Fakultät für Biologie
Wilhelm-Schickard-Institut für Informatik
Eberhard Karls Universität Tübingen

Eigengeschwindigkeitsschätzung einer Flugdrohne mittels optischen Flusses

Diplomarbeit

René Lange

2009

Betreuer

Prof. Dr. Hanspeter A. Mallot
Lehrstuhl Kognitive Neurowissenschaft

Prof. Dr. Andreas Schilling
Lehrstuhl Graphisch-Interaktive Systeme

Eidesstattliche Erklärung

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

Tübingen, den 15. Dezember 2009

Zusammenfassung

In dieser Diplomarbeit wurde die Möglichkeit untersucht, die Eigengeschwindigkeit einer Flugdrohne zu bestimmen, wenn die Informationen dafür aus dem optischen Fluss der Kamerabilder stammen. Zu diesem Zweck wurde ein Verfahren entwickelt, welches mit Hilfe einer Tripelkorrespondenz charakteristische Merkmale im Sichtfeld der Kamera auswählt und eine Eigengeschwindigkeitsänderung berechnet. Das am Lehrstuhl vorhandene *Tracking System* und mehrere virtuelle und reale Filmaufnahmen dienten zur Beurteilung der Leistung und Genauigkeit des Verfahrens.

Danksagung

Diese Diplomarbeit entstand am Lehrstuhl für Kognitive Neurowissenschaft unter der Leitung von Prof. Dr. Hanspeter A. Mallot. Ihm und Prof. Dr. Schilling vom Lehrstuhl für Graphisch-Interaktive Systeme des Wilhelm-Schickard-Instituts für Informatik möchte ich recht herzlich für die Vergabe und Betreuung dieser Diplomarbeit danken.

Danken möchte ich auch meinem Betreuer Fabian Recktenwald, der mir mit Rat und Tat zur Seite stand und sehr viel Verständnis für meine schwierige Anlaufphase zeigte.

Weiterhin möchte ich dem kompletten Lehrstuhl, insbesondere Raum A17, für das vorzügliche Arbeitsklima danken. Explizit sei hier mein Zimmerkollege Tobias Beck genannt, welcher nur so strotzt vor Lebensfreude und der sich ohne murren einen ganzen Arbeitstag mit mir an meinen Rechner gesetzt hat um die abgeschossene Linuxkernel zu reparieren.

Stevie!! Das ist dein Absatz! Vielen Dank, dass ich dich in letzten Wochen, zu den unmöglichsten Zeiten, in Beschlag nehmen durfte.

Nicht zuletzt möchte ich meinen Eltern dafür danken, daß sie mir durch ihre fortwährende Unterstützung das Studium und diese Arbeit ermöglicht haben. Ihnen habe ich am meisten zu danken.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Im Rahmen des EU-Projektes *µDrones - Autonomous Navigation and Environment Sensing*[1] wird ein kleines kompaktes und autonomes UAV (*Unmanned Areal Vehicle*) für Überwachungs- und Erkundungsaufgaben entwickelt. Die baulichen Anforderungen des nach dem VTOL-Prinzip (*Vertical Take Off and Landing*) funktionierenden Quadrocopter ergeben sich hierbei direkt aus dem Einsatzgebiet, welches in räumlich beschränkten Außenbereichen, wie zum Beispiel städtische Umgebungen, anzusiedeln ist. Der Fokus des *µDrones*-Projekts richtet sich darauf, Hard- und Softwaremodule zu entwickeln, welche die Selbständigkeit einer Flugdrohne erhöhen und damit einhergehend die Benutzerfreundlichkeit steigern soll. Hierbei soll die Autonomie der Flugdrohne soweit maximiert werden, dass sie ohne zusätzliche menschliche Befehle und Eingriffe ein einprogrammiertes Missionsziel selbständig ausführen und erfolgreich beenden kann. Als Teil der *µDrones*-Arbeitsgruppe ist es die Aufgabe des Lehrstuhls für Kognitive Neurowissenschaft der Universität Tübingen, sich um die Implementierung von Softwaremodulen zu kümmern, welche auf Basis des optischen Flusses eine Navigation in einer dreidimensionaler Umgebung ermöglicht. Die dazu benötigten Kamerabilder stammen aus der Farbkamera der Flugdrohne. Es handelt sich hierbei um einen Tageslicht Color Sensor mit 470 Zeilen. Die Bilder werden per Funk an die Bodenstation gesendet werden.

In der vorliegenden Arbeit wurde die Möglichkeit untersucht, die Eigengeschwindigkeit einer Flugdrohne zu bestimmen, wenn die Informationen dafür aus dem optischen Fluss der Kamerabilder stammen. Zu diesem Zweck wurde ein Verfahren entwickelt, welches mit Hilfe einer Tripelkorrespondenz charakteristische Merkmale im Sichtfeld der Kamera auswählt und eine Eigengeschwindigkeitsänderung berechnet. Das am Lehrstuhl vorhandene *Tracking System* und mehrere virtuelle und reale Filmaufnahmen dienten zur Beurteilung der Leistung und Genauigkeit des Verfahrens. Für die Bearbeitung dieser Arbeit wurde die Flugdrohne nicht verwendet. Stattdessen kamen für die benötigten

[1] www.ist-microdrones.org

Filmaufnahmen eine *QuickCam Pro 9000* und eine *FinePix S5500* zum Einsatz.

Diese Arbeit ist in fünf Abschnitte unterteilt. In Kapitel 2 wird zunächst auf den biologischen Hintergrund der Arbeit eingegangen. Weiterhin werden die Grundlagen und Voraussetzungen erläutert, welche zum Verständnis der vorliegenden Arbeit essentiell sind. Kapitel 3 beschreibt die in dieser Arbeit entwickelte Methode zur Schätzung der Eigengeschwindigkeit. Es wird erklärt unter welchen Voraussetzungen eine Schätzung überhaupt möglich ist und wie diese erfolgen kann. Anschliessend wird auf programmiertechnische Anpassungen und Erweiterungen eingegangen, welche die Eigengeschwindigkeitsschätzung optimieren. Das Kapitel endet mit einer Beschreibung der Testsequenzen mit welchen das Verfahren überprüft worden ist. Kapitel 4 enthält die Ergebnisse, die schliesslich in Kapitel 5 diskutiert werden.

Kapitel 2

Grundlagen

2.1 Der optische Fluss in der Biologie

Eine wichtige Eigenschaft tierischen Lebens ist die Fähigkeit, sich von einem Ort zum nächsten bewegen zu können. Unabhängig von der Art der Fortbewegung, einschließlich schwimmen, kriechen, fliegen und laufen, ist die Kontrolle der Bewegung durch ein stetiges akkurates Zusammenspiel verschiedener propriospezifischer und exterospezifischer Sensoren gekennzeichnet. Dabei geben propriospezifische Sensoren Auskunft über die räumliche Lage der Gelenke und des Körpers und exterospezifische Sensoren erfassen Informationen aus der Umgebung des Individuums [17][21]. Ein Großteil der Informationen über die Umwelt stammt dabei aus den verarbeiteten Informationen des visuellen Systems.

Der Begriff des optischen Flusses geht auf den Psychologen J. Gibson zurück. Er postulierte, dass Eigenbewegung ein wichtiger Bestandteil der visuellen Wahrnehmung ist und dass Eigenbewegung und Wahrnehmung sich gegenseitig bedingen. Demnach entstehen in einem sich bewegenden visuellen System charakteristische Reizmuster in den Sensoren. Der sich daraus ergebende optische Fluss entsteht durch die relative Bewegung der Objekte der Umgebung und des Betrachters und kann wirksam für Navigationsaufgaben verwendet werden [10][11]. Weitere Untersuchungen ergaben, dass tatsächlich Neuronen existieren, welche große Teile des Blickfelds abdecken, bewegungs- und richtungssensitiv sind und der Verarbeitung des optischen Flusses zugeschrieben werden können [19][20]. So konnte beispielsweise gezeigt werden, dass Bienen und Menschen den optischen Fluss zur Navigation und Objektvermeidung benutzen [29][31].

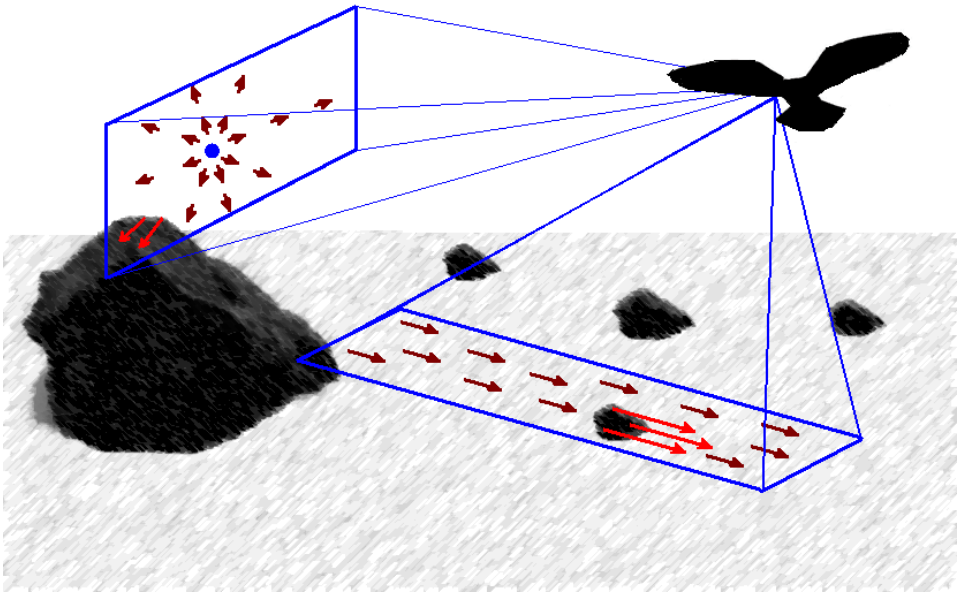


Abbildung 2.1: Mögliche Flusslinien aus der Sicht eines Vogels.

2.2 Der optische Fluss in der Informatik

Die Bestimmung des Bewegungsfeldes (engl. *Motion Field*) ist bei der Verarbeitung von Bewegungsabläufen eines visuellen Systems im Allgemeinen das erste Ziel. Beim Bewegungsfeld handelt es sich um ein zweidimensionales Vektorfeld auf der Bildebene, welches eine Projektionsfläche des dreidimensionalen Raumes darstellt. Bewegt sich ein Objekt relativ zu dieser Bildebene, bewegt sich die zweidimensionale Projektion dieses Objektes innerhalb des Bildes. Aus dem Bewegungsfeld lassen sich Informationen für Bildsegmentierung, Eigenbewegung, räumliche Tiefe und Objektvermeidung ableiten. Zur Berechnung existieren verschiedene Techniken, welche alle miteinander nur Approximationen darstellen und deshalb alternativ die Bezeichnung optischer Fluss (engl. *Optical Flow*) erhalten. Die zugrunde liegende Idee ist, dass momentane Bewegungen von lokalen Regionen des Bildhelligkeitsmusters (engl. *Image Brightness Pattern*) von einem Bild (engl. *Frame*) zum nächsten gemessen werden. Dabei wird angenommen, dass die Helligkeitsmuster bei paarweiser Betrachtung der Bilder erhalten bleiben [27].

Um die prinzipielle Vorgehensweise bei der Berechnung des optischen Flusses zu erläutern, wird für die folgenden Abschnitte der *Lucas Kanade Feature Tracker* nach [5] herangezogen. Es handelt sich hierbei um eine Methode, welche

lokale Ableitungen ersten Grades verwendet, um die Bildbewegung zwischen zwei aufeinanderfolgenden *Frames* zu bestimmen. Der Vorteil des *Lucas Kanade Feature Tracker* gegenüber anderen Methoden beruht auf dem guten Leistungsverhalten, welches sich in vergleichsweise niedrigen Berechnungskosten, einer guten Rauschtoleranz und zuverlässigen Ergebnissen wiederspiegelt [3][9]. Aufbauend darauf wird im letzten Abschnitt das *Generalized Dynamic Image Model* gemäß [25] eingeführt, welches eine Berechnung des optischen Flusses bei Helligkeitsänderungen erlaubt.

2.2.1 Allgemeine Berechnung

Dieser Abschnitt ist eine Zusammenfassung nach [4] und [15]. Zusätzliche Quellen werden kenntlich gemacht.

Bei der Bestimmung des optischen Flusses wird die sichtbare Bewegung der Bildhelligkeit zwischen zwei aufeinanderfolgenden *Frames* erfasst. Die Leuchtdichte in beiden *Frames* wird dabei als konstant angenommen[1]. Eine eindeutige Beschreibung der Helligkeitswerte erfolgt im Weiteren über die Einführung einer Helligkeitsfunktion $I(x, y, t)$, welche jeden Helligkeitswert eindeutig über die Bildkoordinaten (x, y) und die Zeit t definiert. Dieser Ansatz ermöglicht eine Beschreibung der Pixelbewegung dahingehend wie schnell und in welche Richtung sich jeder Helligkeitswert durch die gegebene Bildsequenz bewegt. Eine kleine lokale Positionsverschiebung eines Helligkeitswertes um $(\delta x, \delta y)$ in der Zeit δt lässt sich dann mit Hilfe einer Taylorreihe formulieren. Die neue Position des Helligkeitswertes lautet:

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + H.O.T. \quad . \quad (2.1)$$

Die Abkürzung $H.O.T.$ steht hierbei für die Terme höherer Ordnung der Taylorreihe (engl. *Higher Order Terms*). Diese können aufgrund der infinitisimalen Änderungen der Helligkeitswerte bezüglich Raum und Zeit vernachlässigt werden. Das Prinzip der Pixelverschiebung ist in Abbildung 2.2 nochmals veranschaulicht. Mit Hilfe der obigen Gleichung ist es nun möglich die Hauptbedingungsgleichung des optischen Flusses (engl. *2D Motion Contraint Equation*) zu formulieren. Dazu wird die erste Ableitung der Taylorreihe wie folgt umge-

[1] *Brightness Constancy Assumption*

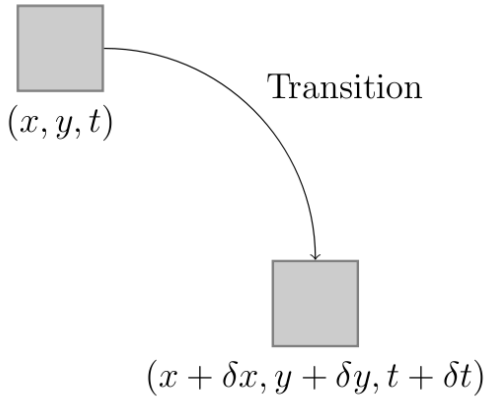


Abbildung 2.2: Betrachtet werden zwei korrespondierende Punkte zum Zeitpunkt t und $t + \delta t$ sowie eine Positionsverschiebung der ursprünglichen Koordinaten (x, y) um δx und δy. Die Bestimmung des optischen Flusses macht eine Berechnung des Transitionsvektors $(\delta x, \delta y)$ notwendig.

formt:

$$\begin{aligned} \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t &= 0 \\ \frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} &= 0 \\ \frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y &= -\frac{\partial I}{\partial t} \quad . \end{aligned} \tag{2.2}$$

Der Vektor $\mathbf{v} = (v_x, v_y) = (\frac{\delta x}{\delta t}, \frac{\delta y}{\delta t})$ beschreibt hierbei die Geschwindigkeit eines sich bewegenden Objekts in x- und y-Richtung und entspricht damit dem optischen Fluss an Position (x, y) zur Zeit t. Die Terme $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$ und $I_t = \frac{\partial I}{\partial t}$ bezeichnen die partiellen Ableitungen der Intensitätsfunktion I und lassen sich beispielsweise mit Hilfe der Simoncelli-Filter nach [28] präzise ermitteln. Eine kompaktere Schreibweise der *2D Motion Contraint Equation* kann nun wie folgt angegeben werden:

$$\begin{aligned} (I_x, I_y) \cdot (v_x, v_y) &= -I_t \\ \nabla I \cdot \mathbf{v} &= -I_t \quad . \end{aligned} \tag{2.3}$$

Es ist ersichtlich, dass die Gleichung (2.3) unterbestimmt ist und die beiden Komponenten von $\mathbf{v}$ nicht ohne weitere Nebenbedingungen auflösbar sind. Diese Unterbestimmtheit wird als Blendenproblem[2] (engl. *aperture problem*) bezeichnet, beschreibt die Mehrdeutigkeit des optischen Flussvektors $\mathbf{v}$ und ist in Abbildung 2.3 veranschaulicht. Aufgrund unzureichender lokaler Helligkeitss-

[2]Spezialfall des allgemeineren Korrespondenzproblems (engl. *generalized aperture problem*)

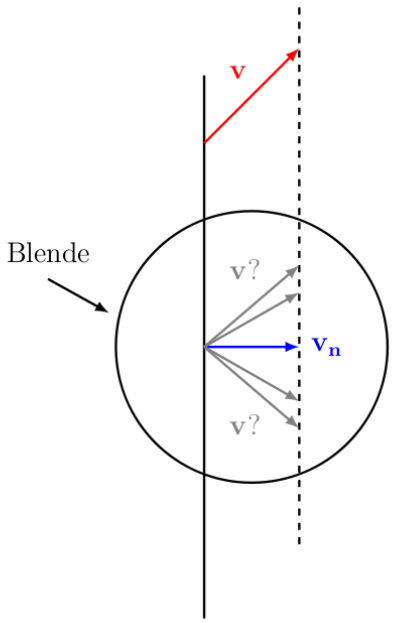


Abbildung 2.3: Das Blendenproblem in der Bewegungsanalyse: Die durchgehende Kante erfährt eine Positionsveränderung zur gestrichelten Linie. Aufgrund der Blende ist es nicht möglich zu unterscheiden, ob die Positionsveränderung horizontal oder diagonal erfolgt ist. Ausschließlich der Normalenvektor $\mathbf{v_n}$ ist bestimmbar.

trukturen ist es beispielsweise nicht möglich, eine sich bewegende Kante im folgenden *Frame* einer Bildsequenz wiederzufinden. Nur die Geschwindigkeitsnormale $\mathbf{v_n}$, welche sich senkrecht zur Kante befindet, kann mit Hilfe von Gleichung (2.4) ermittelt werden.

$$\mathbf{v_n} = \frac{-I_t}{\|\nabla I\|_2} \frac{(I_x, I_y)}{\|\nabla I\|_2} = \frac{-I_t(I_x, I_y)}{\|\nabla I\|_2^2} \tag{2.4}$$

Für die Berechnung sind ausschließlich die partiellen Ableitungen der Intensitätsfunktion nötig, wobei $\|\dots\|_p$ mit $p = 2$ die euklidische Norm repräsentiert. Zur Gewinnung weiterer Nebenbedingungen benutzt der im folgenden Abschnitt vorgestellte *Lucas Kanade Feature Tracker* eine Nachbarschaftsanalyse, gekoppelt mit einer zu minimierenden Kostenfunktion. Diese zusätzlichen Beschränkungen führen zu einen vollständig bestimmten Gleichungssystem, dessen Lösung einen eindeutig bestimmten Geschwindigkeitsvektor $\mathbf{v}$ liefert. Die Motivation der Nachbarschaftsanalyse ist dabei in der sogenannte *Smoothness Assumption* begründet, welche besagt, dass ein Großteil der wahrnehmbaren visuellen Bewegungen durch kontinuierliche, starre Bewegungen rigider, diskreter Objekte verursacht wird. Dadurch sind sich benachbarte Objektpunkte hinsichtlich ihrer Verschiebungsrichtung und Geschwindigkeit ähnlich und erzeugen ein Flussfeld mit feinen Übergängen. Sprunghafte Übergänge entste-

hen nur durch Verdeckung von Objektkanten aufgrund von Tiefeninformationen [25].

2.2.2 *Lucas Kanade Feature Tracker*

Dieser Abschnitt ist eine Zusammenfassung nach [5]. Zusätzliche Quellen werden kenntlich gemacht.

Zur Berechnung des optischen Flusses werden zwei aufeinanderfolgende *Frames* $I = I(x, y)$ und $J = J(x, y)$. Für einen gegebenen Punkt $\mathbf{p} = (p_x, p_y)$ in I soll nun der Verschiebungsvektor $\mathbf{v} = (v_x, v_y)$ ermittelt werden. Hierzu ist die Position $\mathbf{p}' = \mathbf{p} + \mathbf{v} = (p_x + v_x, p_y + v_y)$ in J zu bestimmen, so dass $I(\mathbf{p}) = J(\mathbf{p}')$ gilt. Zur Lösung des bereits im vorherigen Abschnitt erwähnten Blendenproblems wird eine Nachbarschaftsanalyse verwendet. Die Größe dieses sogenannten Integrationsfensters ist mit $(2w_x + 1) \times (2w_y + 1)$ von den beiden Parametern w_x und w_y abhängig. Übliche Zahlenwerte liegen im ganzzahligen Bereich von zwei bis sieben Pixel und haben einen Einfluss auf die lokale Genauigkeit und Robustheit des Algorithmus. Hierbei erlaubt ein kleines Integrationsfenster das erkennen kleiner Bilddetails, während ein großes Integrationsfenster es ermöglicht, ausgedehnte Bildbewegungen zu detektieren und zu verfolgen. Die hieraus entstehende Diskrepanz kann nur mit Hilfe von Zusatzwissen[3] über die zu analysierenden Bilder gelöst werden. Der Unterschied zwischen diesen beiden Möglichkeiten wird mit Hilfe einer pyramidalen Bildrepräsentation verkleinert. Dazu werden jeweils verschiedene Darstellungsebenen der einzelnen *Frames* berechnet (siehe Abbildung 2.4). Für das Bild I gilt: Es entspricht der obersten Ebene der Pyramide und wird mit $I^0 = I$ bezeichnet. Die Breite und Höhe werden mit $n_x^0 = n_x$ und $n_y^0 = n_y$ angegeben. Ausgehend von I^0 werden nun weitere detailärmere Ebenen rekursiv berechnet. Für $0 \leq 2x \leq n_x^{L-1} - 1$ und $0 \leq 2y \leq n_y^{L-1} - 1$ ist die Rekursionsvorschrift durch folgende Gleichung definiert:

$$\begin{aligned} I^L(x,y) = \frac{1}{4} I^{L-1}(2x, 2y) \\ + \frac{1}{8}\Big[I^{L-1}(2x-1, 2y) + I^{L-1}(2x+1, 2y) \\ + I^{L-1}(2x, 2y-1) + I^{L-1}(2x, 2y+1)\Big] \\ + \frac{1}{16}\Big[I^{L-1}(2x-1, 2y-1) + I^{L-1}(2x+1, 2y+1) \\ + I^{L-1}(2x-1, 2y+1) + I^{L-1}(2x+, 2y-1)\Big]. \end{aligned} \tag{2.5}$$

Die Rekursionstiefe L_m $(L = 0, ..., L_m)$ der pyramidalen Bildrepräsentation liegt gewöhnlich im Bereich von zwei und vier. Zur Veranschaulichung wurden

[3] Unter Vorwissen soll z.B. die Geschwindigkeit der Kamerafahrt verstanden werden.

in Abbildung 2.5 verschiedene Darstellungsebenen eines Beispielbildes berechnet. Hierbei wurde die originale Auflösung von 640×480 Bildpunkten schrittweise auf 320×240, 160×120, 80×60, 40×30 und 20×15 reduziert. Mit Hilfe

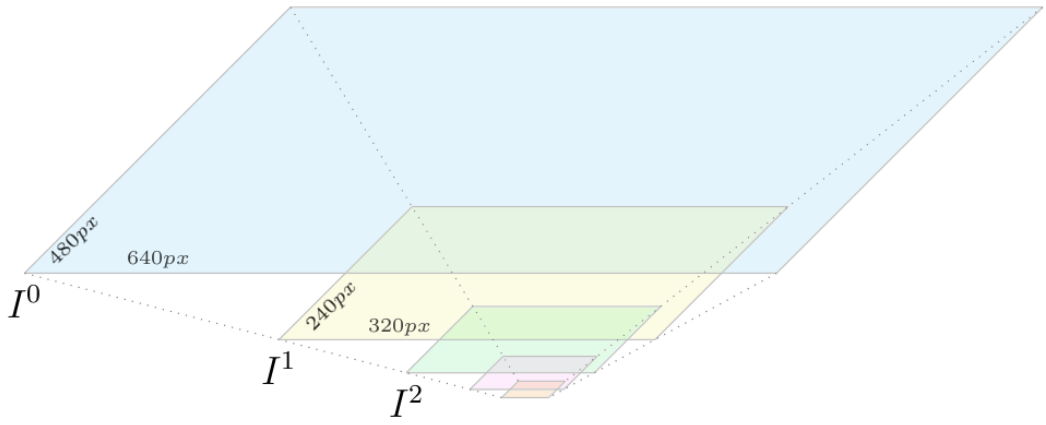


Abbildung 2.4: Schematische Darstellung der pyramidalen Bildrepräsentation. Dies ermöglicht die Erkennung großer Pixelbewegungen.

