0;
    }
  }
}

if (gültig)
{
  gültig−−;

  // Blindbefehle zur Konstanthaltung
  // der Schleifendurchlaufzeit.
  if (2 == 2) 1 += 2 += 3 += 4 += 5;
}
else
{
  // mindestens zwei Durchläufe in Folge
  // ohne längere Unterbrechnung
  Zähler++;
```

```
      int3 += int2;
      int2 += int1;
      int1 += of;

      if (Zähler == df)
      {
        send((int3 – diff1 – diff2 – diff3) / (102*df^3));
        diff3 = int3 – diff1 – diff2;
        diff2 = int3 – diff1;
        diff1 = int3;

        Gültig = 2;
        Zähler = 0;
      }
    }
  }
}
```

Literaturverzeichnis

[AP93] Nicola Ancona and Tomaso Poggio. Optical Flow from 1D Correlation: Application to a simple Time-To-Crash Detector. In *International Journal of Computer Vision*, pages 673–682, 1993.

[Ben10] Christian Benckendorff. Technische Realisierung multimodaler Sensorik für humanoide Roboter. Master's thesis, Humboldt-Universität zu Berlin, 2010.

[BFB94] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12:43–77, 1994.

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 25(11):120,122–125, 2000.

[BSL+09] Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A Database and Evaluation Methodology for Optical Flow. Technical report, Microsoft Research, 2009.

[Dav05] E.R. Davies. *Machine Vision*, chapter 3–4, pages 47–129. Morgan Kaufmann, 3 edition, 2005.

[EB95] H. E. Esch and J. E. Burns. Honeybees use optic flow to measure the distance of a food source. *Naturwissenschaften*, 82(1):38–40, 1995.

[FJ90] David J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *Int. J. Comput. Vision*, 5:77–104, September 1990.

[FJ93] D. J. Fleet and A. D. Jepson. Stability of Phase Information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15:1253–1268, December 1993.

[Gab46] D. Gabor. Theory of Communication. *IEEE*, 93:429–457, 1946.

[GC09] Matthew A. Garratt and Allen Cheung. Obstacle Avoidance in Cluttered Environments using Optic Flow. Technical report, University of New South Wales, University of Queensland, Australia, 2009.

[GIT95] Yonghong Gao, Jouni Isoaho, and Hannu Tenhunen. High Performance Decimation Filter Design for Oversampling Delta-Sigma A/D Converters. Technical report, Royal Institute of Technology - Electronic System Design Laboratory, 1995.

[Hab91] P. Haberäcker. *Digitale Bildverarbeitung*. Hanser Fachbuch, 4 edition, 1991.

[Hil07] Manfred Hild. *Neurodynamische Module zur Bewegungssteuerung autonomer mobiler Roboter*. PhD thesis, Humboldt-Universität zu Berlin, 2007.

[HS80] Berthold K.P. Horn and Brian G. Schunck. Determining Optical Flow. Technical report, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1980.

[IRP94] Michal Irani, Benny Rousso, and Shmuel Peleg. Computing Occluding and Transparent Motions. *International Journal of Computer Vision*, 12:5–16, 1994.

[KH75] C.D. Kuglin and D.C. Hines. The phase correlation image alignment method. In *Proceedings of the IEEE International Conference on Cybernetics and Society*, pages 163–165, 1975.

[Lar94] Rasmus Larsen. *Estimation of visual motion in image sequences*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, DTU, 1994.

[Lee76] David N Lee. A theory of visual control of braking based on information about time-to-collision. *Perception*, 5(4):437–459, 1976.

[LK81] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981.

[LS04] Sooyong Lee and Jae-Bok Song. Mobile Robot Localization Using Optical Flow Sensors. *International Journal of Control, Automation, and Localizat*, 2(4):485–493, 2004.

[Luc84] Bruce D. Lucas. *Generalized Image Matching by the Method of Differences*. PhD thesis, Robotics Institute, Carnegie Mellon University, July 1984.

[MP43] Warren McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133–133, December 1943.