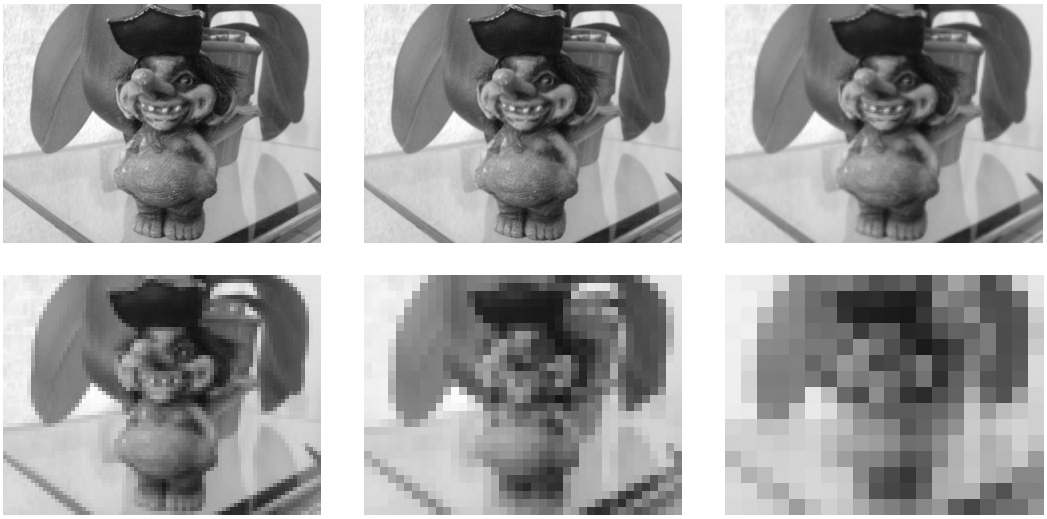


Abbildung 2.5: Beispiel für die pyramidale Repräsentation eines Bildes. Mit Hilfe einer Rekursion werden verschiedene Detailstufen des Ausgangsbildes (links oben) analysiert.

der pyramidalen Bildrepräsentation wird nun, beginnend auf der tiefsten Ebene, der optische Fluss berechnet. Der in der aktuellen Ebene ermittelte optische Fluss dient dabei als anfängliche Schätzung für die nächsthöhere Ebene und wird mit $\mathbf{g} = (g_x, g_y)$ bezeichnet. Die Propagierung der Ergebnisse zwischen

den einzelnen Ebenen wird mit Hilfe der folgenden Kostenfunktion realisiert:

$$\epsilon^L(\mathbf{v}^L) = \sum_{x=p_x^L-w_x}^{p_x^L+w_x} \sum_{x=p_y^L-w_y}^{p_y^L+w_y} (I(x,y) - J(x+g_x^L+v_x^L, y+g_y^L+v_y^L))^2. \quad (2.6)$$

Beginnend auf der tiefsten Ebene der Pyramide wird demnach der Vektor $\mathbf{v}^L = (v_x^L, v_y^L)$ gesucht, der die Kostenfunktion ϵ^L minimiert. Da für die Ebene L_m kein anfänglicher Schätzwert zur Verfügung steht, wird der Vektor $\mathbf{g}^L$ mit $L = L_m$ auf Null initialisiert. Für die nächsthöheren Ebenen lässt sich der initiale Schätzwert nach folgender Gleichung berechnen:

$$\mathbf{g}^{L-1} = 2(\mathbf{g}^L + \mathbf{v}^L). \quad (2.7)$$

Im Anschluss der Berechnung ist für jede Ebene der pyramidalen Bildrepräsentation der endgültige optische Flussvektor $\mathbf{v}$ mit $\mathbf{g}^0 + \mathbf{v}^0$ bestimmbar. Die bisherigen Erklärungen führen zur pyramidalen Implementierung des *Lucas Kanade Feature Tracker*, welche in Form eines Pseudocodes (siehe Algorithm 1 auf der nächsten Seite) veranschaulicht ist.

2.2.3 Merkmalsauswahl in einem Bild

Dieser Abschnitt ist eine Zusammenfassung nach [5]. Zusätzliche Quellen werden kenntlich gemacht.

Der im vorherigen Abschnitt vorgestellte Algorithmus erlaubt es den Verschiebungsvektor $\mathbf{v}$ zwischen den korrespondierenden Bildpunkten $I(\mathbf{p})$ und $J(\mathbf{p}')$ zu bestimmen. Ein relevanter Vorverarbeitungschritt ist die Auswahl des Bildpunktes $\mathbf{p}$, welchen es zu verfolgen gilt. Dies ist nötig, da nicht jeder Bildpunkt gleich gut für eine optische Flussberechnung geeignet ist. Dies wird als Merkmalsauswahl (engl. *Feature-Selection*) bezeichnet und erfolgt über die Eigenwertberechnung der invertierbaren Gradientenmatrix G (siehe Schritt 6 und Schritt 12 im Algorithm 1 auf der nächsten Seite). Der Auswahlprozess lässt sich in den folgenden Schritten zusammenfassen:

1. Berechne für jedes Pixel in Bild I die Gradientenmatrix G und deren kleinsten Eigenwert λ_m.
2. Das größte berechnete λ_m wird als λ_{max} bezeichnet.
3. Verwerfe alle Bildpunkte, deren λ_m kleiner als ein festgelegter Bruchteil (5-10%) von λ_{max} sind.
4. Aus der gewonnenen Menge wähle die Bildpunkte, deren λ_m am größten in einer 3×3 Nachbarschaft ist. Verwerfe die restlichen Bildpunkte.

Algorithm 1 PYRAMIDAL-LK-FT $({I^L}_{L=0,\ldots,L_m},\ {J^L}_{L=0,\ldots,L_m})$

1. Initialize: $\mathbf{g}^{L_m} = [g_x^{L_m} \quad g_y^{L_m}]^T = [0 \quad 0]^T$
2. **for** $L = L_m$ to 0 **do**
3. *Calculate* $\mathbf{p}$ *in* I^L*:* $\mathbf{p}^L = [p_x \quad p_y]^T = \mathbf{p}/2^L$
4. *Determine:* $I_x(x,y) = \dfrac{I^L(x+1,y) - I^L(x-1,y)}{2}$
5. *Determine:* $I_y(x,y) = \dfrac{I^L(x,y+1) - I^L(x,y-1)}{2}$
6. *Calculate spatial gradient matrix:*
$$G = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{x=p_y-w_y}^{p_y+w_y} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_x^2(x,y) \end{bmatrix}$$
7. *Initialize:* $\mathbf{d}^0 = [0 \quad 0]^T$
8. *Initialize:* $k = 1$
9. **while** $k \leq K$ **or** $||\eta^k|| \geq$ accuracy threshold **do**
10. *Calculate image difference:*
$$\delta I_k(x,y) = I^L(x,y) - J^L(x + g_x^L + d_x^{(k-1)}, y + g_y^L + d_y^{(k-1)})$$
11. *Calculate image mismatch vector:*
$$\mathbf{b}_k = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{x=p_y-w_y}^{p_y+w_y} \begin{bmatrix} \delta I_k(x,y) I_x(x,y) \\ \delta I_k(x,y) I_y(x,y) \end{bmatrix}$$
12. *Optical Flow:* $\boldsymbol{\eta}^k = G^{-1}\mathbf{b}_k$
13. *Guess for next iteration:* $\mathbf{d}^k = \mathbf{d}^{k-1} + \boldsymbol{\eta}^k$
14. *Increment:* $k = k + 1$
15. **end while**
16. *Optical Flow at Level L:* $\mathbf{v}^L = \mathbf{d}^K$
17. *Guess for Level* $L-1$*:* $\mathbf{g}^{L-1} = [g_x^{L-1} \quad g_y^{L-1}]^T = 2(\mathbf{g}^L + \mathbf{v}^L)$
18. **end for**
19. *Final optical flow:* $\mathbf{v} = \mathbf{g}^0 + \mathbf{v}^0$
20. *Location of* $\mathbf{p}'$ *on J:* $\mathbf{p}' = \mathbf{p} + \mathbf{v}$

5. Filtere die verbliebene Menge von Bildpunkten ein letztes Mal, so dass ein vorher festgelegter Minimalabstand (z.B. 5 oder 10 Pixel) zwischen zwei Bildpunkten gewährleistet ist.

Die nach diesem Auswahlverfahren übrig gebliebenen Bildpunkte sind die sogenannten *Feature* des Bildes. Sie sind in der Regel gut verfolgbar und dienen dem *Feature Tracker* Algorithmus als Eingabe.

2.2.4 *Generalized Dynamic Image Model*

Dieser Abschnitt ist eine Zusammenfassung nach [24] und [25]. Zusätzliche Quellen werden kenntlich gemacht.

Verändert sich die Helligkeit zwischen zwei aufeinanderfolgenden *Frames*, so ist es nach der bisherigen Definition des optischen Flusses nicht - allenfalls unzureichend - möglich ein zutreffendes Verschiebungsfeld zu berechnen. Die Menge der Ereignisse, welche zu einer Änderung der Bildhelligkeit führen, werden hierbei als radiometrische Informationen zusammengefasst. In Bezug darauf sei beispielsweise die Bewegung einer Lichtquelle oder der Schattenwurf genannt. Auch die Manipulation der Kameraeinstellungen durch eine Anpassung der Blendenöffnung und Belichtungszeit, um über- und unterbeleuchtete Szenerien in unkontrollierten Umgebungen optimal darzustellen, zählt hier dazu.
Im nun vorgestellten *Generalized Dynamic Image Model*, welcher als modifizierter *Lucas Kanade Feature Tracker* angesehen werden kann, erfolgt eine Trennung der insgesamt wahrgenommenen Bildtransformation in geometrische und radiometrische Komponenten. Zudem wird die Voraussetzung der Leuchtdichtenkonsistenz der Bildpunkte zwischen benachbarten *Frames* derart gelockert, dass eine lineare Transformation zwischen Helligkeitswerten erlaubt wird. Mathematisch lässt sich dies wie folgt darstellen:

$$I(x + \delta x, y + \delta y, t + \delta t) = M(x, y, t)I(x, y, t) + C(x, y, t) \tag{2.8}$$

Hierbei werden die geometrischen Veränderungen weiterhin implizit durch die Punktkorrespondenzen beschrieben, wohingegen die radiometrischen Transformationen explizit über die Einführung des *Multiplier Field* $M = 1 + \delta m$ und des *Offset Field* $C = 0 + \delta c$ definiert sind. Dies gilt für $\delta t \to 0$ und die Annahme, dass beide Funktionen einer inkrementellen Änderung unterliegen. Erstere Annahme bedingt, dass auch δm und δc gegen Null streben und die zeitlichen Ableitungen m_t und c_t wie folgt bestimmt werden können:

$$m_t = \lim_{\delta t \to 0} \frac{\delta m}{\delta t} \quad \text{und} \quad c_t = \lim_{\delta t \to 0} \frac{\delta c}{\delta t} \quad . \tag{2.9}$$

Erweitert man die Funktion (2.8) linksseitig mit Hilfe der Taylorreihe nach (2.1) und substituiert rechtsseitig die Definitionen des *Multiplier Field* und *Offset Field*, erhält man:

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy - I\delta m + \delta c = -\frac{\partial I}{\partial t}dt \quad . \tag{2.10}$$

Eine abschliessende Division durch δt und eine Grenzwertbetrachtung mit $\delta t \rightarrow 0$ führt zu

$$\begin{aligned} \frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} - I\frac{\delta m}{\delta t} + \frac{\delta c}{\delta t} &= -\frac{\partial I}{\partial t}\frac{\delta t}{\delta t} \\ (I_x, I_y) \cdot (v_x, v_y) - Im_t + c_t &= -I_t \\ \nabla I \cdot \mathbf{v} - Im_t + c_t &= -I_t \quad . \end{aligned} \tag{2.11}$$

Dies stellt die überarbeitete *2D Motion Contraint Equation* dar. Im besonderen Fall erhalten wir für $m_t = c_t = 0$ die Gleichung (2.3). Eine Berechnung von δx, δy, δm und δc erfolgt mit

$$\sum_W \begin{bmatrix} I_x^2 & I_xI_y & -I_xI & -I_x \\ I_xI_y & I_y^2 & -I_yI & -I_y \\ -I_xI & -I_yI & I^2 & I \\ -I_x & -I_y & I & 1 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ \delta m \\ \delta c \end{bmatrix} = \sum_W \begin{bmatrix} -I_xI_t \\ -I_yI_t \\ II_t \\ I_t \end{bmatrix} \tag{2.12}$$

Entsprechend dem *Lucas Kanade Feature Tracker* wird auch hier für jeden zu berechnenden Bildpunkt eine Nachbarschaftsanalyse durchgeführt. In (2.12) ist die Nachbarschaftregion durch W beschrieben und umfasst ein Integrationsfenster von 9×9 Pixel, welches zentral über jeden Bildpunkt gelegt wird. Alle weiteren Einträge sind die bereits vorher formulierten partiellen Ableitungen der Intensitätsfunktion I (siehe Abschnitt 2.2.1). Für eine erfolgreiche Berechnung des jeweiligen Flussvektors ist auch hier notwendig, dass die in Gleichung (2.12) enthaltene Matrix invertierbar ist (siehe Abschnitt 2.2.3).

2.3 Dreidimensionale Interpretation des optischen Flusses

Die Rekonstruktion dreidimensionaler Informationen, ausgehend von einer Serie von Bildern aus unterschiedlicher Perspektive ein und derselben Szene, ist ein zentrales Problem der *Computer Vision*. Dabei ist es egal, ob diese Bilder parallel von einem Multi-Kamerasystem erstellt worden sind, oder nacheinander durch eine einzelne Kamera, welche sich relativ zur besagten Szene bewegt. Ausgehend von den gemessenen Bilddaten, das heisst den Pixelkoordinaten der

Bilder, existieren zwei Möglichkeiten für eine dreidimensionale Rekonstruktion der Szene [12] [23]:

Der erste Ansatz verzichtet auf die Kalibrierung des Kamerasystems und verwendet zur Rekonstruktion ausschließlich Bildpunktkorrespondenzen. Es lässt sich zeigen, dass auf Basis der Bildpunktkorrespondenzen, in der Regel mit wenig Aufwand, die Szenerie bis auf einen Skalierungsfaktor wiederhergestellt werden kann [7] [23].

Die zweite Möglichkeit führt über die Einführung eines Kameramodells, welches eine Beziehung zwischen den zweidimensionalen Pixelkoordinaten und den dreidimensionalen Koordinaten herstellt. Exemplarisch sei hier das Tsai-Kameramodell genannt, welches elf innere und äußere Parameter (auch intrinsische und extrinsische Parameter genannt) besitzt und in einer 3×4 Projektionsmatrix P zusammengefasst wird. P und die darin enthaltenen Kameraparameter, dazu gehören unter anderem die Brennweite der Kamera, die Linsenverzerrung und die Position und Orientierung der Kamera im Raum, können mit Methoden der Kamerakalibrierung berechnet werden.

2.3.1 Kamerakalibrierung nach Tsai

Dieser Abschnitt ist eine Zusammenfassung nach [22] und [30]. Zusätzliche Quellen werden kenntlich gemacht.

Die Methode nach Tsai beruht auf dem Projektionsmodell einer Lochkamera. Eine Lochkamera ist das einfachste Modell einer Kamera, besitzt anstelle eines Objektivs ein einfaches Loch (engl. *pinhole*) und verfügt über eine ideale Abbildungvorschrift von dreidimensionalen Weltkoordinaten zu zweidimensionalen Bildkoordinaten. Die Kalibrierungsmethode nach Tsai ermöglicht es Abbildungsfehler einer Kamera mit photographischem Linsensystem dahingehend zu berechnen, dass eine Korrektur der Bilder ermöglicht wird und die transformierten Bilder den Bildern einer Lochkamera entsprechen. Hierzu müssen die fünf inneren Parameter:

- f - Brennweite,
- κ_1 - Koeffizient zur Beschreibung der radialen Linsenverzerrung,
- c_x, c_y - Koordinaten des Zentrums der radialen Linsenverzerrung und Durchstoßpunkt der optischen Achse des Kamerakoordinatensystems mit der Sensorebene der Kamera,
- s_x - Skalierungsfaktor um mögliche Hardwareschwächen im Framegrabber auszugleichen,

sowie die sechs äußeren Parameter:

R_x, R_y, R_z - Rotationsmatrizen für Abbildung zwischen Weltkoordinaten und Kamerakoordinaten,

t_x, t_y, t_z - Translationskomponenten für Abbildung zwischen Weltkoordinaten und Kamerakoordinaten,

bestimmt werden. Während die intrinsischen Parameter die interne Geometrie der Kamera beschreiben, geben die extrinsischen Parameter Auskunft über die Position und Orientierung der Kamera im Weltkoordinatensystem. Zusätzlich zu den elf Kameraparametern besitzt das Tsai-Modell sechs intrinsische Kamerakonstanten, welche hardwarespezifisch festgelegt sind. Diese lauten:

N_{cx} - Anzahl Sensorelemente der Kamera in x-Richtung,
N_{fx} - Anzahl der Pixel des Framegrabber in x-Richtung,
d_x - Mittenentfernung angrenzender Sensorelemente in x-Richtung,
d_y - Mittenentfernung angrenzender Sensorelemente in y-Richtung,
d_{px} - Pixelgröße im Framegrabber in x-Richtung,
d_{py} - Pixelgröße im Framegrabber in y-Richtung.

Die allumfassende Abbildung der dreidimensionalen Weltkoordinaten in zweidimensionale Pixelkoordinaten läßt sich hierbei in vier Schritte einteilen: Im ersten Schritt werden die dreidimensionalen Weltkoordinaten (x_w, y_w, z_w) auf dreidimensionale Kamerakoordinaten (x, y, z) abgebildet, wobei die Abbildungsvorschrift durch die Gleichung

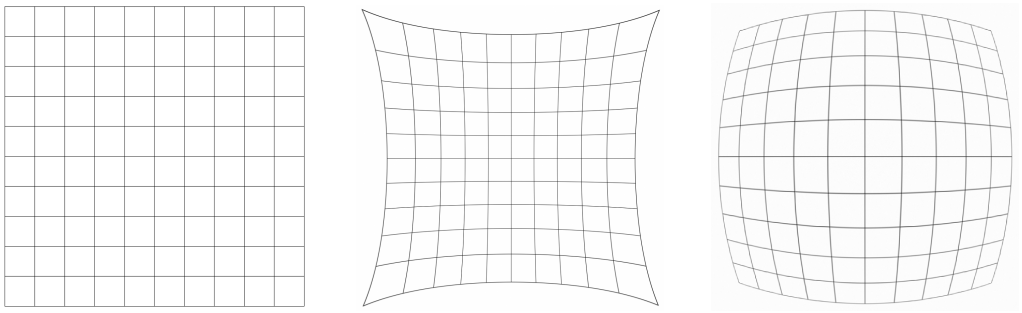
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.13}$$

festgelegt ist. Die 3×3 Rotationsmatrix R wird hierbei über die drei Eulerwinkel θ, ϕ, ψ (Yaw-Pitch-Roll) beschrieben. Im zweiten Schritt werden die dreidimensionalen Kamerakoordinaten (x, y, z) auf zweidimensionale Bildkoordinaten (x_u, y_u) abgebildet.

$$x_u = f\frac{x}{z} \quad , \quad y_u = f\frac{y}{z} \quad . \tag{2.14}$$

Die Indizierung der Bildkoordinaten mit u steht hierbei für unverzerrt (engl. *undistorted*), da die Berechnung nur die Kalibrierung der Brennweite f erfordert. Die Berücksichtigung der radialen Linsenverzerrungen[4] (siehe Abbildung 2.6) und damit die Bestimmung der verzerrten (engl. *distorted*) tatsächlichen

[4]Möglicherweise existierende tangentiale Linsenverzerrungen werden nicht berücksichtigt, da vernachlässigbar.



(a) **(b)** **(c)**

Abbildung 2.6: Für eine genaue 3D-Analyse müssen eventuell vorhandene radiale Linsenverzerrungen ausgeglichen werden. **(a)** keine Verzerrzung. **(b)** Kissenverzerrung. **(c)** Tonnenverzerrung.

Bildkoordinaten (x_d, y_d) erfolgt im dritten Schritt und ist mathematisch wie folgt festgehalten:

$$x_d = x_u(1 + \kappa_1 r^2)^{-1} \quad , \quad y_d = y_u(1 + \kappa_1 r^2)^{-1} \tag{2.15}$$

mit

$$r = \sqrt{x_u^2 + y_u^2} \quad .$$

Der zur Berechnung benötigte Verzerrungskoeffizient κ_1 ist dabei ausreichend genau über ein Polynom zweiten Grades definiert. Der vierte und letzte Schritt dient der Abbildung der tatsächlichen Koordinaten in der Bildebene zu den endgültigen (engl. *final*) Pixelbildkoordinaten (x_f, y_f). Die Berechnung erfolgt mit

$$x_f = \frac{x_d}{s_x d_x} + c_x \quad , \quad y_f = \frac{y_d}{d_y} + c_y \tag{2.16}$$

und setzt eine Kalibrierung des Skalierungsparameter s_x und des Parameterpaares (c_x, c_y) voraus. Um alle erforderlichen Parameter bestimmen zu können benötigt die Methode nach Tsai ein Kalibrierungsobjekt. Dies ist ein Objekt, dessen Geometrie im Weltkoordinatensystem bekannt ist (siehe Schachbrettmuster in Abbildung 2.7). Für eine vollständige Kalibrierung ist es zusätzlich erforderlich, dass die Kalibrierungspunkte nicht in einer Ebene parallel zur Bildebene liegen[5]. Es erfolgt zunächst eine Schätzung der extrinsischen Parameter und der Brennweite f mit Hilfe der Methode der kleinsten Quadrate. In einem zweiten Schritt werden über eine nichtlineare Optimierung, die restlichen Parameter bestimmt und die Schätzungen aller Parameter verfeinert.

[5]Bei Koplanarität der Kalibrierungspunkte ist eine vereinfachte Kalibrierung dennoch möglich. Der Skalierungsfaktor s_x kann dann jedoch nicht berechnet werden.

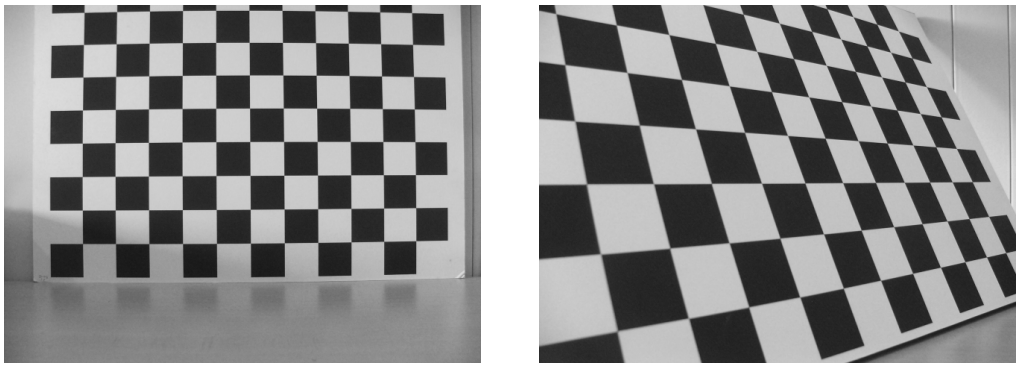


Abbildung 2.7: Eine parallele (*links*) und nichtparallele (*rechts*) Ausrichtung des planen Kalibrierungsobjektes bzgl. der Bildebene entscheidet, ob eine vereinfachte oder vollständige Kalibrierung des Kamerasystems möglich ist.

2.3.2 Eigenbewegungsschätzung nach Kanatani

Dieser Abschnitt ist eine Zusammenfassung nach [16]. Zusätzliche Quellen werden kenntlich gemacht.

Um die Eigengeschwindigkeit der Flugdrohne bestimmen zu können, sind im Weiteren Tiefeninformationen nötig. Eine Weiterverarbeitung der Flussdaten ist daher unerlässlich. Unter der Annahme, dass ein Großteil der Flussvektoren durch die Eigenbewegung (engl. *Egomotion*) der Flugdrohne zustande kommen, wird im Folgenden ein Algorithmus vorgestellt, welcher den optischen Fluss einer allgemeinen Kamerabewegung, relativ zu einer gewöhnlich geformten Oberfläche, als Eingabe verwendet und eine Eigenbewegung- und Tiefenschätzung zurückliefert.

Ein grundlegender Schritt in der Berechnung ist die sphärische Darstellung (engl. *Image Sphere Representation*) der Punkte der Bildebene[6]. Hierzu wird eine hypothetische räumliche Bildoberfläche mit Einheitsradius definiert. Das Zentrum dieser Bildoberfläche entspricht dem Zentrum der Kameralinse und beschreibt den Kamerastandpunkt (engl. *Viewpoint*). Jeder Punkt in der Bildebene kann nunmehr durch einen normierten Vektor $\mathbf{m}$, welcher dem Zentrum entspringt und in Richtung des jeweiligen Bildpunktes zeigt, repräsentiert werden. Geht man davon aus, dass sich die Kamera mit einer Rotationsgeschwindigkeit $\boldsymbol{\omega}$ um den Kamerastandpunkt und einer Translationsgeschwindigkeit $\mathbf{v}$ relativ zu einer Oberfläche bewegt, so wird auf der Bildebene ein optischer Fluss erzeugt, welcher durch die Ableitung der normalisierten Vektoren $\mathbf{m}$ nach der Zeit repräsentiert ist. Hierbei gewährleistet die Normalisierung die Orthogonalität der abgeleiteten, sogenannten *N-velocity*-Vektoren, $\dot{\mathbf{m}}$ zu $\mathbf{m}$. Die Änderungsrate der Blickrichtung der Kamerabewegung $\{\boldsymbol{\omega}, \mathbf{v}\}$ relativ zu einer Oberfläche mit Tiefe $r(\mathbf{m})$ entspricht der Rotation und Translation der

[6] Voraussetzung für diese Transformation ist ein kalibriertes Kamerasystem.

betrachteten Szenerie um $\{-\boldsymbol{\omega}, -\mathbf{v}\}$ und kann durch folgenden Gleichung dargestellt werden:

$$\dot{\mathbf{m}} = -\boldsymbol{\omega} \times \mathbf{m} - \frac{P_m \mathbf{v}}{r(\mathbf{m})}. \tag{2.17}$$

Die Matrix P_m bildet hierbei den Translationsvektor $\mathbf{v}$ orthogonal auf die Ebene der Einheitsoberflächennormalen $\mathbf{m}$. Diese wahrnehmbare[7] Projektion wird durch die Distanz $r(\mathbf{m})$ nachträglich invers skaliert. Die Projektionsmatrix wird definiert:

$$P_m = P_m^T = I - \mathbf{m}\mathbf{m}^T. \tag{2.18}$$

Damit eine Berechnung der Tiefe nach Gleichung (2.23) erfolgen kann, ist es

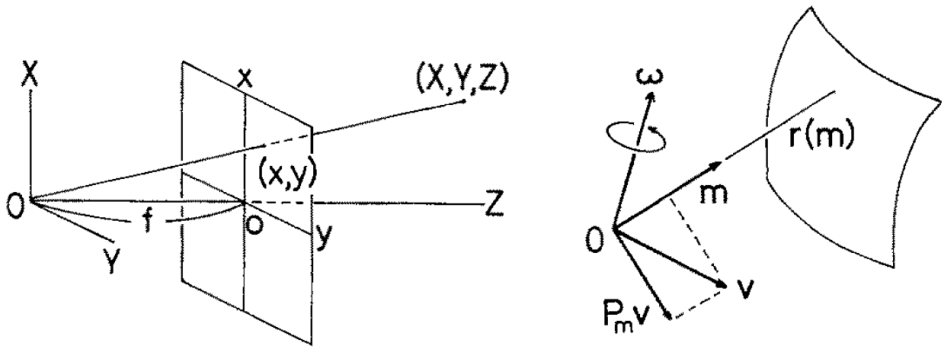
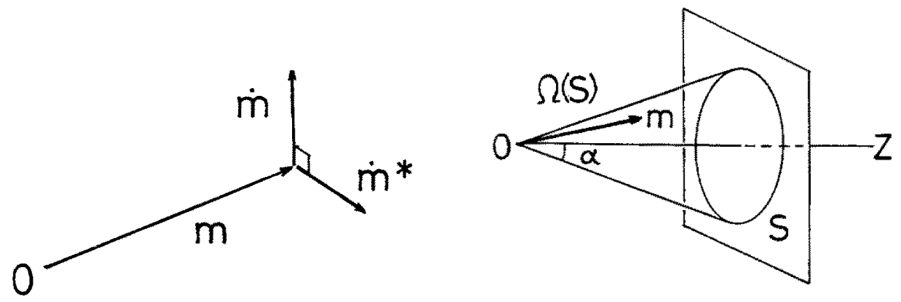


Abbildung 2.8: *Image Sphere Representation.*

zunächst notwendig die Bewegungsparameter $\{\boldsymbol{\omega}, \mathbf{v}\}$ zu bestimmen. Hierzu ist es förderlich das betrachtete Flussfeld $\dot{\mathbf{m}}(\mathbf{m})$ zu rotieren. Genauer: Jeder *N-velocity*-Vektor $\dot{\mathbf{m}}$ wird um 90° seines zugehörigen $\mathbf{m}$ gedreht. Die aus dieser Rotation resultierenden Vektoren werden mit $\dot{\mathbf{m}}^* = \mathbf{m} \times \dot{\mathbf{m}}$ bezeichnet. Die Idee ist in Abbildung 2.9a visualisiert. Für den gegebenen gedrehten optischen Fluss (engl. *Twisted Flow*) $\dot{\mathbf{m}}^*(\mathbf{m})$ werden ferner die Tensoren $\mathfrak{L} = (L_{ij})$, $\mathfrak{M} = (M_{ijk})$ und $\mathfrak{N} = (N_{ijkl})$ definiert:

$$\begin{aligned} L_{ij} &= \int_{\Omega(S)} \dot{\mathbf{m}}_i^* \dot{\mathbf{m}}_j^* \, d\Omega(\mathbf{m}) \\ M_{ijk} &= \int_{\Omega(S)} \dot{\mathbf{m}}_i^* \mathbf{m}_j \mathbf{m}_k \, d\Omega(\mathbf{m}) \\ N_{ijkl} &= \int_{\Omega(S)} \mathbf{m}_i \mathbf{m}_j \mathbf{m}_k \mathbf{m}_l \, d\Omega(\mathbf{m}) \end{aligned} \tag{2.19}$$

[7]Kamerabewegungen entlang der Sichtlinie (engl. *Line of Sight*) verursachen keine Bildbewegung.



(a) **(b)**

Abbildung 2.9: (a) Die Rotation aller *N-velocity*-Vektoren führt zum *Twisted Flow*. **(b)** Der feste Winkel $\Omega(S)$ und der damit korrespondierende Sehbereich S definieren das Flussfeld $\dot{\mathbf{m}}(\mathbf{m})$.

Die Integration erfolgt jeweils über alle möglichen Winkel, welche eingenommen werden können, wenn ein bestimmter Sehbereich S (engl. *Field of View*) betrachtet wird (siehe Abbildung 2.9b). Während die Tensoren $\mathfrak{L}$ und $\mathfrak{M}$ mit Hilfe des beobachteten optischen Fluss berechnet werden, erfolgt die Bestimmung des Tensors $\mathfrak{N}$ ausschließlich aufgrund der Geometrie des Sehbereichs S.

Die Bestimmung der Rotationsgeschwindigkeit und der Translationsgeschwindigkeit der Kamerabewegung erfolgt mit Hilfe der folgenden dargestellten Gleichungen. Die Translationsgeschwindigkeit $\mathbf{v} = (\mathbf{v}_i)$ ist definiert als der Einheitseigenvektor der Matrix $A = (A_{ij})$ mit kleinstem Eigenwert, wobei die Matrix folgende Form hat:

$$A_{ij} = L_{ij} - \sum_{k,l,m,n=1}^{3} M_{ikl}\, N^{-1}_{klmn}\, M_{jmn} \tag{2.20}$$

Die Rotationsgeschwindigkeit $\boldsymbol{\omega}$ ist durch

$$\boldsymbol{\omega} = \frac{1}{2}[tr(K) + 3(\mathbf{v}, K\mathbf{v})]\mathbf{v} - 2K\mathbf{v} \tag{2.21}$$

determiniert, wobei die Matrix $K = (K_{ij})$ wie folgt bestimmt wird:

$$K = (K_{ij}) = -\sum_{k,l,m=1}^{3} N^{-1}_{ijkl}\, M_{mkl}\, \mathbf{v}_m \tag{2.22}$$

Mit Hilfe der errechneten Bewegungsparameter $\{\boldsymbol{\omega}, \mathbf{v}\}$ kann die Tiefe $r(\mathbf{m})$ mit

$$r(\mathbf{m}) = \frac{1 - (\mathbf{m}, \mathbf{v})^2}{|\mathbf{m}, \boldsymbol{\omega}, \mathbf{v}| - (\mathbf{m}, \mathbf{v})} \tag{2.23}$$

ermittelt werden. Diese Gleichung wird als sogenannte Bewegungsparallaxegleichung (engl. *Motion Parallax Equation*) bezeichnet, wobei $|\mathbf{m}, \boldsymbol{\omega}, \mathbf{v}|$ das Spatprodukt beschreibt. Der Klammerausdruck $(\cdot, \cdot)$ wird als Schreibweise für das Skalarprodukt zweier Vektoren verwendet. Nach [16] wird das Problem der Eigenbewegungsbestimmung mathematisch wie folgt formuliert

$$(\dot{\mathbf{m}}^*, \mathbf{v}) + (\mathbf{m}, K\mathbf{m}) = 0 \tag{2.24}$$

und beschreibt die durch *Essential Parameters* $\{K, \mathbf{v}\}$ und *Twisted Flow* $\dot{\mathbf{m}}^*$ formulierte Epipolargleichung. Die Lösung der Gleichung ist über die Methode der kleinsten Quadrate möglich. Eine eventuell auftretende systematische Verzerrung (engl. *bias*) der Lösung wird dabei durch die propagierte und in den folgenden Schritten zusammengefasste Renormierungsmethode (engl. *Renormalization*) ausgeglichen.

1. Berechne die Tensoren $\mathfrak{k} = (L_{ij})$, $\mathfrak{M} = (M_{ijk})$, $\mathfrak{N} = (N_{ijkl})$ und $\mathfrak{N}^{-1} = (N^{-1}_{ijkl})$. Berechne zusätzlich die Momentmatrix $M = (M_{ij})$ wie folgt:

$$M = \int_{\Omega(S)} \mathbf{m}\mathbf{m}^T \, d\Omega(\mathbf{m}) \quad .$$

2. Berechne Matrix $A = (A_{ij})$ nach Gleichung (2.20) und Matrix $B = (B_{ij})$ wie folgt[8]:

$$B = \Omega(S)\delta_{ij} - M_{ij} \quad .$$

3. Bestimme den durch $A\mathbf{v} = \mathbf{c}B\mathbf{v}$ determinierten Translationsvektor $\mathbf{v}$ ($\mathbf{c}$ ist der dem kleinsten Eigenwert zugehörige, auf 1 normierte Eigenvektor dieses verallgemeinerten Eigenwertproblems).

4. Berechne Rotationsvektor $\boldsymbol{\omega}$ und Matrix K mit Hilfe der Gleichungen (2.21) und (2.22).

[8] δ_{ij} = Kronecker-Delta

Kapitel 3

Design und Implementierung

3.1 Motivation

Ziel dieser Arbeit war es, herauszufinden, ob es möglich ist die Eigengeschwindigkeit einer Flugdrohne bestimmt werden kann, wenn die Informationen dazu aus dem optischen Fluss von Kamerabildern stammen. Es stellte sich nun die Frage, unter welchen Annahmen dies möglich ist. Ein wesentlicher Faktor ist die Vieldeutigkeit des optischen Flusses. Es zeigt sich, dass die Wiedergewinnung dreidimensionaler Information mit Hilfe eines kalibrierten Kamerasystems kein eindeutiges Ergebnis produziert. Ohne externe Information zur Szenerie kann eine Rekonstruktion nur bis zu einer Ähnlichkeitstransformation bestimmt werden (siehe Abbildung 3.1). Es ist beispielsweise nicht möglich, die absolute Position und Orientierung der in Abbildung 2.5 dargestellten Figur[1] festzustellen. Auch die Frage ob die Höhe der Figur mehrere Meter oder nur wenige Zentimeter beträgt, ist nicht bestimmbar. Einzig die TTC (engl. *Time-to-Collision*) lässt sich ohne jegliche äußere Informationen der abgebildeten Welt bestimmen. Sie beschreibt anhand der optischen Expansionsrate wie schnell sich ein Objekt nähert und wann es folglich zu einen Zusammenstoß mit dem Betrachter kommen wird [1][2].

3.1.1 Geschwindigkeitsschätzung

Das Problem der Skalierung wirkt sich auf die Bestimmung der Translationsgeschwindigkeit der Kamera dahingehend aus, dass nur relative Geschwindigkeitsänderungen bezüglich einer normierten Anfangsgeschwindigkeit berechnet werden können. Eine Veranschaulichung erfolgt in Abbildung 3.2 und zeigt eine mögliche Trajektorie der Flugdrohne relativ zu einer Oberfläche. Zur Berechnung der absoluten Geschwindigkeit zu jeden beliebigen Zeitpunkt muss

[1] http://www.flaaronning.no/troll/

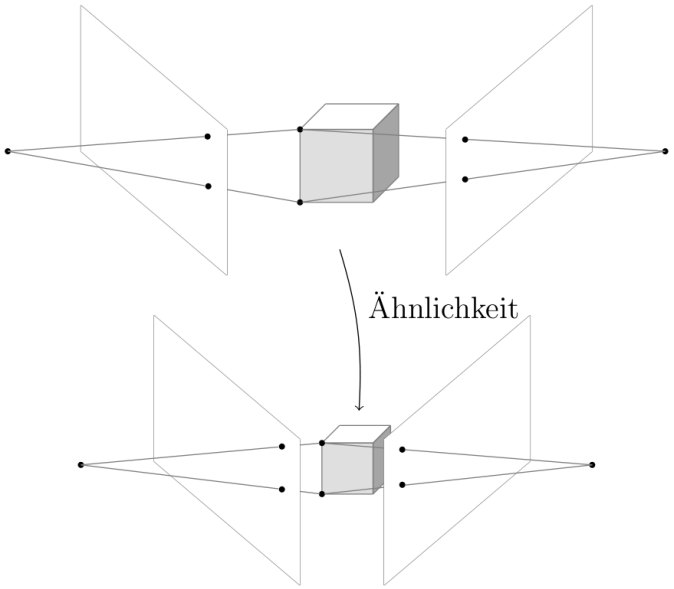


Abbildung 3.1: Die Rekonstruktion dreidimensionaler Informationen mit einen kalibrierten Kamerasystem ist nur bis zu einer Ähnlichkeitstransformation möglich. Zwei unterschiedlich große kubische Objekte erzeugen identische Bildpunktkorrespondenzen.

daher eine Startgeschwindigkeit mit Hilfe anderer Sensoren bestimmt werden (zum Beispiel mit Hilfe von GPS-Koordinaten). Es sollen nun charakteristische Merkmale im Sichtfeld der Kamera verfolgt werden und anhand von diesen eine Eigengeschwindigkeitsschätzung erfolgen. Wie bereits angesprochen ist es nicht möglich zu unterscheiden, ob die projizierte Bewegung des verfolgten *Feature* f_1 auf der Bildebene durch ein entferntes Objekt und eine große Kamerabewegung oder durch ein nahes Objekt und eine kleine Kamerabewegung generiert wird. Diese Tatsache führt dazu, dass mathematische Größen, wie zum Beispiel die Tiefenwerte d_i oder die Geschwindigkeiten v_i, mit einem Skalierungsfaktor k_i behaftet sind[2]. Des Weiteren sind nicht alle Flussvektoren für eine Geschwindigkeitsschätzung geeignet: Als ungeeignet zählen Flussvektoren, welche durch sich unabhängig bewegende Objekte im Blickfeld entstehen (engl. *Independent Motion Vector*[3]). Sie müssen mit Hilfe der in Abschnitt 2.3.2 vorgestellten Eigenbewegungsschätzung nach Kanatani herausgefiltert werden. Die Bildpunkte der übriggebliebenen und somit der Eigenbewegung zugeordneten Flussvektoren[4] werden nun über eine Zeitspanne von drei Frames verfolgt. Die relative Geschwindigkeitsänderung kann im Weiteren über den Quotienten $\frac{d_i}{d_{i+1}}$ berechnet werden, wobei die Bestimmung der Tiefenwerte durch Gleichung

[2]Der Index i stellt, soweit nicht explizit anders angegeben, einen Bezug zum jeweiligen *Frame* der Filmsequenz her und ist im Kontext als Flussberechnung zwischen *Frame* i und *Frame* $(i+1)$ zu verstehen.

[3]Abk. IDMV

[4]Abk. nonIDMV oder nonIDM-Vektor

(2.23) ermöglicht wird. Für die Gültigkeit dieser Annahme ist es unerläßlich, dass beide Tiefenwerte über dieselbe Kameraposition berechnet werden. Ein Quotient größer Eins beschreibt hierbei eine positive Beschleunigung und ein Quotient kleiner Eins demnach eine negative Beschleunigung. Es ist keine Geschwindigkeitsänderung erfolgt, wenn der Quotient gleich Eins ist.

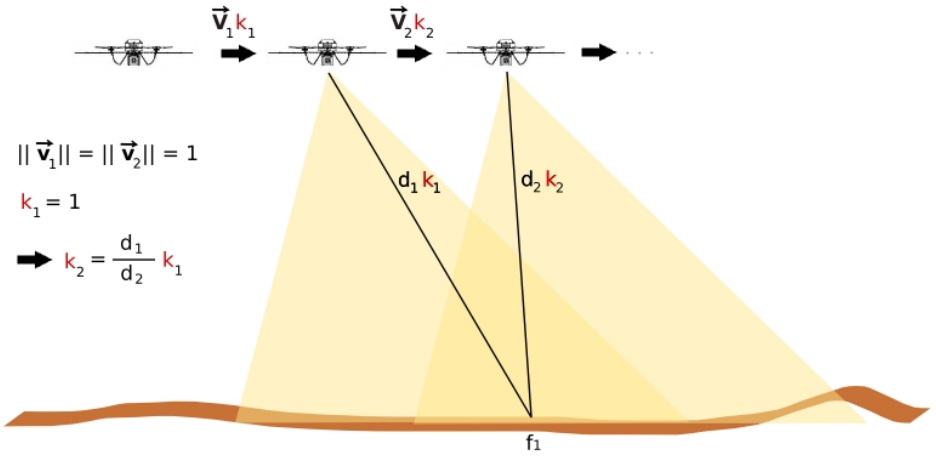


Abbildung 3.2: Mögliche Trajektorie der Flugdrohne relativ zu einer Oberfläche. Unter bestimmten Annahmen kann eine relative Eigengeschwindigkeitsschätzung erfolgen.

3.1.2 Verfolgung eines Bildpunktes über mehrere Bilder

Zur Berechnung der relativen Geschwindigkeitsänderung ist es notwendig, charakteristische Bildpunkte über drei Frames zu verfolgen. Dabei zeigt sich zunächst - mit Bezug auf die Ausführungen in Abschnitt 2.2.3 -, dass die Menge der *Feature*, welche für die Flussberechnung von *Frame* i nach *Frame* $(i+1)$ verwendet werden, nicht identisch sein muss mit der Menge der *Feature* für die Berechnung von $(i+1) \rightarrow (i+2)$. Für eine erfolgreiche Beschleunigungsschätzung muss sichergestellt werden, das ein *Feature* über drei aufeinanderfolgende *Frames* identifiziert werden kann (Tripelkorrespondenz). Um genau dies sicherzustellen beinhaltet die nun im Folgenden vorgestellte Methode eine zusätzliche Flussberechnung sowie drei ϵ-Umgebungen. Die Einzelschritte der Bildpunktverfolgung lauten:

1. Bestimme nonIDMV von $i \rightarrow (i+1)$.
2. Bestimme nonIDMV von $(i+1) \rightarrow (i+2)$.
3. Bestimme nonIDMV von $i \rightarrow (i+2)$
4. In einer Subpixelumgebung ϵ_{i+1} vergleiche Endpunkte der nonIDMV aus Schritt 1 mit Anfangspunkte der nonIDMV aus Schritt 2.

5. In einer Subpixelumgebung ϵ_{i+2} vergleiche Endpunkte der nonIDMV aus Schritt 2 mit Endpunkte der nonIDMV aus Schritt 3.

6. In einer Subpixelumgebung ϵ_i vergleiche Anfangspunkte der nonIDMV aus Schritt 3 mit Anfangspunkte der nonIDMV aus Schritt 1.

Die Subpixelumgebungen sind in allen drei Fällen gleich groß gewählt (implementiert wurde ein Radius von 0.5). Die grundlegende Idee ist nocheinmal in Abbildung 3.3 veranschaulicht und lässt erkennen, dass für die geforderte Tripelkorrespondenz getestet wird, ob Vektorpaare existieren, deren Summation einen der Vektoren aus Schritt 3 ergeben. Die Tiefenwerte der erfolgreich vermittelten Vektorpaare dienen im Weiteren der Geschwindigkeitsschätzung. Da in dieser Methode der optische Fluss stets vorwärts entlang der Zeitachse berechnet wurde, ist es zuvor jedoch nötig, die Tiefenwerte der nonIDMV aus Schritt 1 von der Kameraposition C_i nach C_{i+1} zu transformieren. Die Korrektur ist durch die Gleichung

$$d_{neu} = |(d_{alt}\,\mathbf{m}) - \mathbf{v}| \tag{3.1}$$

zu beschreiben. Die Definitionen der beiden Vektoren $\mathbf{m}$ und $\mathbf{v}$ sind Abschnitt 2.3.2 beschrieben.

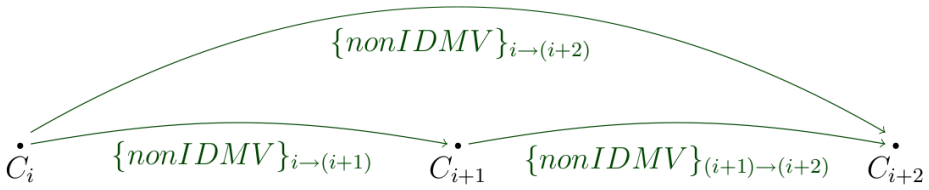


Abbildung 3.3: Bildpunktverfolgung über mehrere Frames. Die zusätzliche Flussberechnung von Frame i nach Frame $(i + 2)$ stellt die Tripelkorrespondenz sicher.

3.2 Die Implementierung im Detail

3.2.1 Parameter des GDIM

Der in dieser Diplomarbeit verwendete GDIM-Algorithmus (es konnte auf eine bereits implementierte Version gemäß [26] zurückgegriffen werden) besitzt eine Reihe von Parametern, welche einen Einfluss auf die Güte der Flussberechnung haben. Da die Präzision der Eigengeschwindigkeitsschätzung abhängig ist von der Genauigkeit der Eigenbewegungsschätzung nach Kanatani, welche wiederum abhängig ist von der Korrektheit der berechneten Flussvektoren nach

GDIM, soll nun auf drei Parameter eingegangen werden, deren Voreinstellungen überschrieben wurden. Dies geschieht mit dem Ziel, die Ausgangsmenge der repräsentativen Flussvektoren zu maximieren. Zu jeden Parameter ist die Deklaration der C-Funktion zum überschreiben mit angegeben.