[OGB04] Paul Y. Oh, William E. Green, and Geoffrey Barrows. Neural nets and optical flow for autonomous micro-air-vehicle navigation. In *Proceedings of the ASME International Mechanical Engineering Congress and Exposition*, 2004.

[OH03] Miroslav Oljaca and Tom Hendrick. Combining the ADS1202 with an FPGA Digital Filter for Current Measurement in Motor Control Applications, June 2003. Texas Instruments Application Report.

[Pre00] Letizia Lo Presti. Efficient modified-sinc filters for sigma-delta A/D converters. *IEEE Transactions on Circuits and Systems II*, 47(11):1204–1213, November 2000.

[PRF10] Geoffrey Portelli, Franck Ruffier, and Nicolas Franceschini. Honeybees change their height to restore their optic flow. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 196(4):307–313, April 2010.

[Rei57] W.E. Reichardt. Autokorrelationsauswertung als Funktionsprinzip des Zentralnervensystems. *Zeitschrift Naturforschung*, 12b:448–457, 1957.

[Rei61] W.E. Reichardt. *Autocorrelation, a principle for evaluation of sensory information by the central nervous system*, pages 303–317. John Wiley, New York, 1961.

[RF04] Franck Ruffier and Nicolas Franceschini. Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind

reaction. In *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, 2004.

[Roj96] Raúl Rojas. *Theorie der neuronalen Netze – Eine systematische Einführung*. Springer, 4 edition, 1996.

[Sin90] A. Singh. An estimation-theoretic framework for image-flow computation. Technical report, Department of Computer Science, Columbia University, 1990.

[SK07] Kahlouche Souhila and Achour Karim. Optical Flow based robot obstacle avoidance. Technical report, Centre de Développement des Technologies Avancées, 2007.

[SZLC96] M. V. Srinivasan, S. W. Zhang, M. Lehrer, and T. S. Collett. Honeybee Navigation en route to the Goal: Visual Flight Control and Odometry. *Journal of Experimental Biology*, 199:237–244, 1996.

[Tho01] Richard F. Thompson. *Das Gehirn: Von der Nervenzelle zur Verhaltenssteuerung*. Spektrum Akademischer Verlag, 3 edition, 2001.

[TMT00] Mäenpää Topi, Pietikäinen Matti, and Ojala Timo. Texture Classification by Multi-Predicate Local Binary Pattern Operators. In *Proceedings of the International Conference on Pattern Recognition*, volume 3, pages 3951–3954. IEEE Computer Society, 2000.

[UGVT88] S. Uras, F. Girosi, A. Verri, and V. Torre. A Computational Approach to Motion Perception. *Biological Cybernetics*, 60(2):79–87, 1988.

[VM05] Andrew Vardy and Ralf Möller. Biologically Plausible Visual Homing Methods Based On Optical Flow Techniques. *Connection Science, Special Issue: Navigation*, 17:47–90, 2005.

[vSS85] Jan P. H. van Santen and George Sperling. Elaborated Reichardt detectors. *J. Opt. Soc. Am. A*, 2(2):300–320, 1985.

[Wol07] Andreas Wollstein. Rekurrente neuronale Netze zur Detektion des optischen Flusses. Diplomarbeit, Humboldt-Universität zu Berlin, 2007.

[WS02] Laurenz Wiskott and Terrence J. Sejnowski. Slow feature analysis: unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

[WS07] J. J. Wolken and E. Shin. Photomotion in Euglena gracilis. *Journal of Eukaryotic Microbiology*, 5:39–46, 2007.

[ZL00] Dengsheng Zhang and Guojun Lu. An edge and color oriented optical flow estimation using block matching. In *in Proc. of International Conference on Signal Processing (part of 16th World Computer Congress)*, pages 1026–1032, 2000.

[ZSWS10] Simon Zingg, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. MAV Navigation through Indoor Corridors Using Optical Flow. Technical report, Autonomous Systems Lab, ETH Zurich, 2010. IEEE International Conference on Robotics and Automation (ICRA 2010).

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Einverständniserklärung

Ich erkläre hiermit mein Einverständnis, dass die vorliegende Arbeit in der Bibliothek des Instituts für Informatik der Humboldt-Universität zu Berlin ausgestellt werden darf.

Berlin, den 28. April 2011