Parameter basierend auf den minimalen Eigenwert

```
void set_minEig(float eig);
```

Dieser Parameter bietet eine einfache Möglichkeit der qualitativen Filterung der Flussvektoren. Als Filterkriterium dienen hierbei die Eigenwerte der Matrix die aus den Grauwerten der Pixelumgebung besteht. Diese Eigenwerte sind ein Maß für die Strukturiertheit der Pixelumgebung. Ist diese zu gering, kann in der Regel kein zuverlässiger Fluss berechnet werden. Eine Klassifizierung in „gute" und „schlechte" Flussvektoren erfolgt über den Vergleich des Eigenwertes mit einem festgelegten Schwellenwert. Dies ist der *minEig*-Parameter. Der Vorgabewert für den minimalen Eigenwert ist 1.0. Es ist anzumerken, dass die Wahl des Parameters einen direkten Einfluss auf die Gesamtmenge der berechenbaren Flüsse hat und sich nicht lediglich das Größenverhältnis der beiden Klassen zueinander ändert. Große Werte führen demnach zu einer steigenden Anforderung an die Strukturiertheit, was in der Regel dazu führt, dass das Flussfeld zunehmend „löchriger" wird. Im schlimmsten Fall stehen der Eigenbewegungsberechnung nur noch Flussvektoren aus einem kleinen Teilbereich des Bildes zur Verfügung[5]. In kombinierter Verwendung mit der in Abschnitt 3.2.2 erläuterten flussvektorgestützten Filterung zeigt sich, dass der Schwellenwert sogar verringert werden kann und dies die Schätzung der Eigenbewegungsparameter positiv beeinflusst (siehe Anhang B.1). Für diese Arbeit wurde der *minEig*-Parameter auf 0.1 gesetzt, was einen guten Kompromiss zwischen der Anzahl der berechenbaren Vektoren und deren Güte darstellt

Veränderung der Rekursionstiefe

```
void set_pyrLvl(int lvl);
```

Durch Aufruf der entsprechenden C-Funktion ist es möglich, den Vorgabewert der Rekursionstiefe der pyramidalen Bildrepräsentation zu ändern (siehe Abschnitt 2.2.2). Für die Eigengeschwindigkeitschätzung wurde die Rekursionstiefe von vier auf fünf erhöht. Dies entspricht der maximal zulässigen Rekursionstiefe überhaupt. Die Auswirkung dieser Anpassung kann in Abbildung 3.4 nachvollzogen werden. Beide Flussbilder wurden aus dem gleichen Bildpaar

[5]Für eine aussagekräftige Schätzung ist eine gleichmäßige Verteilung der Flussvektoren über das gesamte Blickfeld von Vorteil. Eine Überrepräsentation bestimmter Bildabschnitte sollte daher vermieden werden.

berechnet und stammen aus der *Corridor-Map* (siehe Abschnitt 3.3). Aufgrund der bewusst sehr hoch gewählten Geschwindigkeit[6] und der Enge des Korridors sind die Pixelverschiebungen im linken und rechten Randbereich besonders groß, so dass bei zu gering gewählten *pyrLvl*-Parameter keine sinnvolle Flussberechnung mehr erfolgen kann (siehe Verwirbelungen in Abbildung 3.4a).

Veränderung der Auflösung

```
void set_bins( int x, int y, max);
```

Die Höchstzahl der zu berechnenden Flussvektoren erfolgt über diesen Parameter. Hierzu wird das Bild in Kästen (engl. *bins*) aufgeteilt. Die Auflösung dieses erzeugten Rasters ist abhängig von der Wahl von x und y, wobei x die Anzahl der *bins* in x-Richtung beschreibt und y analog für die y-Richtung des Bildes zuständig ist. Der *max*-Wert sorgt für eine gleichmäßige Verteilung der Flussvektoren über das komplette Bild und gibt die maximale Anzahl an Flussvektoren pro Kasten an. Statt dem Standardwert von $(8, 6, 3)$ wurden hier die Werte $(80, 60, 1)$ benutzt. Daraus ergibt sich eine maximale Anzahl von zu berechnenden Flussvektoren 4.800.

3.2.2 Flussvektorgestützte Filterung

```
void set_flowfilter(float AngDev, float PixGap);
```

Ausgehend von den Ausführungen im vorherigen Abschnitt bezüglich des *minEig*-Parameters wurde eine Funktion implementiert, welche die qualitative Filterung der Flussvektoren verfeinert. Die Klassifizierung erfolgt nun nicht mehr ausschließlich auf Grundlage der Strukturiertheit der Pixelumgebung, sondern zusätzlich durch Informationen, welche durch die bereits im GDIM implementierte *runback*-Funktion bereitgestellt werden. Durch den Aufruf dieser Funktion wird - ausgehend vom Endpunkt des jeweils betrachteten Flussvektors - der optische Fluss rückwärts berechnet.

Dieser zu jedem Flussvektor zugehörige rückwärtsgeschätzte Vektor dient als Grundlage für die im Weiteren erklärte Filterung. Ein Flussvektor $\mathbf{v}$ wird im Sinne der flussvektorgestützten Filterung als repräsentativ bezeichnet, wenn die korrespondierende Rückprojektion $\mathbf{v}_b$ innerhalb eines Toleranzbereichs das Inverse des Flussvektors $\mathbf{v}$ ist. Dieser Toleranzbereich wird über zwei Paramater modelliert. Während *AngDev* die maximal erlaubte Winkelabweichung beschreibt, schildert *PixGap* den maximal erlaubten Pixelabstand zwischen Anfangspunkt von $\mathbf{v}$ und Endpunkt von $\mathbf{v}_b$.

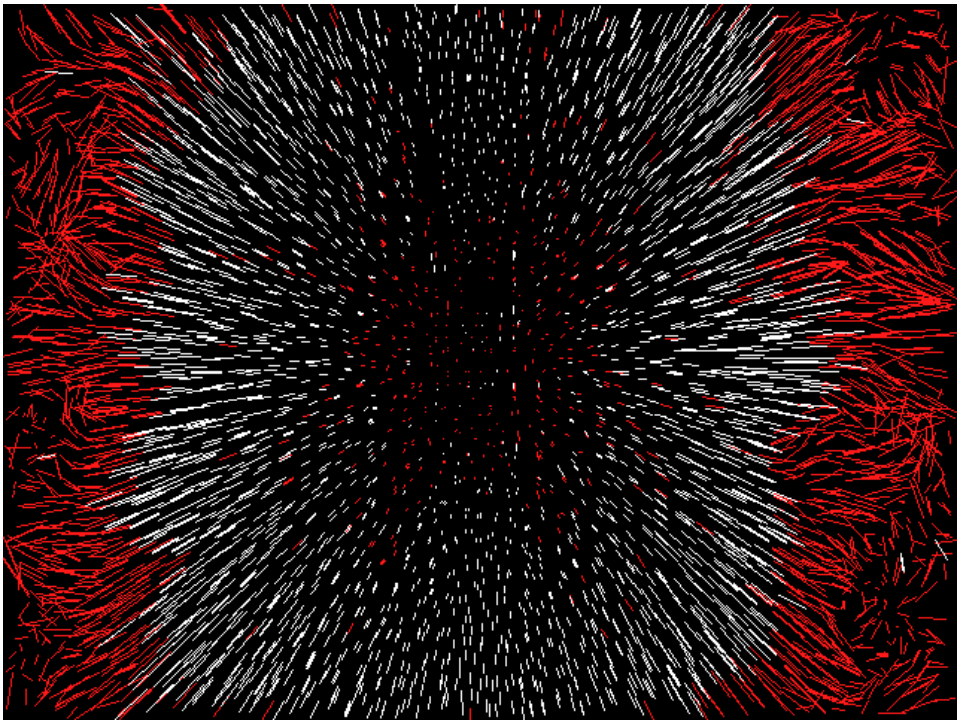
[6] Zur Veranschaulichung dieser Besonderheit wurde der optische Fluss zwischen *Frame i* und *Frame (i+3)* berechnet

Um die Funktionsweise zu demonstrieren, wurde in Abbildung 3.4 der optische Fluss für zwei unterschiedliche Rekursionstiefen berechnet. Wie bereits erwähnt, kommt es aufgrund der hohen Pixelverschiebungen im linken und rechten Randbereich der Abbildung 3.4a zu Verwirbelungen. Die Flussvektoren in diesen Bereichen des Bildes sind unbrauchbar und werden mit Hilfe der flussvektorgestützten Filterung erfolgreich erkannt (rot eingefärbte Vektoren) und aussortiert. Für eine allumfassend verbesserte Vorfilterung der Flussvektoren wurde die Winkelabweichung auf 3° und der maximale Pixelabstand auf 0.1 eingestellt (siehe Anhang B.2).

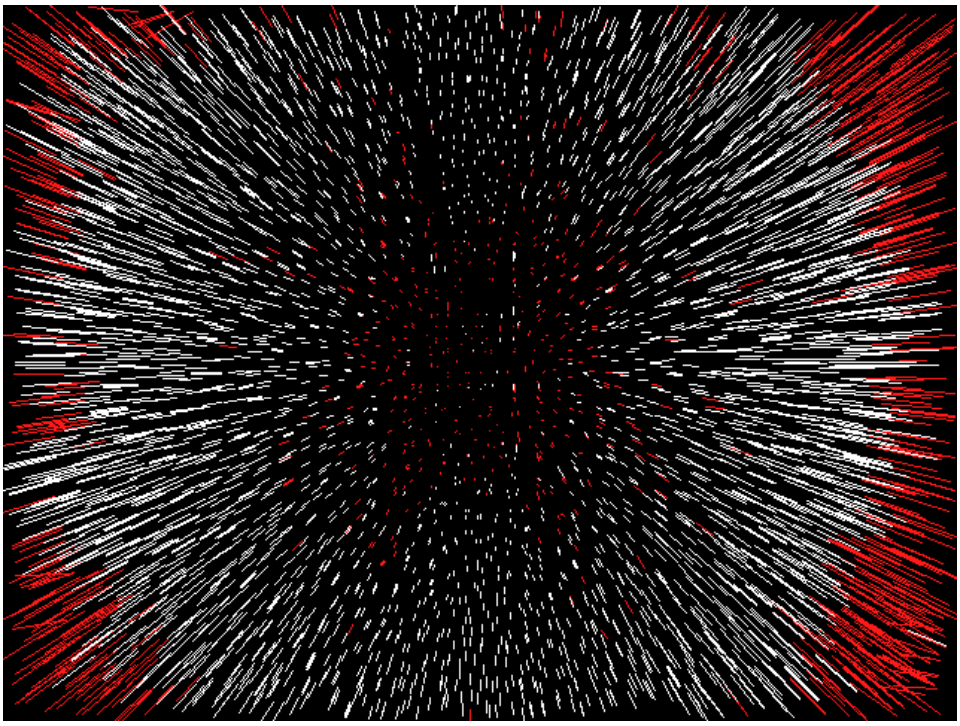
3.2.3 Robuste Schätzung der Eigenbewegung

Zur Eigengeschwindigkeitsberechnung sind Tiefenschätzungen von nonIDM-Vektoren nötig. Dazu wird der durch die bisherigen Maßnahmen vorgefilterte optische Fluss nun an den Algorithmus von Kanatani (siehe Abschnitt 2.3.2) übergeben. Für eine robuste Eigenbewegungsschätzung ist der Kanatani in den sogenannten RANSAC (engl. *Random Sample Consensus*) Algorithmus gemäß [8] eingebettet. Es handelt sich hierbei um ein Verfahren, welches aus einer gegebenen Datenmenge (Flussvektoren) die Parameter eines Modells (Eigenbewegungsparameter) optimal zu schätzen versucht. Die nach [8] zugeschriebene hohe Zuverlässigkeit des RANSAC spiegelt sich darin wieder, dass eine sinnvolle Schätzung der Modellparameter, welche die gegebenen Daten erklären, auch dann noch erfolgt, wenn die Messwerte eine signfikante Anzahl an Ausreißern (engl. *Outlier*) enthält. Für eine erfolgreiche Detektion und Eliminierung der Ausreißer betrachtet RANSAC in n Iterationen jeweils eine zufällig gewählte Teilmenge der gegebenen Datenmenge und schätzt basierend darauf ein Modell. Die Größe dieser Teilmenge ist dabei auf die minimale Anzahl benötigter Messwerte, welche zur Bestimmung der Modellparameter nötig sind, beschränkt. Unter Beachtung eines vorher festzulegenden Toleranzbereiches wird anschliessend die Güte des aktuell betrachteten Modells getestet. Datenpunkte aus der Gesamtmenge welche dem Modell innerhalb des Toleranzbereichs genügen, werden in die sogenannte Konsensusmenge (engl. *Consensus Set*) aufgenommen. Die finalen Modellparameter werden zum Schluss aus der größten der n Konsensusmengen berechnet.

In dieser Arbeit konnte auf eine Implementierung gemäß [26] zurückgegriffen werden. Diese besitzt - speziell den RANSAC betreffend - zwei Modifizierungen. Beide werden im Folgenden kurz beschrieben. Die erste Anpassung betrifft die zufällige Auswahl der zur Modellinitialisierung benötigten Messwerte aus der Gesamtmenge. Für eine eindeutige Bewegungsschätzung ist der Kanatani auf mindestens acht Flussvektoren angewiesen, welche idealerweise über das komplette Blickfeld verteilt sein sollten. Um dies zu gewährleisten, wird der Bildbereich in vier Abschnitte aufgeteilt. In jedem Quadranten werden nun zufällig zwei Flussvektoren ausgewählt. Sollte einer der Quadranten



(a)



(b)

Abbildung 3.4: Flussvektorgestützte Filterung. Die rot eingefärbten Flüsse werden nicht weiterverwendet. **(a)** *pyrLvl*-Parameter gleich 4. **(b)** *pyrLvl*-Parameter gleich 5.

keinen Flussvektor beinhalten, so erfolgt eine zufällige Auswahl aus dem gesamten Bildbereich. Die zweite Anpassung betrifft den Toleranzbereich. Um die Güte eines Modells zu beschreiben, wird normalerweise ein vorher festgelegter Schwellenwert benötigt. Stattdessen wird an dieser Stelle jedoch für jedes Modell eine sogenannte Fehlermodellkurve bestimmt, welche zum Vergleich der Modelle bezüglich ihrer Güte herangezogen wird. Innerhalb von 250 Iterationen wird damit das „beste" Modell ermittelt. Für dieses Modell wird, ausgehend von seinen mittleren Fehlerwert, mit Hilfe des Kittler-Verfahrens ein Schwellenwert dynamisch bestimmt [18]. Flussvektoren aus der Konsensusmenge dieses Modells unterschreiten diesen Schwellenwert und dienen dem Kanatani zur endgültigen Eigenbewegungsschätzung.

3.2.4 Expansionspunktgestützte Filterung

```
void set_FOEproximity(float radius);
```

Bei dem Expansionspunkt[7] (engl. *Focus of Expansion*) handelt es sich um einen Punkt in der Bildebene. Genauer: Es ist ein gemeinsam entstehender Schnittpunkt aller verlängerten nonIDMV. Eine wichtige Eigenschaft des FOE ist, dass der optische Fluss mit abnehmender Entfernung zum FOE gegen Null strebt und letztendlich direkt im FOE seinen Ruhepol hat. Hier ist der optische Fluss gleich Null. Vektoren in der Nähe des FOE haben einen stark schwankenden Tiefenwert. Sie sind daher ungeeignet für eine Eigengeschwindigkeitsbestimmung der in dieser Arbeit lancierten Methode. Ausgehend von diesen Überlegungen wurde eine weitere Filtermethode implementiert, welche Flussvektoren in einem vorgegebenen Radius um den FOE herum verwirft. Diese Methode kann nur angewandt werden kann, wenn ein FOE existiert und bestimmt werden kann. In diesen Zusammenhang wird auf die kommenden Seitwärtsflug-Testsequenzen hingewiesen, in welchen die Kamera im rechten Winkel zur Flugrichtung ausgerichtet wurde. Für die Diplomarbeit wurde ein Radius von 100 gewählt. Eine Darstellung dieser Filtermethode erfolgt in Abbildung 3.5.

3.2.5 Tripelkorrespondenz-Filter und Schätzer

```
void set_tripleProximity(float epsilon);
```

Flussvektoren, welche alle vorangestellten Maßnahmen zur Qualitätssicherung bestanden haben, werden abschließend mit Hilfe der implementierten Methode der Tripelkorrespondenz gefiltert. Eine Erklärung zur Motivation und Funktionsweise dieser Methode wurde bereits in Abschnitt 3.1.1 gegeben. Das Ergeb-

[7] Abk. FOE

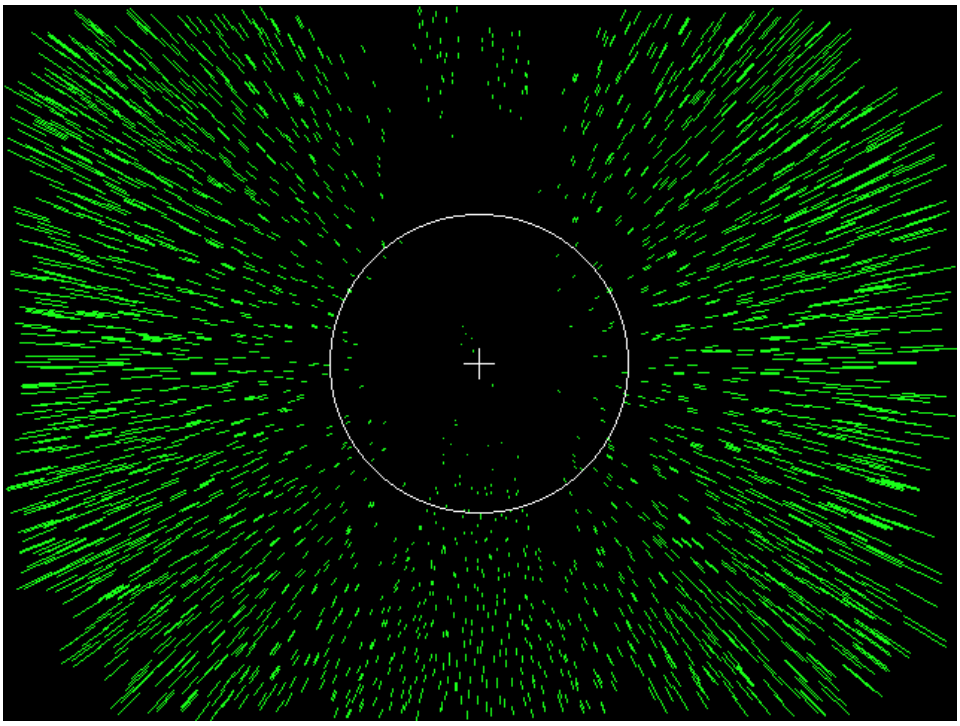


Abbildung 3.5: Nach erfolgreicher Berechnung der Eigenbewegungsparameter (grüne Vektoren entsprechen nonIDMV), kann der Expansionspunkt (weißes Kreuz) bestimmt werden. Vektoren, welche sich innerhalb des dargestellten Kreises befinden, werden verworfen.

nis ist in Abbildung 3.6 präsentiert. Dargestellt sind jene nonIDM-Vektoren welche von *Frame* i nach $(i+1)$ und von *Frame* $(i+1)$ nach $(i+2)$ berechnet wurden. Jedes auf diese Art und Weise erhaltene Vektorpaar entspricht einem Messwert (Quotient der Tiefenwerte) der für die Eigengeschwindigkeitsschätzung benutzt wird. Als Schätzer der tatsächlichen Geschwindigkeit wird der Median der Menge von Geschwindigkeitsmessungen gewählt, da dieser robust gegen Ausreißer ist.

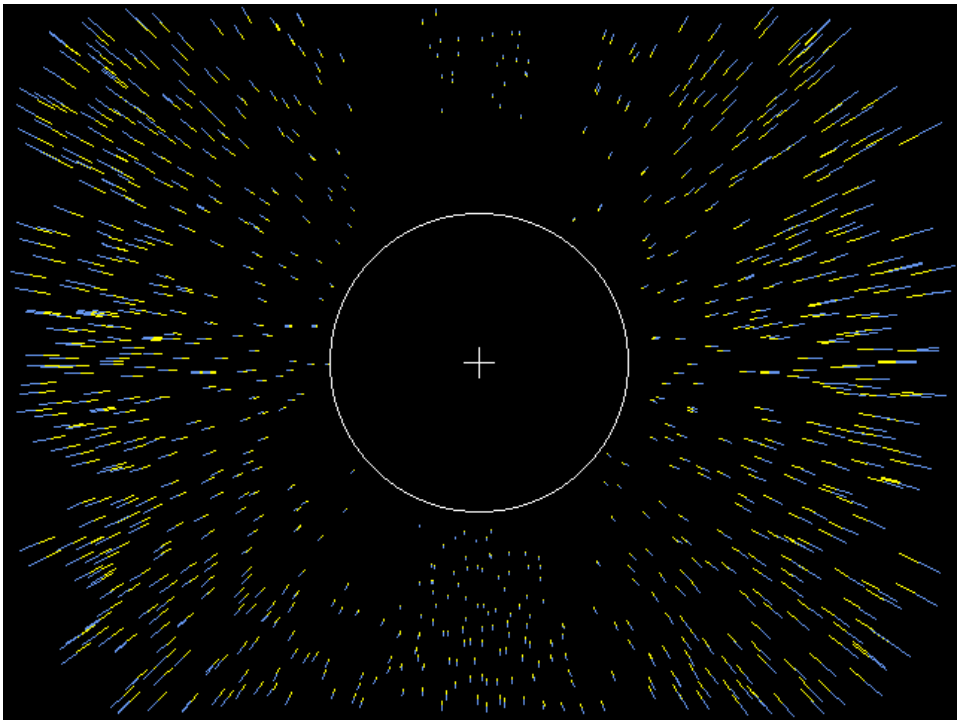


Abbildung 3.6: Filterung nonIDMV entsprechend der Tripelkorrespondenz. Darstellt sind $nonIDMV_{i\rightarrow(i+1)}$ (gelb eingefärbt) und $nonIDMV_{(i+1)\rightarrow(i+2)}$ (blau eingefärbt). Jedes Vektorpaar (gelb/blau) stellt einen Messwert für die Eigengeschwindigkeitsschätzung zur Verfügung.

3.3 Testsequenzen

Um die Leistung und Genauigkeit zu testen, wurden virtuelle und reale Testsequenzen benutzt. Auf diese soll nun kurz eingegangen werden.

3.3.1 Virtuelle Testsequenzen

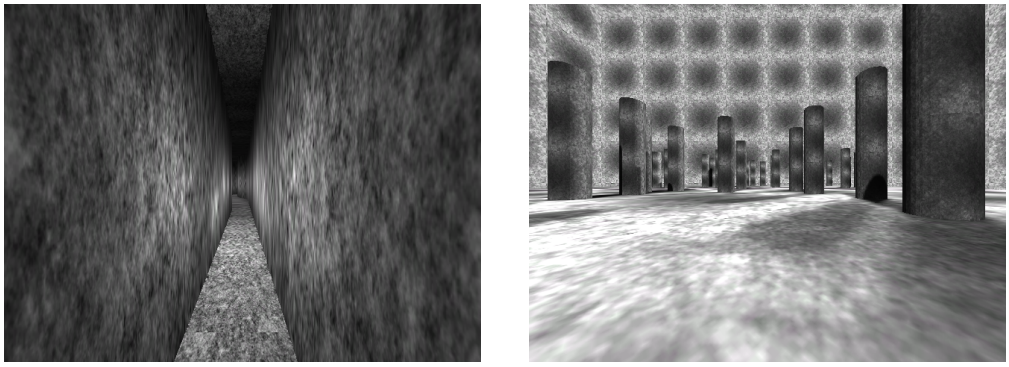
Es konnte hier auf eine bereits funktionstüchtige virtuelle Testumgebung von [26] zurückgegriffen werden, welche mit Hilfe von *Crystal Space*[8] erstellt wurde. Die Motivation zur Benutzung einer virtuellen Umgebung ergab sich aus der Möglichkeit die Rotationsmatrix und den Translationsvektor für jedes Frame frei zu manipulieren. In Kombination mit der optimal texturierten und beleuchteten Umgebung, sowie der Simulation eines idealen verzerrungsfreien Kamerasystems, ist eine bestmögliche Flussberechnung und Eigenbewegungsschätzung gewährleistet. Zur Datengenerierung wurden zwei unterschiedliche Karten (engl. *Maps*) benutzt. Während die *Corridor-Map* eine enge abgegrenzte Umgebung (engl. *Closed Quarter*) in Form eines Korridors darstellt, bietet die *Columns-Map* eine weitläufige Fläche (engl. *Open Space*) mit eingestreuten säulenförmigen Objekten. Einen Eindruck über das Aussehen beider Karten erhält man in Abbildung 3.7. Folgende Testsequenzen wurden generiert:

- Translationsflug (entlang einer Achse)
  - Vorwärtsflug
    - keine Beschleunigung
    - positive Beschleunigung in 3 Stufen
    - negative Beschleunigung in 3 Stufen
  - Rückwärtsflug
    - keine Beschleunigung
    - positive Beschleunigung in 3 Stufen
    - negative Beschleunigung in 3 Stufen
  - Seitwärtsflug (Blickrichtung 90° zur Flugrichtung)
    - keine Beschleunigung

Für jeden der Translationsflüge wurde nun noch eine Variante mit Nickbewegung[9] (engl. *Pitch*) der virtuellen Drohne untersucht. Für die Beschleunigungssequenzen (positiv als auch negativ) in drei Stufen wurde die Bewegung ab *Frame* 25, 50 und 75 jeweils verdoppelt bzw. halbiert. Für den Seitwärtsflug

[8] www.crystalspace3d.org
[9] Rotationsbewegung um die Querachse

bei konstanter Geschwindigkeit und den Vorwärtsflug bei konstanter Geschwindigkeit wurden zudem Testsequenzen mit unterschiedlichem Sichtfeld[10] (engl. *Field of View*) generiert. Alle Aufnahmen hatten eine Auflösung von 640 × 480 Bildpunkten. Für die Berechnung einer Geschwindigkeitskurve wurde eine Bildwiederholungsrate[11] (engl. *Frames per Second*) von 21 FPS gewählt. Dieser Wert ist biologisch motiviert und stellt einen Schwellenwert da. Ab diesen verschwimmen Einzelbilder auf der menschlichen Netzhaut und sind nicht mehr als solche erkennbar.



(a) **(b)**

Abbildung 3.7: Es wurden zwei verschiedene virtuelle Umgebungen für die Generierung von Sequenzen benutzt. **(a)** *Corridor-Map.* **(b)** *Columns-Map.*

3.3.2 Reale Testsequenzen

Die erzeugten Testsequenzen lassen sich in zwei Kategorien unterteilen: strikter Translationsflug und Freiflug. Erstere wurden mit einer Webcam *QuickCam Pro 9000* von Logitech aufgenommen. Diese besitzt einen horizontalen Öffnungswinkel von 41° und einen vertikalen Öffnungswinkel von 33°. Ein Foto des Versuchsaufbaus ist in Abbildung 3.8 zu sehen. Für die Aufnahmen wurde die Webcam auf einer Führungsschiene befestigt und an dieser entlang verschoben. An fest definierten Punkten der Schiene erfolgten dann Einzelbildaufnahmen in einer Auflösung von 960 × 720, welche zusammengesetzt die jeweilige Testsequenz ergaben. Folgende Testsequenzen wurden generiert:

- Translationsflug (entlang einer Achse)
  - Vorwärtsflug
    - keine Beschleunigung
    - positive Beschleunigung in 2 Stufen

[10] Abk. FOV
[11] Abk. FPS

        * negative Beschleunigung in 2 Stufen
    - Rückwärtsflug
        * keine Beschleunigung
        * positive Beschleunigung in 2 Stufen
        * negative Beschleunigung in 2 Stufen
    - Seitwärtsflug (Blickrichtung 90° zur Flugrichtung)
        * keine Beschleunigung
        * positive Beschleunigung in 2 Stufen
        * negative Beschleunigung in 2 Stufen

Für Bewegungen ohne Beschleunigung wurde die Kamera stetig um 2 cm an der Schiene entlang verschoben und ein Bild aufgenommen. So enstand eine Sequenz von 46 Frames. Für die Beschleunigungssequenzen (positiv als auch negativ) in zwei Stufen wurde die Bewegung nach der 7. und 14. Bildaufnahme jeweils verdoppelt bzw. halbiert. Ursprünglich war beabsichtigt, auch die Freiflugaufnahmen mit der Videofunktion der Webcam durchzuführen. Hier kam es zu Problemen mit schwankenden Bildwiederholungsraten und Bildstörungen. Aus diesen Gründen wurde auf eine *FinePix S5500* von Fujifilm zurückgegriffen. Sie besitzt einen horizontalen Öffnungswinkel von 50°, einen vertikalen Öffnungswinkel von 39° und erstellt Filmaufnahmen in einer Auflösung von 640×480 mit einer Bildwiederholungsrate von 30 FPS. Folgende Testsequenzen wurden aufgenommen:

- Steig-Sink-Flug
    - Ausgehend vom Boden wurde die Kamera in einer vertikalen Bewegung in Schulterhöhe gebracht und dann wieder heruntergelassen. Die Blickrichtung der Kamera wurde beibehalten.
- Vor-Zurück-Flug
    - Ausgehend vom Boden wurde die Kamera in einer vertikalen Bewegung in Schulterhöhe gebracht. Es erfolgte eine Vorwärts- und eine Rückwärtsbewegung. Danach wurde die Kamera wieder herabgelassen. Die Blickrichtung der Kamera wurde beibehalten.
- Links-Rechts-Flug
    - Ausgehend vom Boden wurde die Kamera in einer vertikalen Bewegung in Schulterhöhe gebracht. Es erfolgte eine Seitwärtsbewegung nach links und rechts. Danach wurde die Kamera wieder herabgelassen. Die Blickrichtung der Kamera wurde beibehalten.

- Kreis-Flug
  - Ausgehend vom Boden wurde die Kamera in einer vertikalen Bewegung in Schulterhöhe gebracht. Es erfolgte eine Kreisbewegung parallel zum Boden, wobei die Blickrichtung der Kamera beibehalten wurde. Danach wurde die Kamera wieder herabgelassen.
- Kombinations-Flug
  - Ausgehend vom Boden wurde die Kamera in einer vertikalen Bewegung in Schulterhöhe gebracht. Es erfolgte eine Kombination der vorher durchgeführten Flüge, wobei die Ausrichtung der Kamera nicht beibehalten wurde. Danach wurde die Kamera wieder herabgelassen.

Der Start- und Landepunkt war in allen Testsequenzen gleich. Unterstützt wurden die Aufnahmen durch das am Lehrstuhl vorhandene *Tracking System*. Das Resultat waren Filmaufnahmen, in der jedes *Frame* durch den Standort der Kamera (Orientierung und Position) identifiziert werden konnte. Die sich daraus ergebende Trajektorie wurde als Vergleich herangezogen.

(a) **(b)**

Abbildung 3.8: (a) Versuchsaufbau mit Logitech Webcam, Führungsschiene und Notebook. **(b)** Betrachtete Szenerie für den ersten Teil der Testsequenzen.

Kapitel 4

Ergebnisse

4.1 Virtuelle Testsequenzen

Es werden die Ergebnisse der Testflüge der virtuellen Umgebung präsentiert. Die Ergebnisse der durchgeführten Testsequenzen sind in beiden virtuellen Karten sehr ähnlich. Aufgrund der Fülle und Redundanz der Daten wird in Abschnitt 4.4 exemplarisch der Translationsflug ohne Beschleunigung in der *Corridor-Map* und der *Columns-Map* verglichen. Alle weiteren Ergebnisse beziehen sich auf die *Corridor-Map*. Ausnahmen werden explizit gekennzeichnet und erwähnt.

4.1.1 Strikter Translationsflug ohne Nickbewegung

Es folgt die Präsentation der Ergebnisse der strikten Translationsflüge ohne Nickbewegung. Für die Berechnung der Geschwindigkeitskurven wurde eine Bildwiederholungsrate von 21 FPS angenommen. Die initiale Geschwindigkeit wurde bei Vorwärts- und Seitwärtsflügen auf $1\frac{m}{s}$ und bei Rückwärtsflügen auf $-1\frac{m}{s}$ festgelegt. Die Ergebnisse der Seitwärtsflüge stammen aus der *Columns-Map*. Hier erfolgte ein Test bei konstanter Geschwindigkeit. Eine Wiederholung für eine positive und negative Beschleunigung wurde nicht durchgeführt.

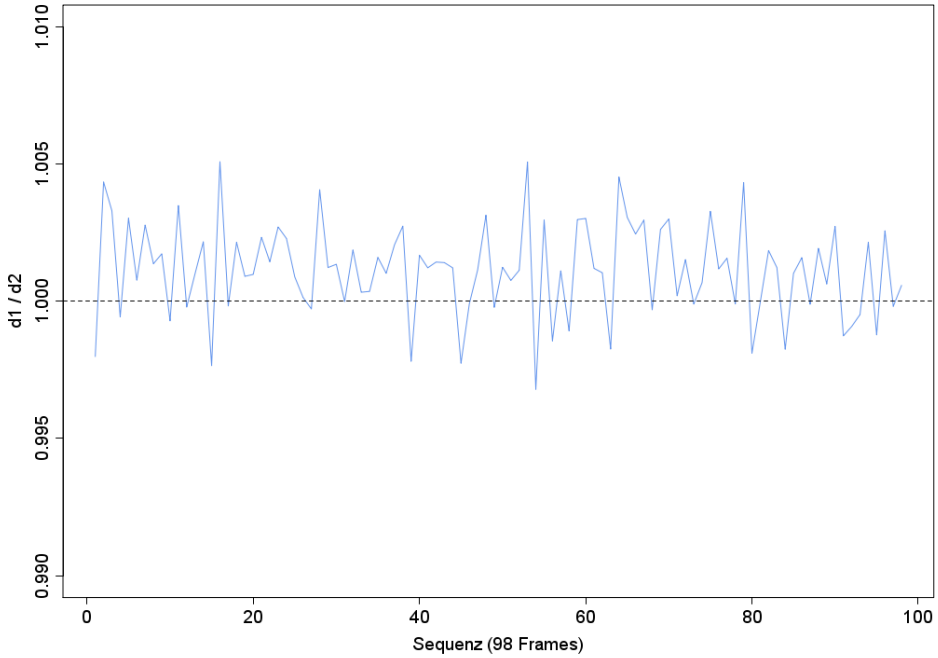
In Abbildung 4.1 ist der strikte Translationsflug in Vorwärtsrichtung ohne Beschleunigung und ohne Nickbewegung zu sehen. Der wahre Wert ist als gestrichelte Linie in der Abbildung dargestellt. Man erkennt eindeutig eine positive Tendenz der berechneten Tiefenquotienten. Dies führt dazu, dass die geschätzte Eigengeschwindigkeit bereits nach 4 Sekunden um $+0,12\frac{m}{s}$ angestiegen ist.

In der Abbildung 4.2 wurde eine dreistufig beschleunigte Bewegung (jeweils Geschwindigkeitsverdopplung) durchgeführt. Wie zu erwarten, erkennt man im

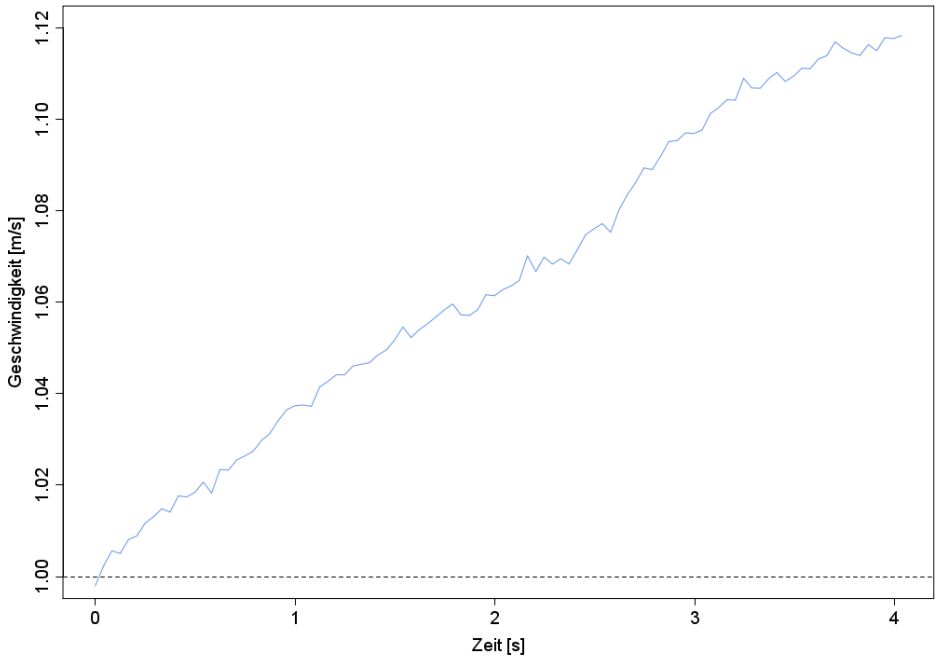
Beschleunigungsgraph in Abbildung 4.2a bei *Frame* 25, 50 und 75 die Bechleunigung. Die Werte der Beschleunigung sind an den Signalspitzen zu erkennen. Die Abbildung 4.2b zeigt die Geschwindigkeit. Wie zuvor sind die wahren Werte als gestrichelte Linie eingezeichnet. Die dreistufige Erhöhung der Geschwindigkeit ist gut zu erkennen. Dasselbe gilt für Abbildung 4.3. Auch hier lässt sich eine positive Tendenz der Tiefenquotienten sehen. So ist die in drei Stufen beschleunigte Bewegung $+0,76\frac{m}{s}$ über den Idealwert. Bei der abgebremsten Bewegung ist die Differenz im zweistelligen Nachkommabereich zu erkennen.

Für die Abbildungen 4.4, 4.5 und 4.6 gelten analog die bereits genannten Aussagen. Die Stufen der Geschwindigkeitsänderung sind gut zu erkennen, jedoch ist jeder geschätzte Tiefenquotient mit einen unbekannten positiven Fehler behaftet.

Abbildung 4.7 zeigt eine nichtbeschleunigte Seitwärtsbewegung ohne Nickbewegung. Diese Testsequenz stammt aus der *Columns-Map*. Der FOE ist außerhalb des Bildbereichs. In der Beschleunigungskurve sind eine große Anzahl von Signalspitzen, so dass die Geschwindigkeitskurve an einer Stelle ein Maximum von $+269\frac{km}{h}$ erreicht. Die Ursache für diese schlechte Eigengeschwindigkeitsschätzung liegt in einer fehlerhaften Eigenbewegungsschätzung des Kanatani aufgrund eines des zu geringen FOV. Hier wird auf entsprechende Untersuchung im Abschnitt 4.3 verwiesen.

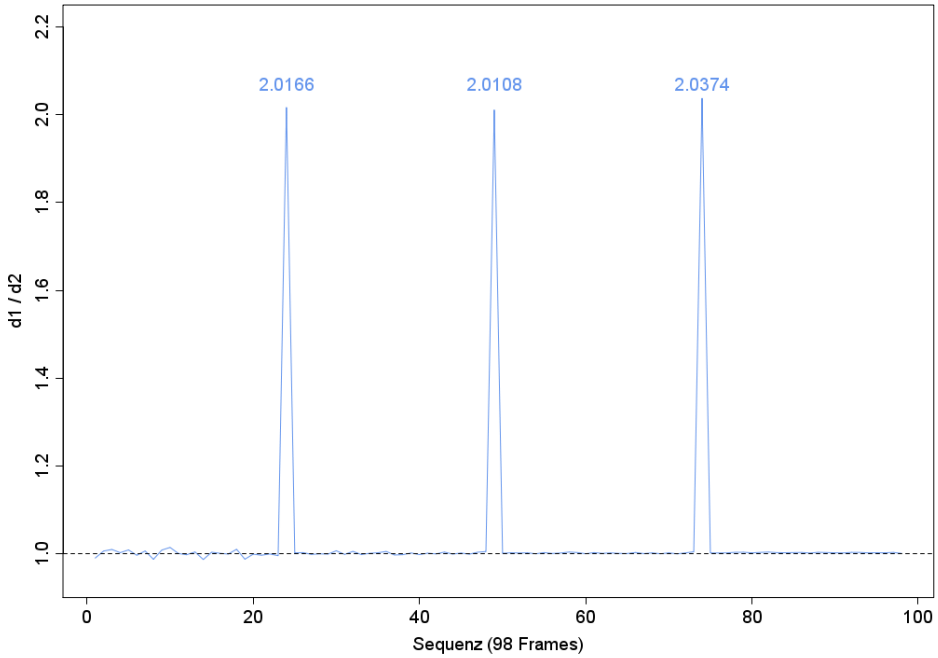


(a)

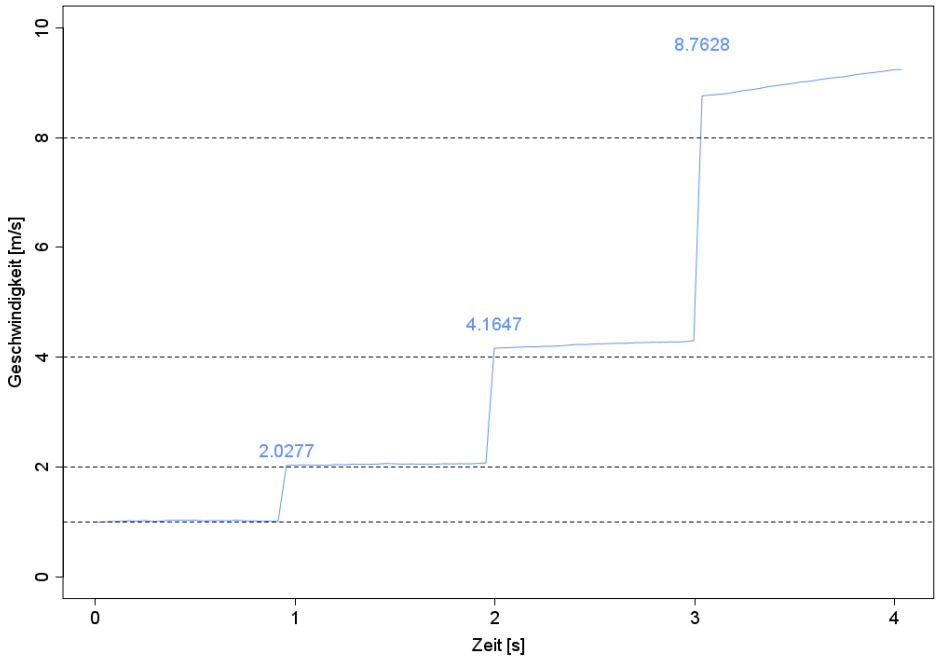


(b)

Abbildung 4.1: Strikter Translationsflug in Vorwärtsrichtung ohne Beschleunigung und ohne Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

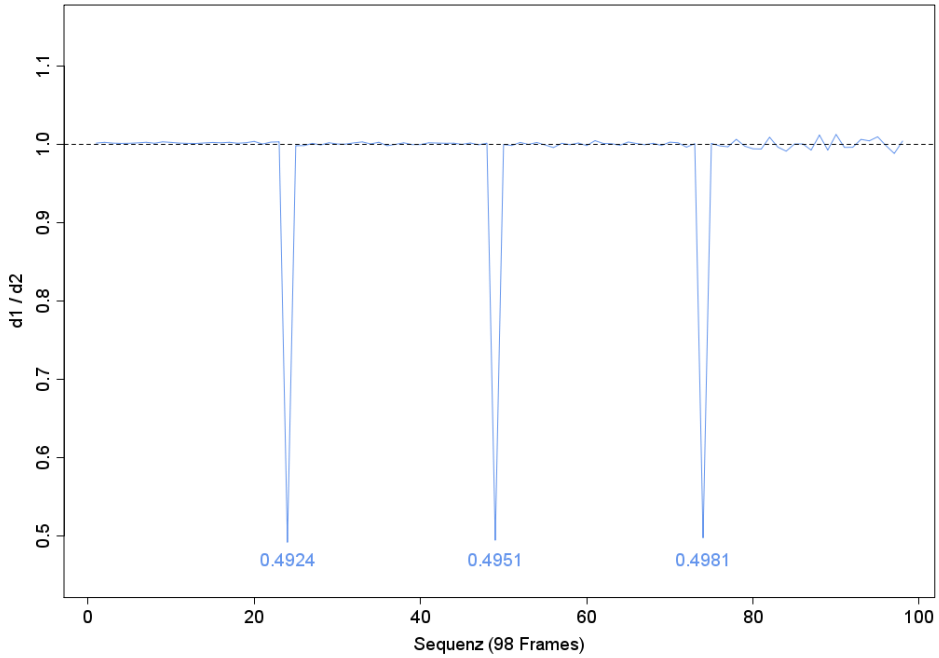


(a)

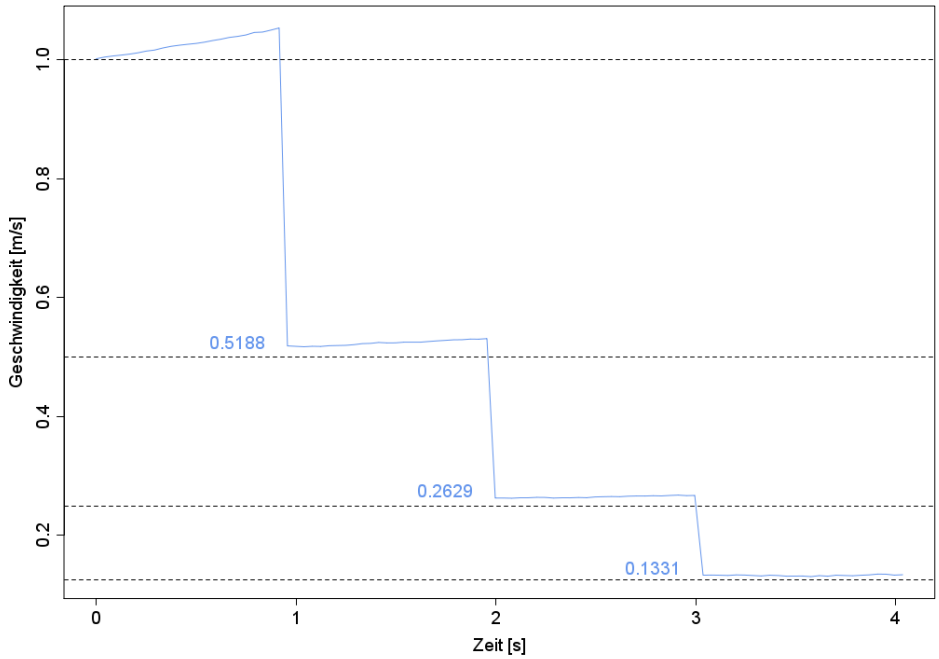


(b)

Abbildung 4.2: Strikter Translationsflug in Vorwärtsrichtung mit positiver Beschleunigung und ohne Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

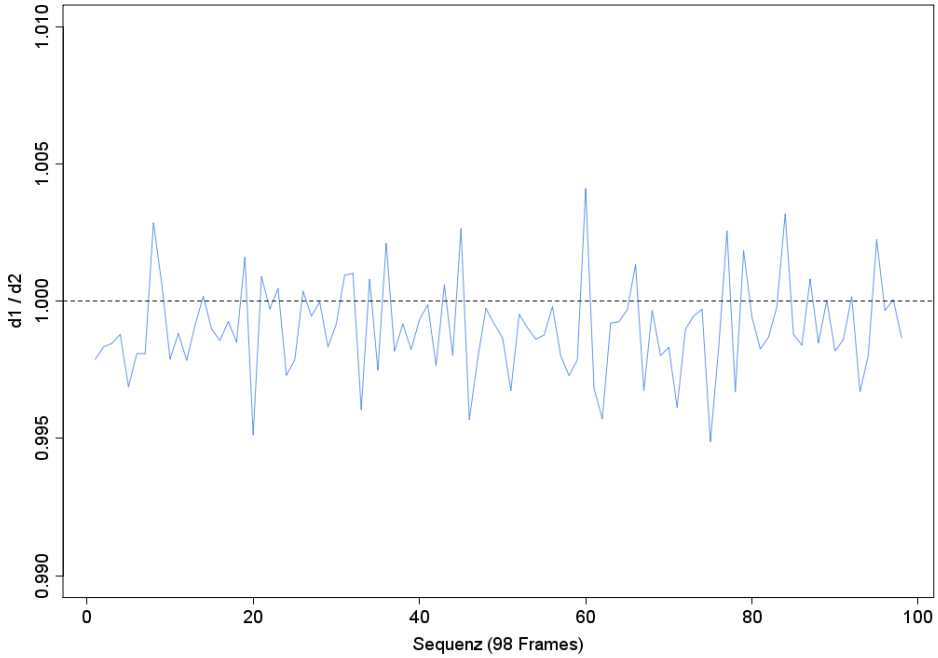


(a)

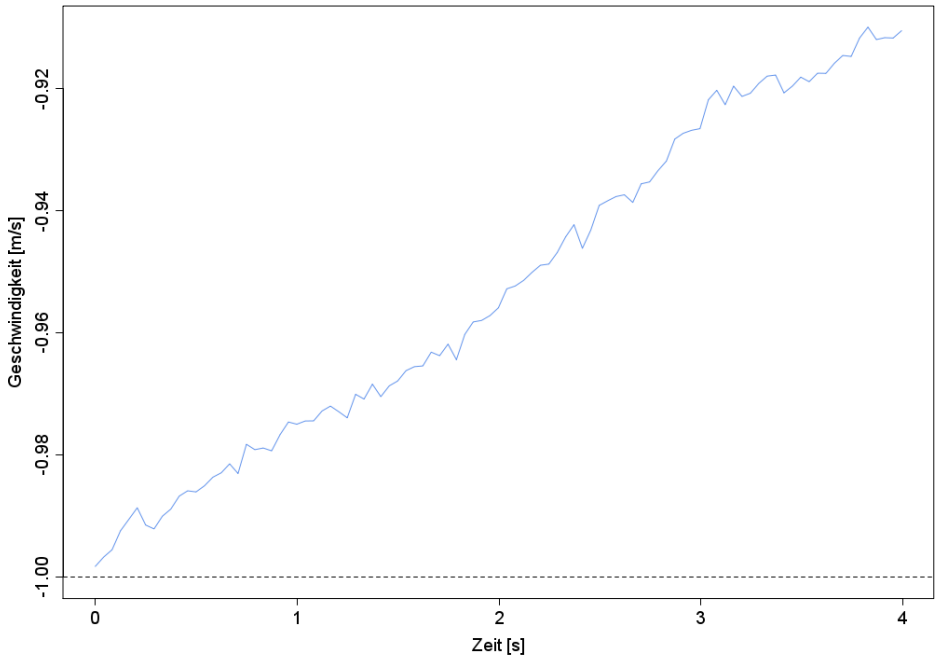


(b)

Abbildung 4.3: Strikter Translationsflug in Vorwärtsrichtung mit negativer Beschleunigung und ohne Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

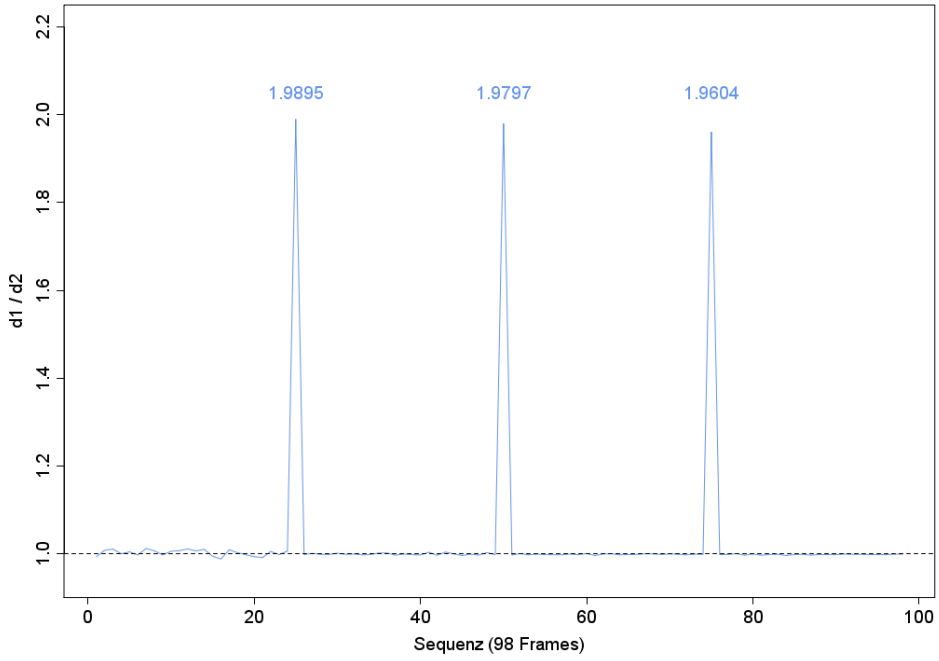


(a)

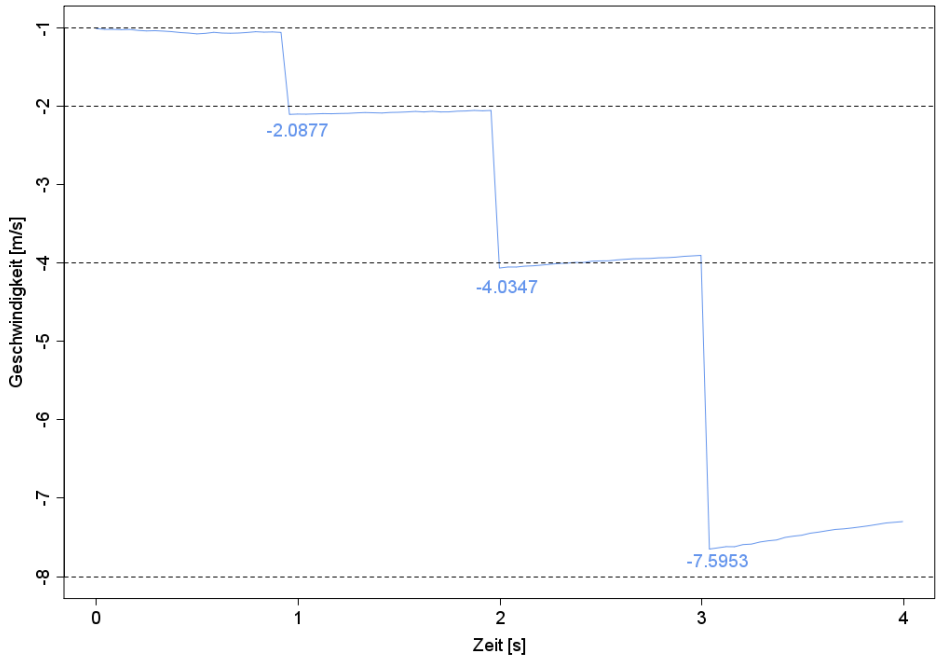


(b)

Abbildung 4.4: Strikter Translationsflug in Rückwärtsrichtung ohne Beschleunigung und ohne Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

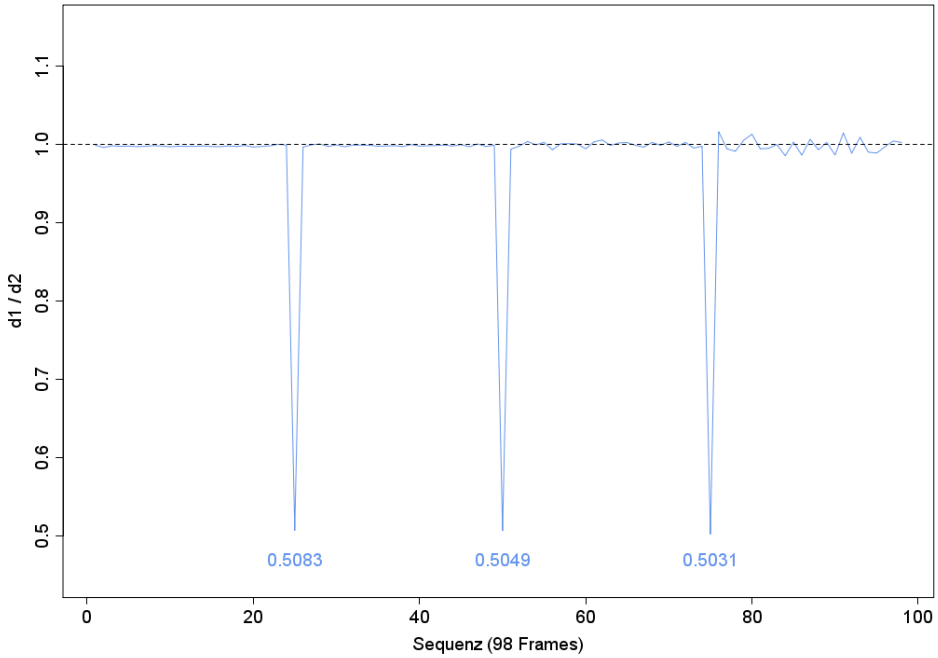


(a)

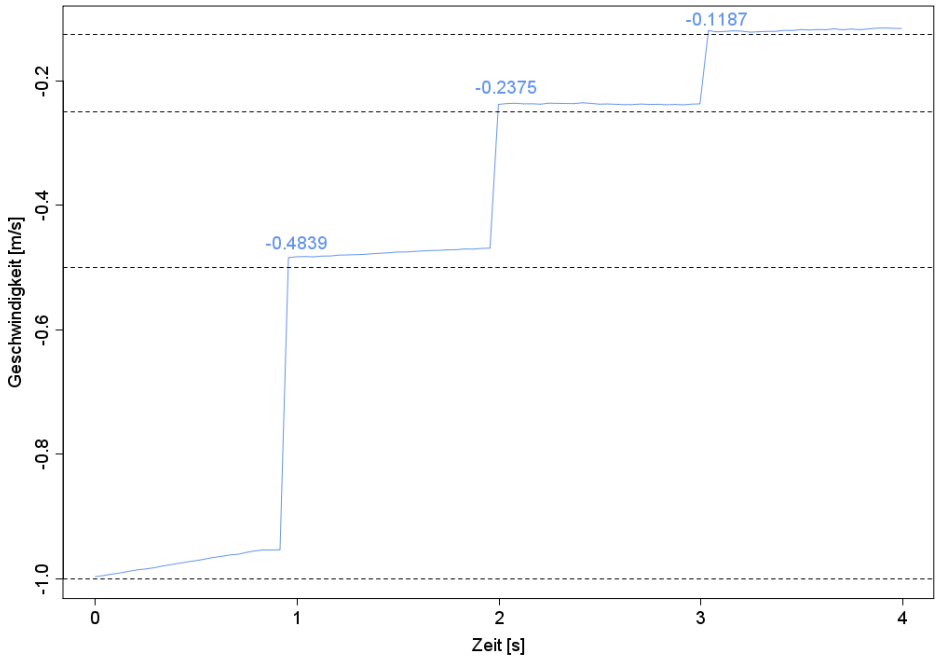


(b)

Abbildung 4.5: Strikter Translationsflug in Rückwärtsrichtung mit positiver Beschleunigung und ohne Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

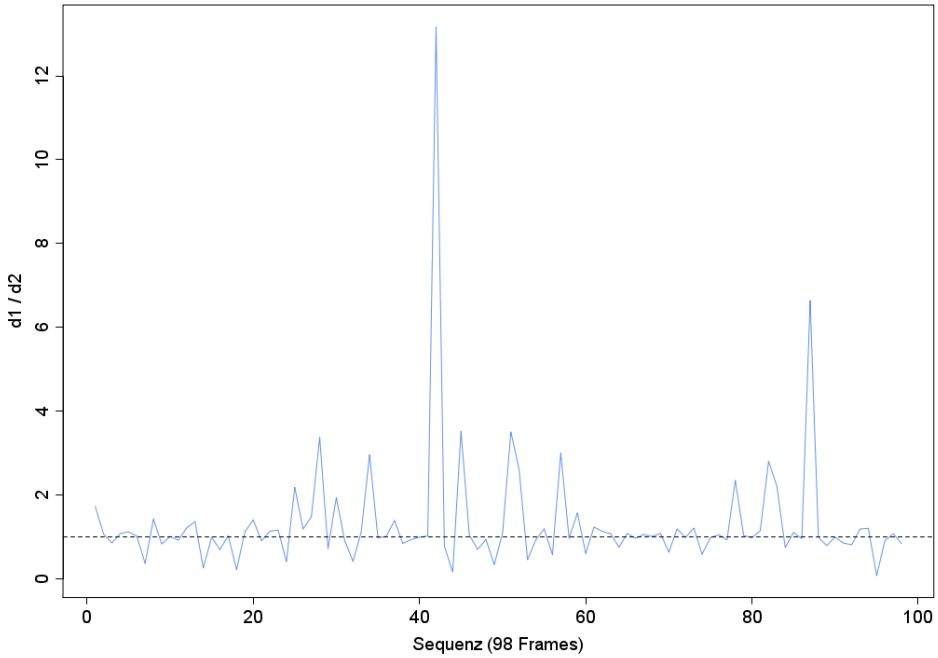


(a)

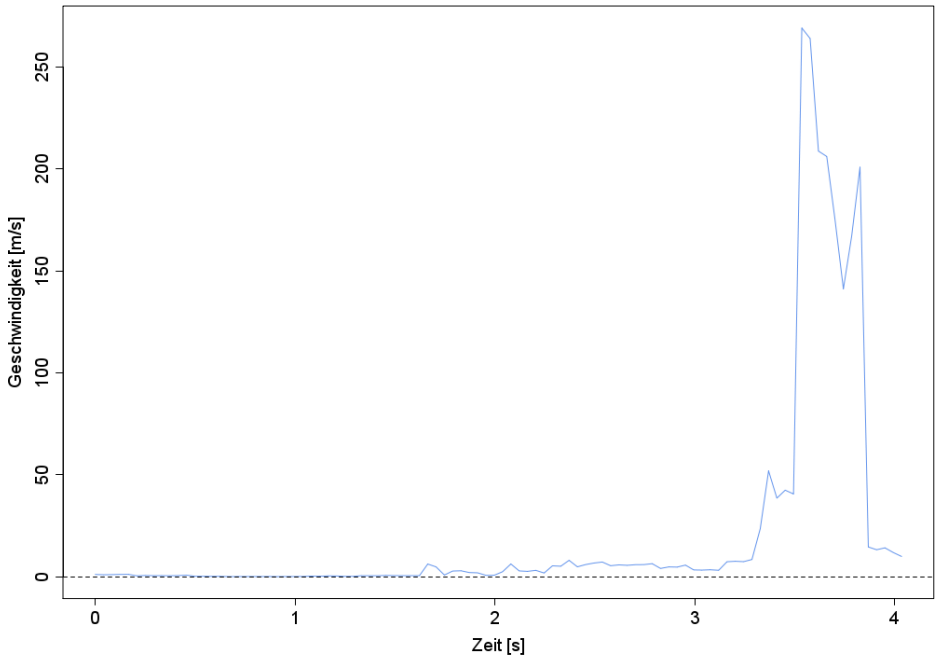


(b)

Abbildung 4.6: Strikter Translationsflug in Rückwärtsrichtung mit negativer Beschleunigung und ohne Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.



(a)



(b)

Abbildung 4.7: Strikter Translationsflug in Seitwärtsrichtung ohne Beschleunigung und ohne Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

4.1.2 Translationsflug mit Nickbewegung

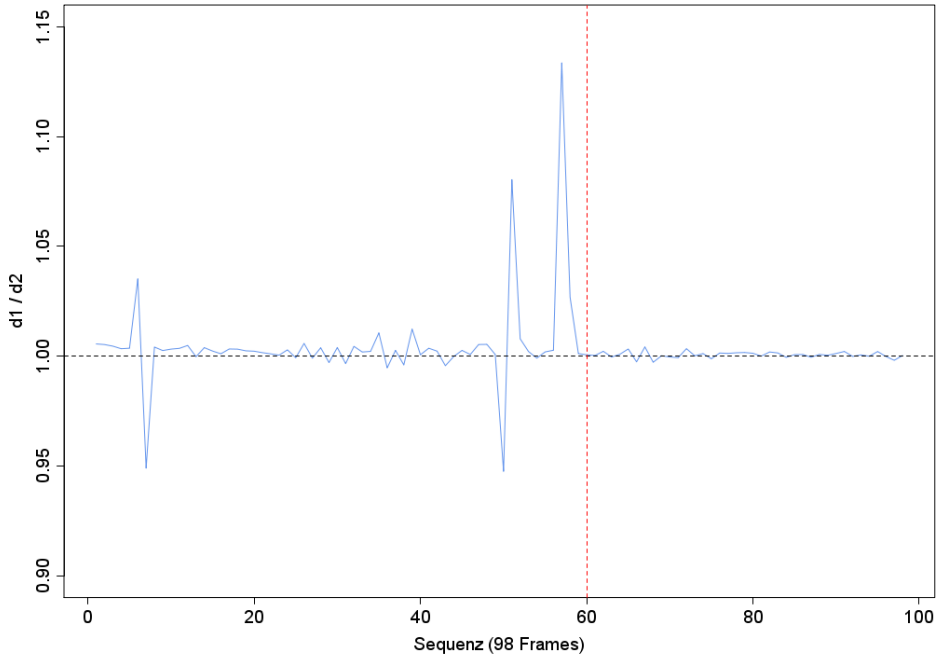
Es folgt die Präsentation der Ergebnisse der strikten Translationflüge mit Nickbewegung. Für die Berechnung der Geschwindigkeitskurven wurde eine Bildwiederholungsrate von 21 FPS angenommen. Die initiale Geschwindigkeit wurde bei Vorwärtsflügen auf $1\frac{m}{s}$ und bei Rückwärtsflügen auf $-1\frac{m}{s}$ festgelegt. Die Nickbewegung der virtuellen Flugdrohne beginnt mit dem ersten Frame der Flugsequenz und ist bei Frame 60 beendet, was im Plot durch eine rote Linie gekennzeichnet ist. Die Drohne fliegt in diesem angestellten Winkel nun bis zum Sequenzende. Seitwärtsflüge wurden keine durchgeführt.

Es ist anzumerken, dass die hier virtuell durchgeführte Nickbewegung nicht mit einer Nickbewegung in der Realität vergleichbar ist. Unter realen Bedingungen führt eine einsetzende Nickbewegung zu einer positiven Beschleunigung in die abnickende Richtung. Diese Beschleunigung ist beendet, wenn der Nickwinkel sich nicht mehr ändert. Dies wurde bei den Testsequenzen nicht berücksichtigt.

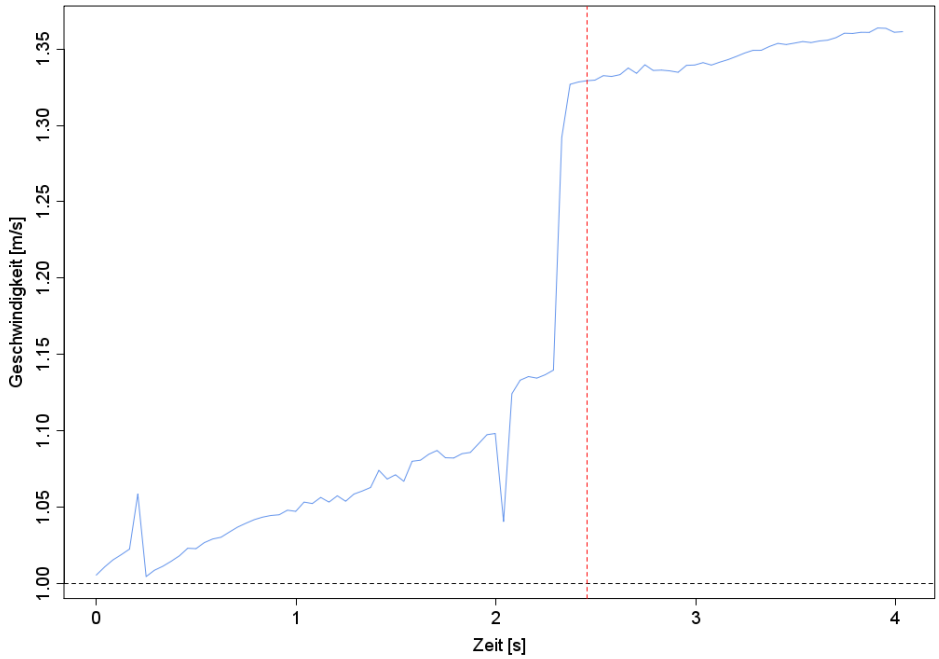
In Abbildung 4.8 sind die Ergebnisse des nichtbeschleunigten Translationsflugs mit Nickbewegung gezeigt. Die auftretenden entgegengesetzten Signalspitzen ähnlicher Amplitude (*Frame* 7 und 8 und *Frame* 49 und 50) haben höchstwahrscheinlich ihren Ursprung in einer schwankenden Samplingrate, welche bei der Generierung der Testdaten auftrat. Die positive Signalspitze kurz vor Beendigung der Rotation könnte eine Eigenheit in der 3D-Umgebung sein. Verringert sich der Winkel zwischen zwei Achsen, so existiert ein Schwellenwert in der Art, dass der Winkel ab einen bestimmten Punkt einfach einrastet. Diese einleuchtende Erklärung wurde jedoch bislang nicht belegt. Ungeachtet dessen ist zwischen diesen eben erläuterten Fehlerspitzen eine positive Tendenz der Geschwindigkeitsschätzung auszumachen. Diese Erklärungen gelten ebenso für die Abbildungen 4.9 bis 4.13.

Die Abbildungen 4.9 und 4.10 zeigen eine dreistufig beschleunigte bzw. abgebremste Translationsbewegung mit Nickbewegung. Die drei charakteristischen Signalstufen in *Frame* 25, 50 und 75 sind gut zu erkennen.

Für die Abbildungen 4.11 bis 4.13 gelten die entsprechenden - bereits aufgeführten - Aussagen dieses Kapitels.

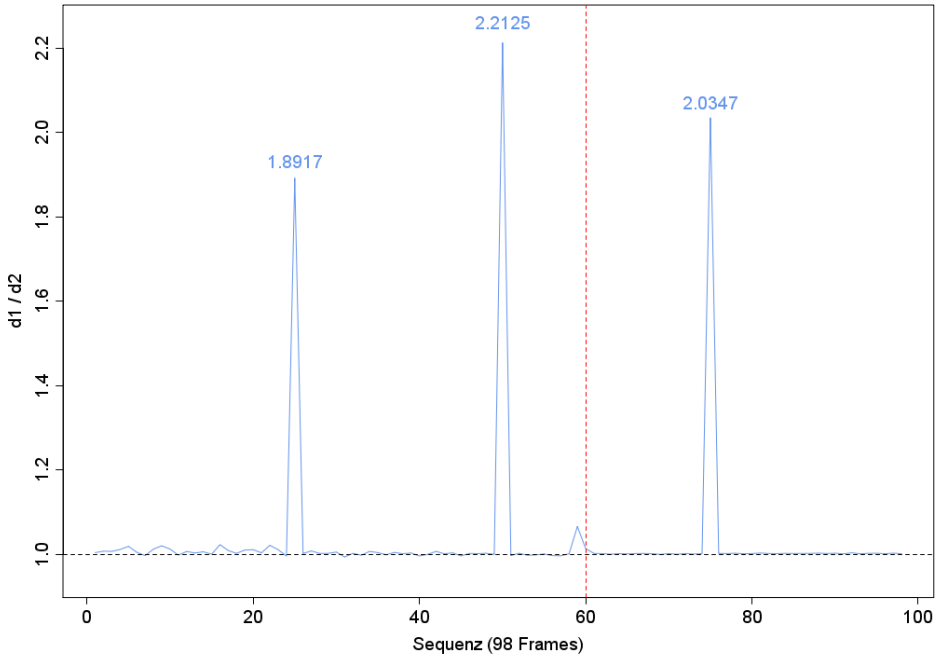


(a)

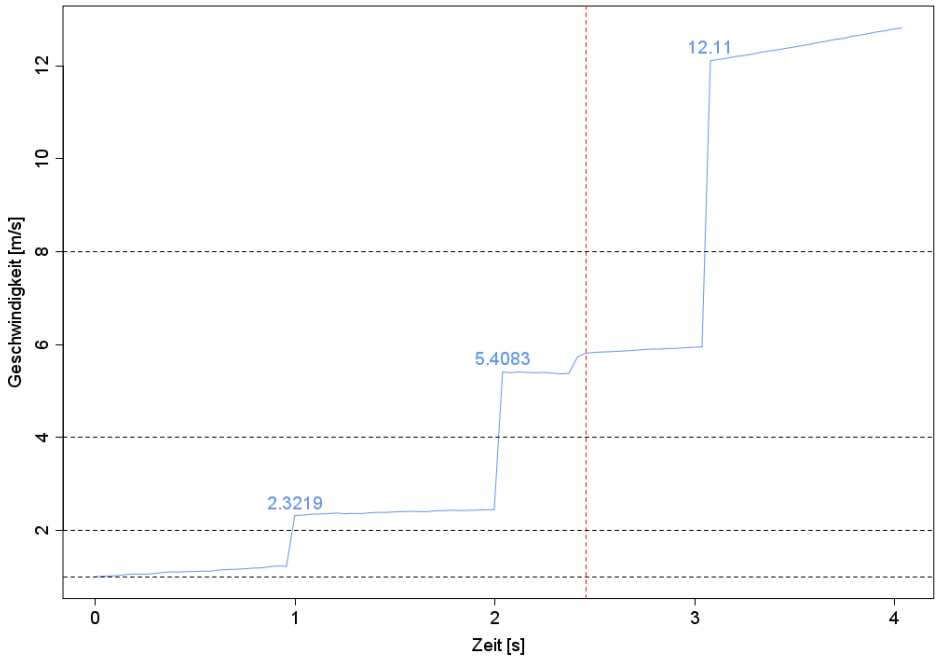


(b)

Abbildung 4.8: Translationsflug in Vorwärtsrichtung ohne Beschleunigung mit Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

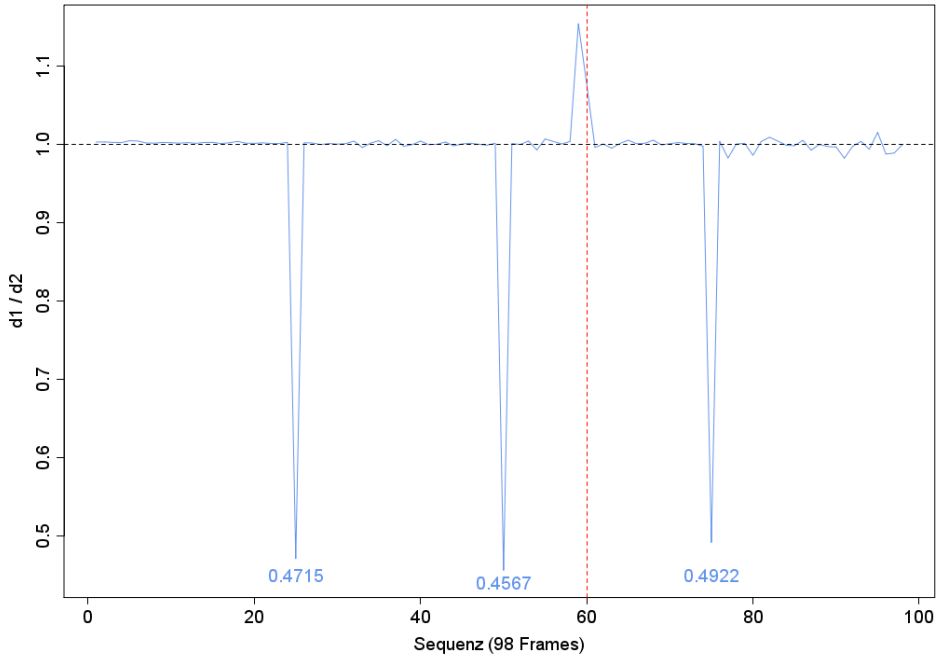


(a)

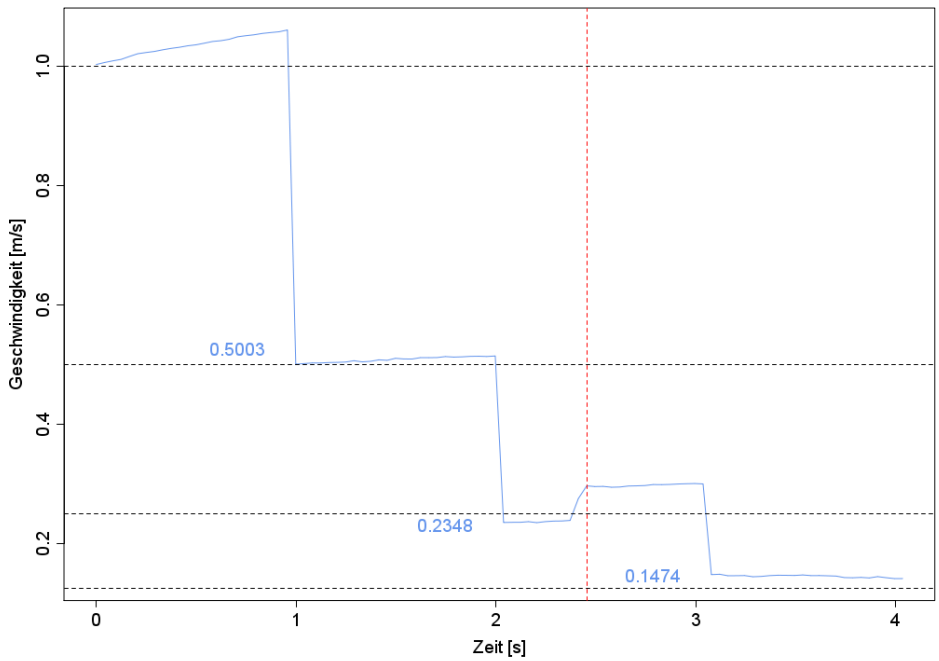


(b)

Abbildung 4.9: Translationsflug in Vorwärtsrichtung mit positiver Beschleunigung mit Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

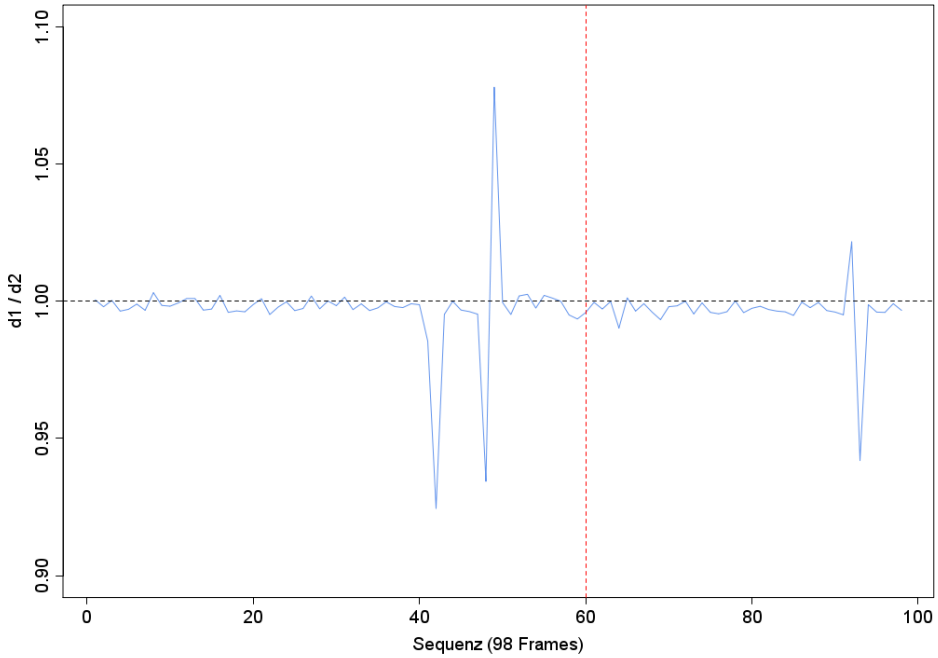


(a)

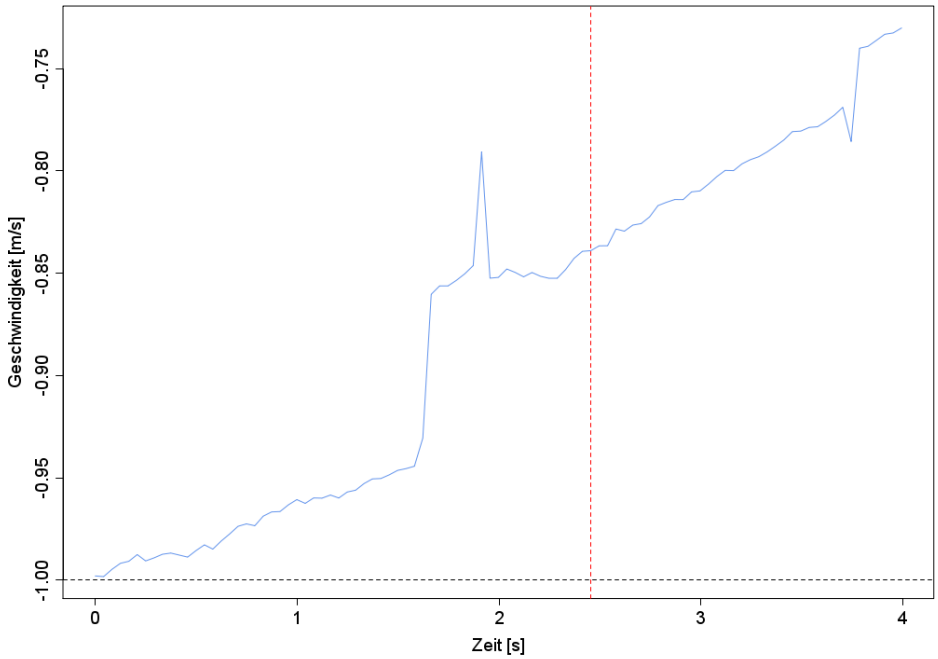


(b)

Abbildung 4.10: Translationsflug in Vorwärtsrichtung mit negativer Beschleunigung mit Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

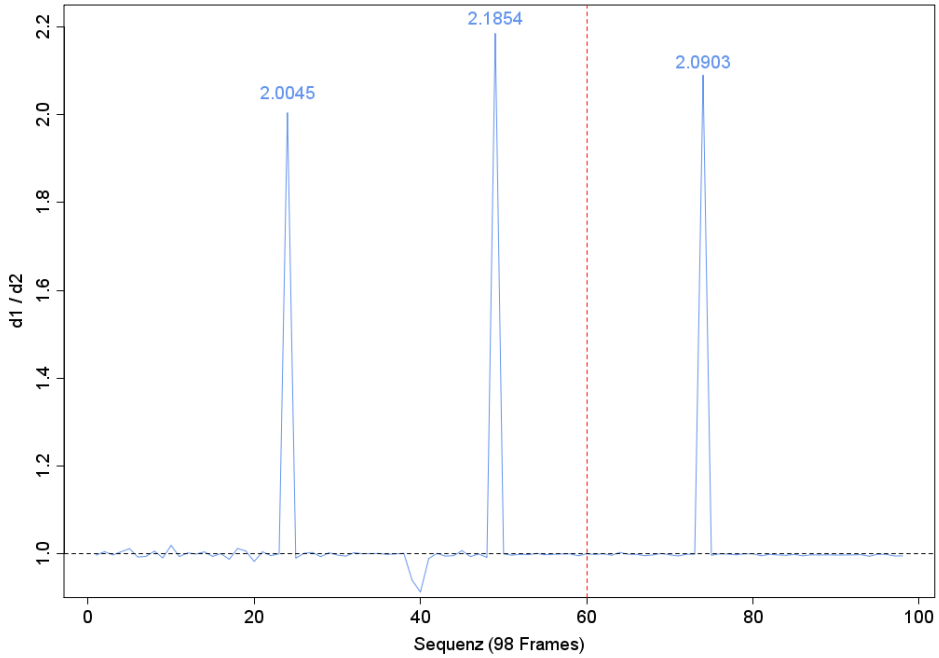


(a)

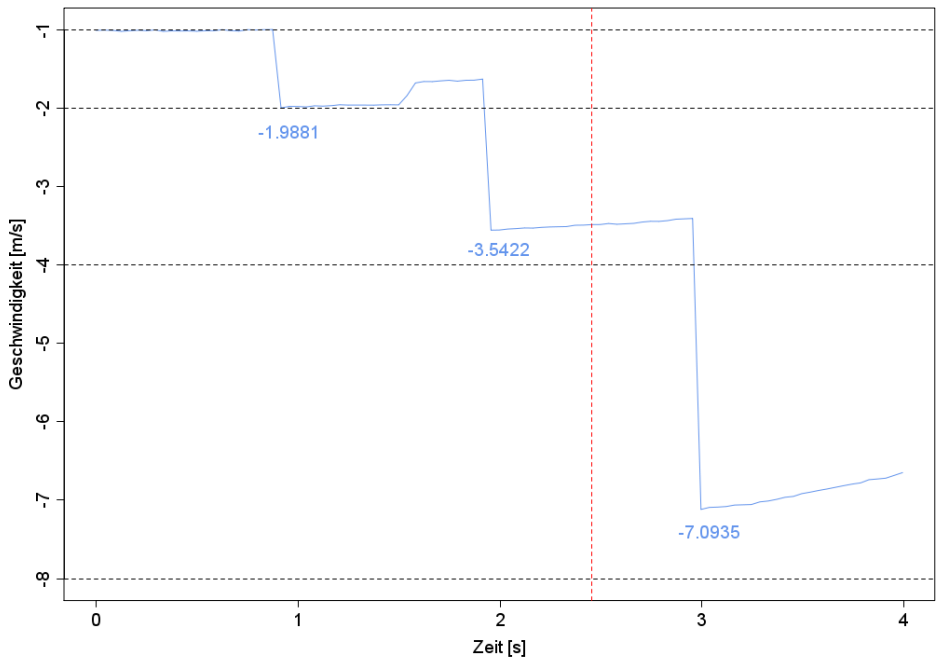


(b)

Abbildung 4.11: Translationsflug in Rückwärtsrichtung ohne Beschleunigung mit Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

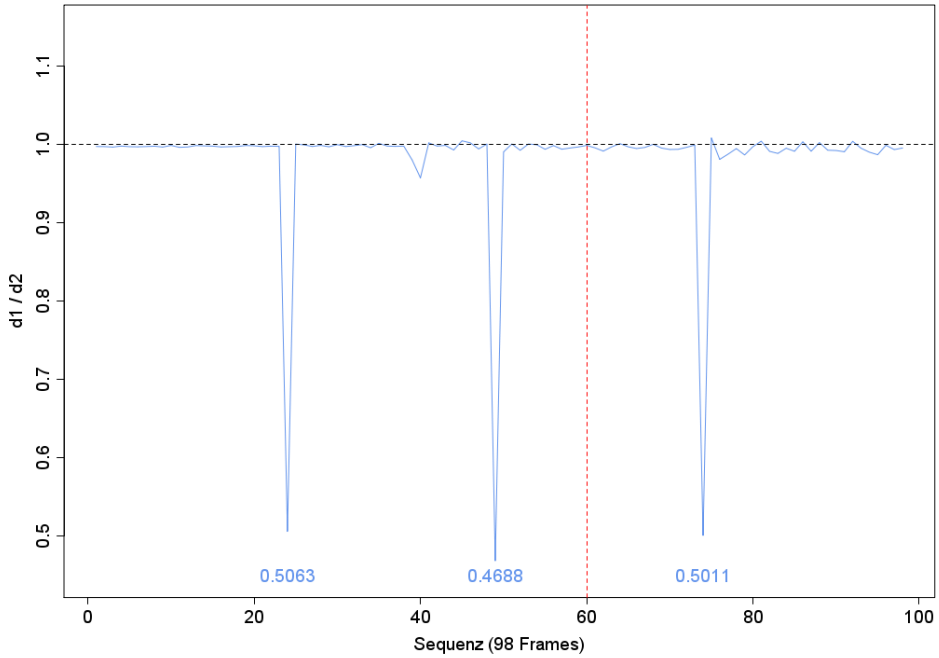


(a)

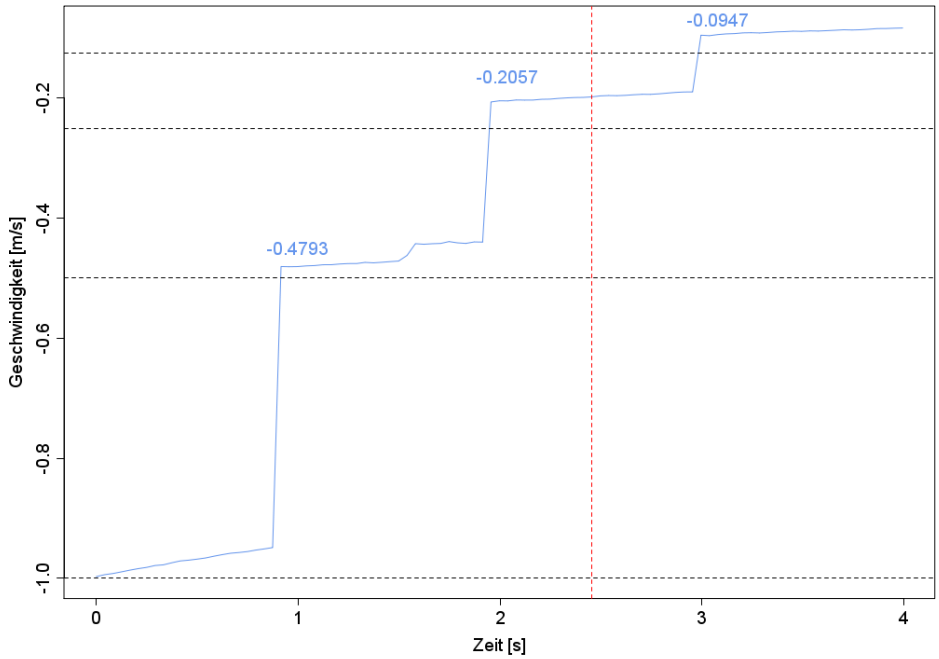


(b)

Abbildung 4.12: Translationsflug in Rückwärtsrichtung mit positiver Beschleunigung mit Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.



(a)



(b)

Abbildung 4.13: Translationsflug in Rückwärtsrichtung mit negativer Beschleunigung mit Nickbewegung. **(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

4.2 Reale Flugsequenzen

4.2.1 Strikter Translationsflug

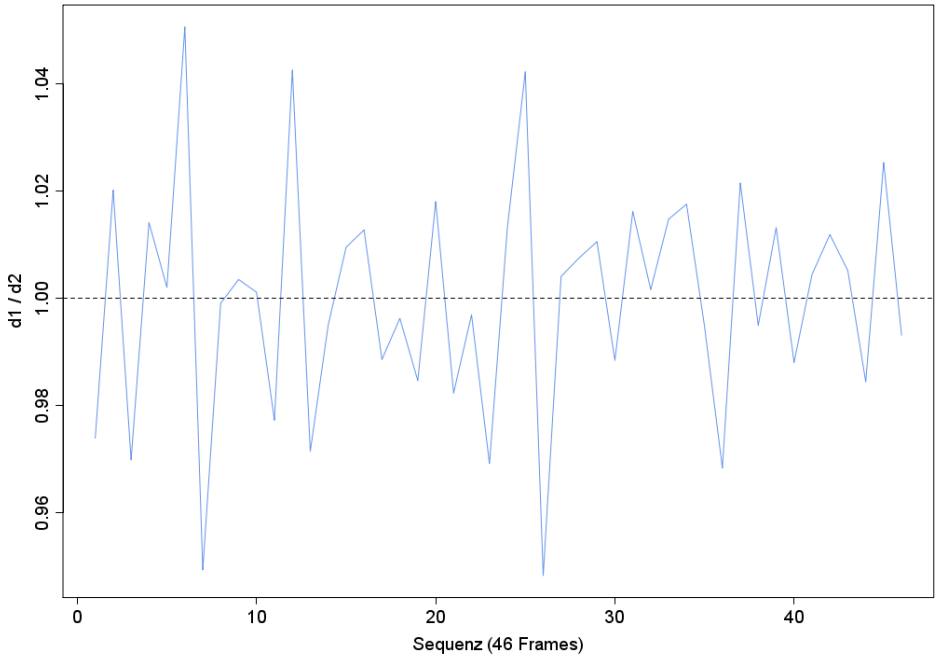
Im Folgenden werden die Ergebnisse des ersten Versuchsaufbaus präsentiert. Es handelt sich hier um die strikten Translationsflüge, welche mit Hilfe der *QuickCam Pro 9000* und einer Führungsschiene erzeugt wurden. Für die Berechnung der Geschwindigkeitskurven wurde eine Bildwiederholungsrate von 21 FPS angenommen. Die initiale Geschwindigkeit wurde bei Vorwärts- und Seitwärtsflügen auf $1\frac{m}{s}$ und bei Rückwärtsflügen auf $-1\frac{m}{s}$ festgelegt. Der Aufbau der Plots entspricht den vorangegangen Erklärungen dieses Kapitels. Der einzige Unterschied ist, dass aufgrund der Länge der Führungsschiene nur 2 Beschleunigunsstufen realisiert werden konnten. Für Details zum Versuchsaufbau wird auf Abschnitt 3.3 verwiesen.

Die Abbildung 4.14 zeigt den strikten Translationsflug in Vorwärtsrichtung ohne Beschleunigung. Im Vergleich zu den entsprechenden virtuellen Testsequenzen schwankt die Beschleunigungskurve bei den realen Flugsequenzen stärker.

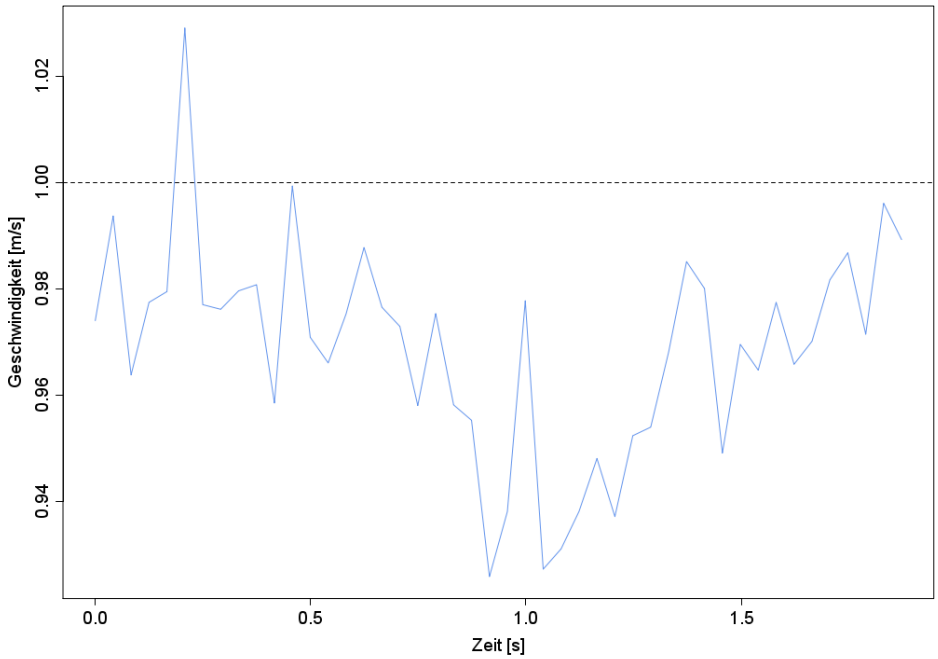
Die Abbildungen 4.15 und 4.16 zeigen die positive bzw. negative Beschleunigung des strikten Translationsfluges. Die charakteristischen Signalspitzen sind gut zu erkennen.

Die entsprechenden Beschleunigungsarten für den Rückwärtsflug sind den Abbildungen 4.17 bis 4.19 dargestellt. Das bisherige beobachtete Verhalten lässt sich auf diese Kurven ebenfalls übertragen.

Die letzten drei Testflüge des ersten Versuchsaufbau sind Seitwärtsflüge in allen drei Beschleunigungsarten. Die Abbildung 4.20 zeigt eine nichtbeschleunigte Seitwärtsbewegung, während die Abbildungen 4.21 und 4.22 eine positive bzw. negative beschleunigte Seitwärtsbewegung in zwei Stufen darstellen. Die Signalspitzen der Beschleunigungsstufen sind erkennbar.

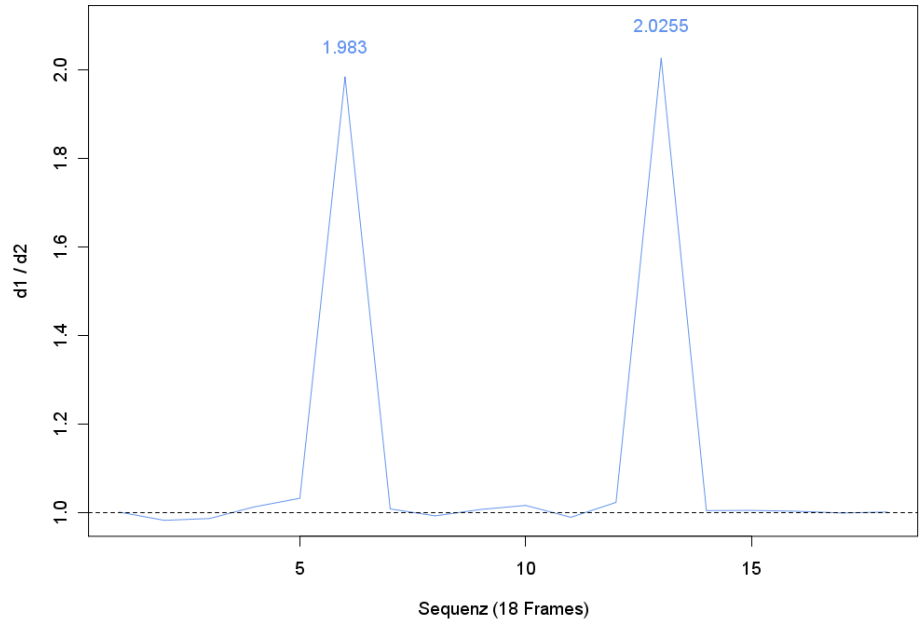


(a)

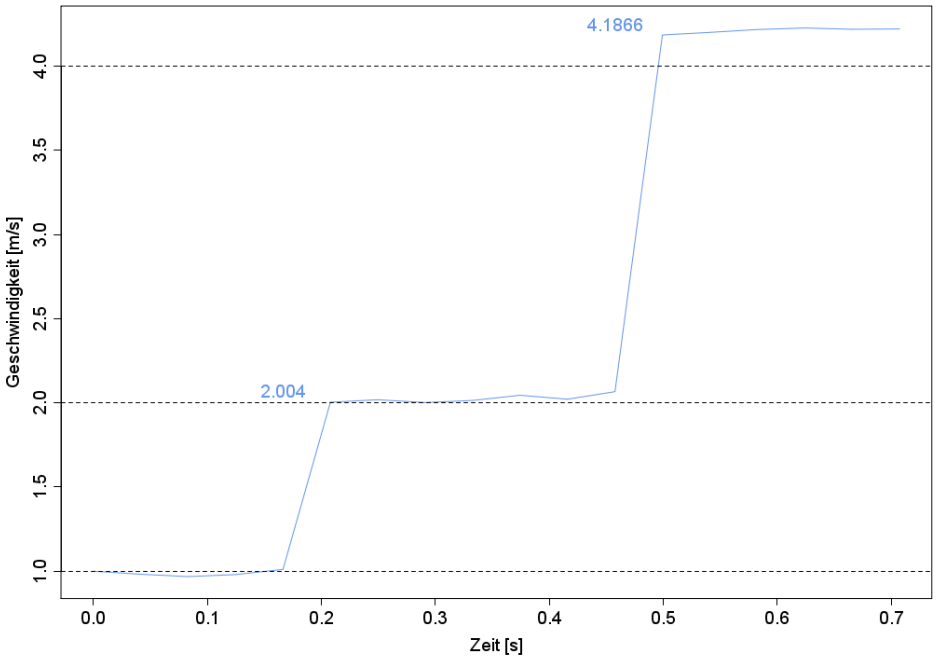


(b)

Abbildung 4.14: Strikter Translationsflug in Vorwärtsrichtung ohne Beschleunigung.**(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

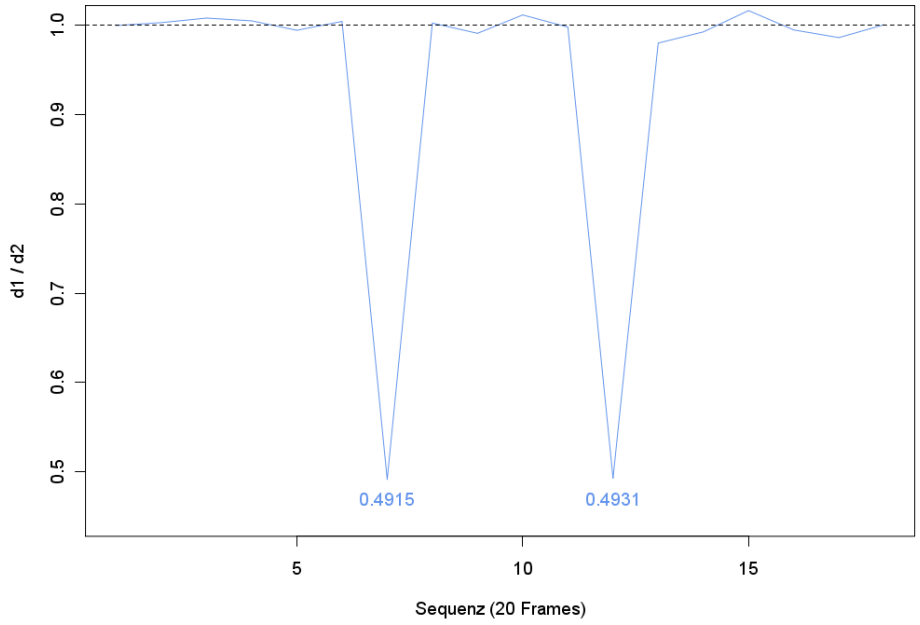


(a)

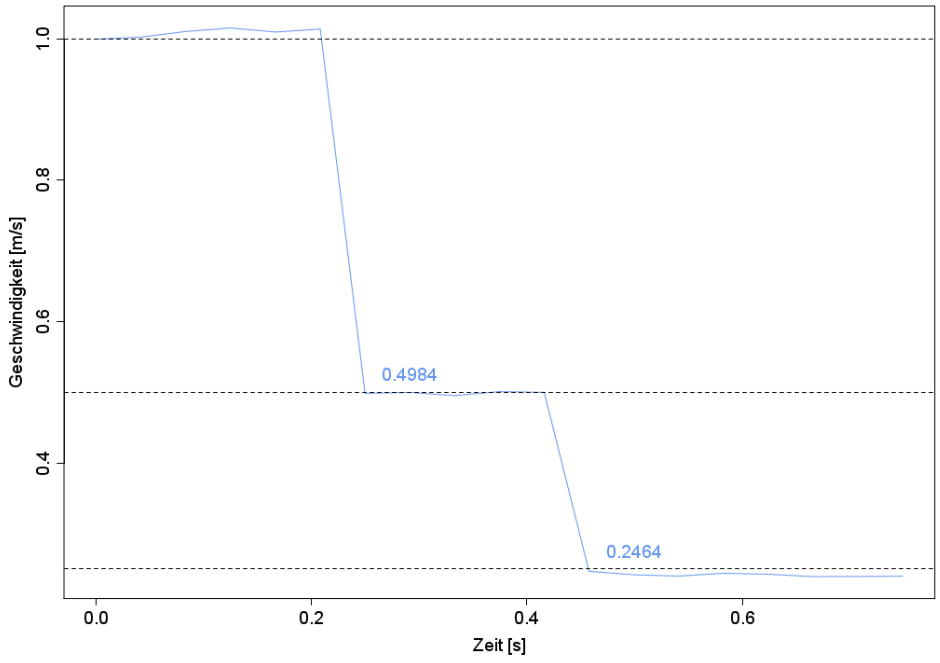


(b)

Abbildung 4.15: Strikter Translationsflug in Vorwärtsrichtung. Positive Beschleunigung.**(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

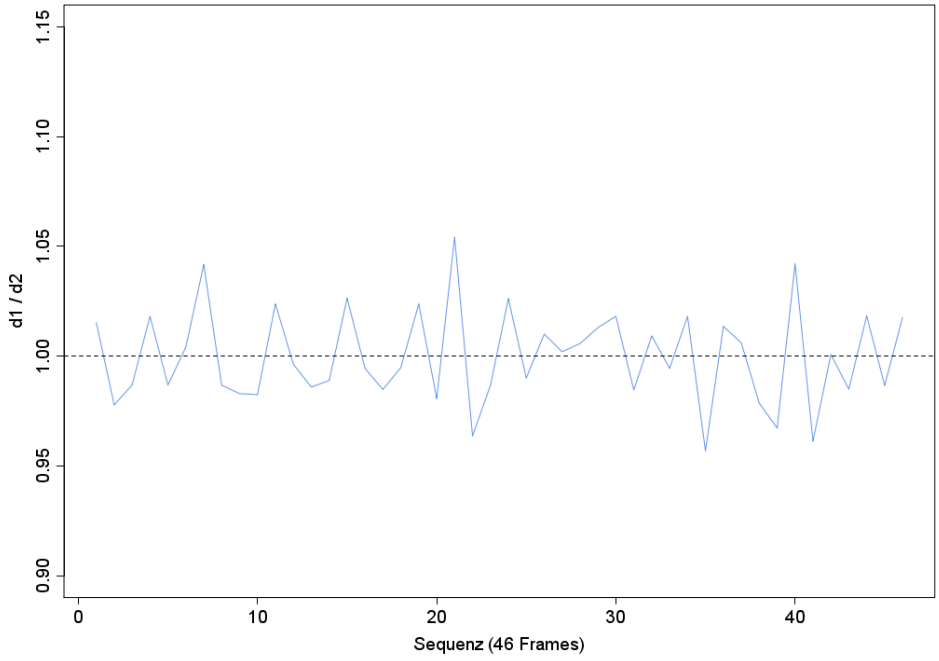


(a)

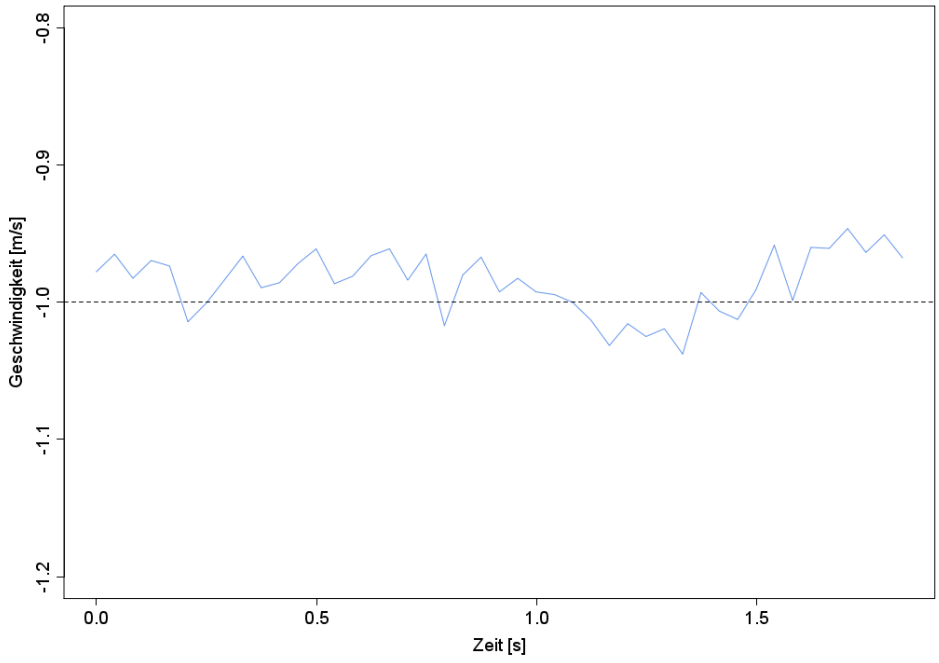


(b)

Abbildung 4.16: Strikter Translationsflug in Vorwärtsrichtung. Negative Beschleunigung.**(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

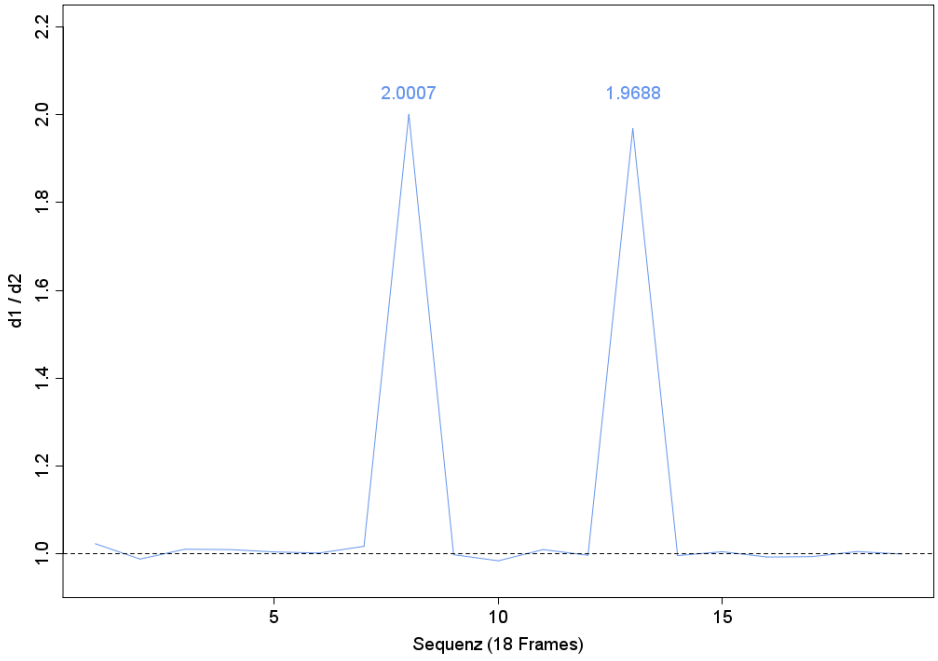


(a)

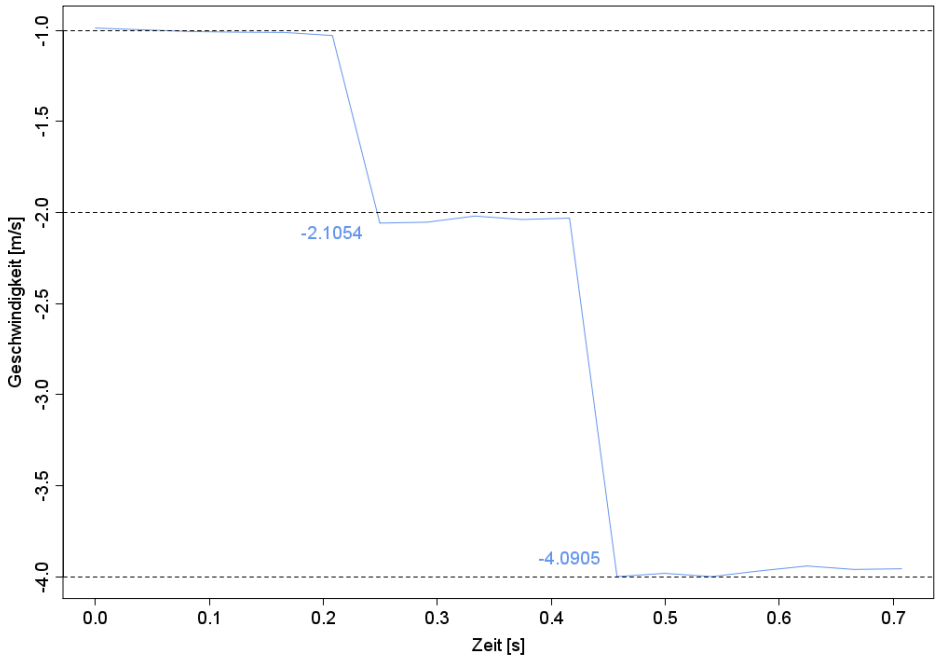


(b)

Abbildung 4.17: Strikter Translationsflug in Rückwärtsrichtung ohne Beschleunigung.**(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

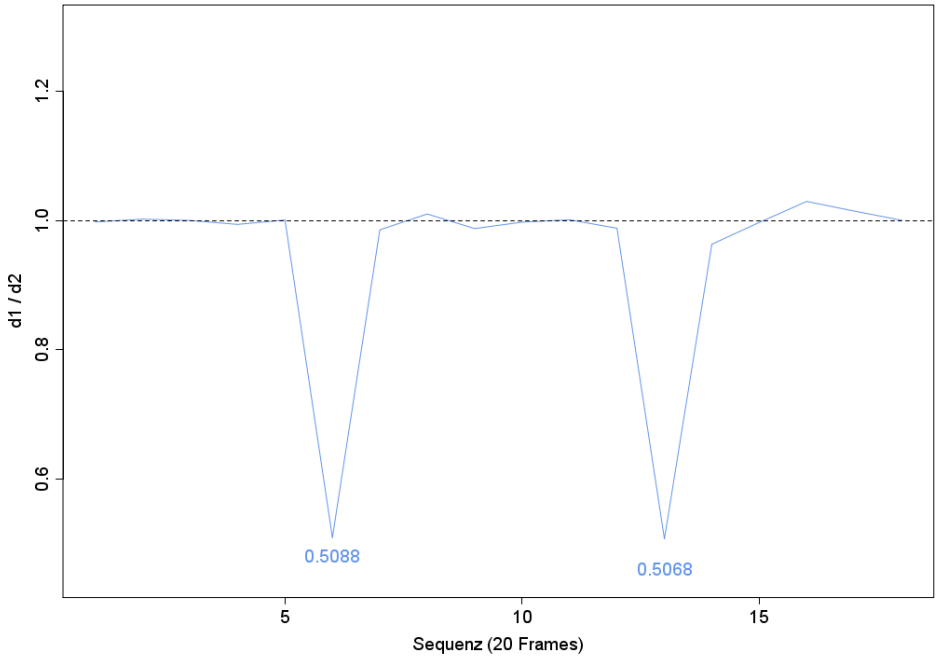


(a)

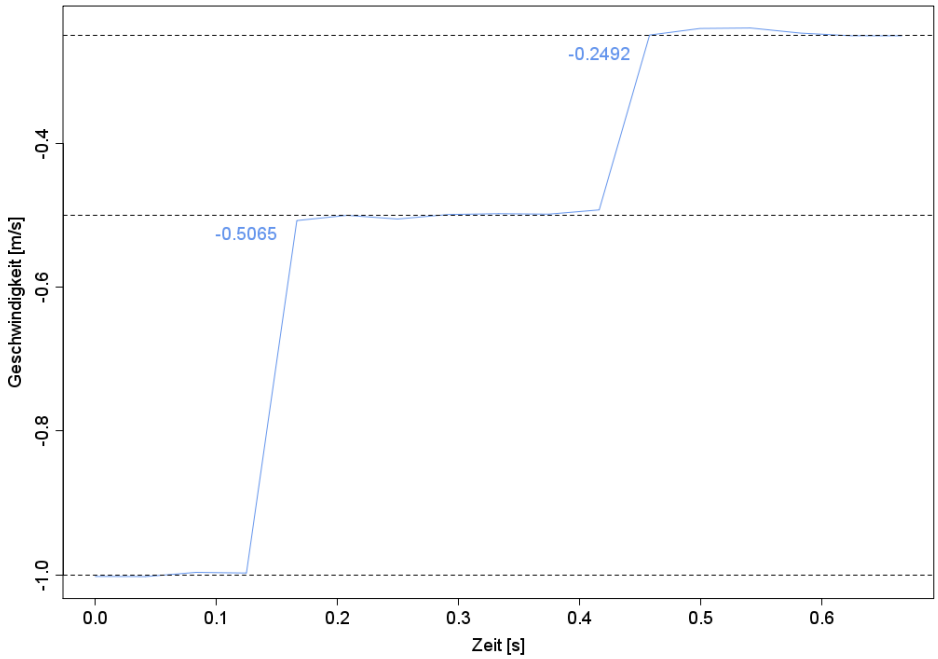


(b)

Abbildung 4.18: Strikter Translationsflug in Rückwärtsrichtung. Positive Beschleunigung.**(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

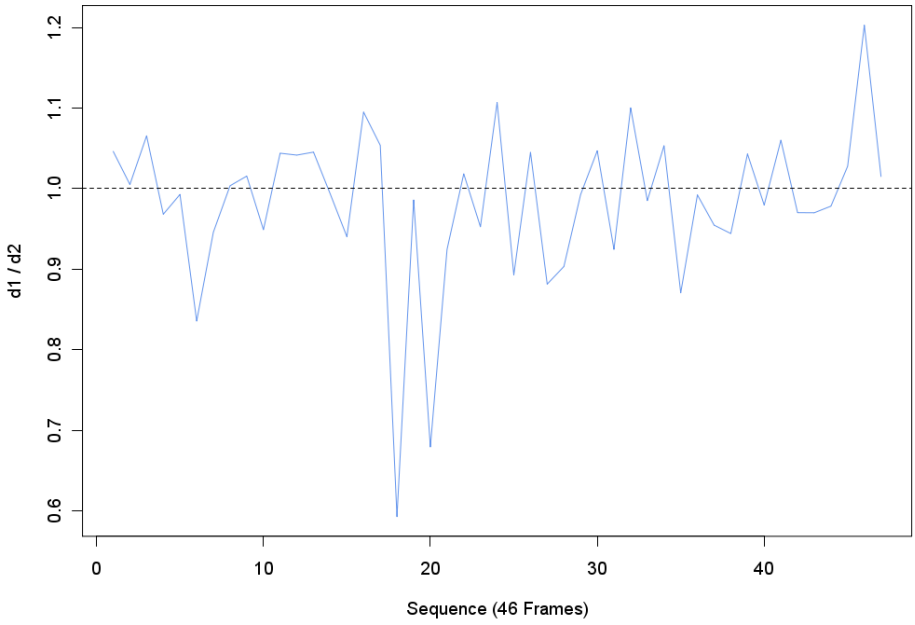


(a)

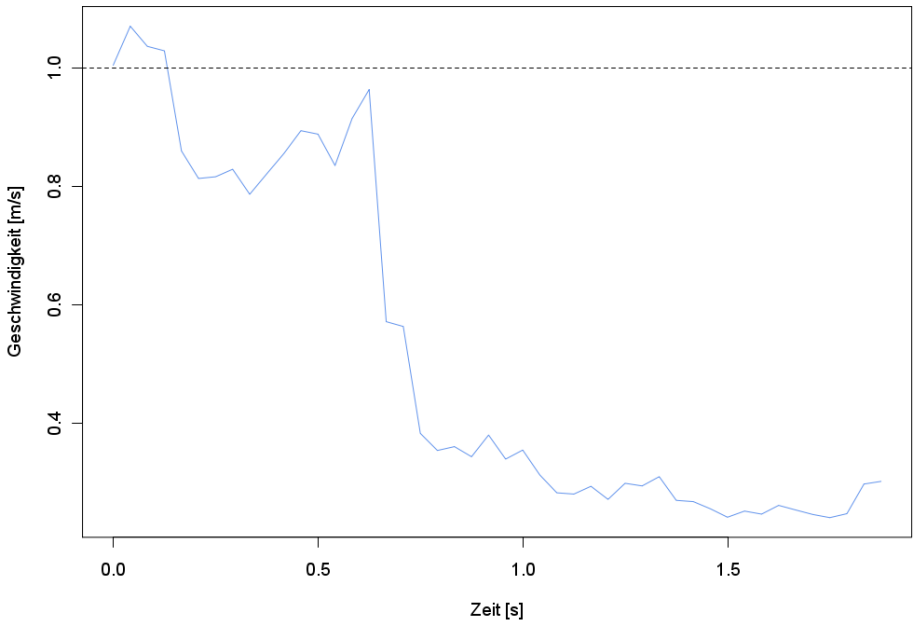


(b)

Abbildung 4.19: Strikter Translationsflug in Rückwärtsrichtung. Positive Beschleunigung.**(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

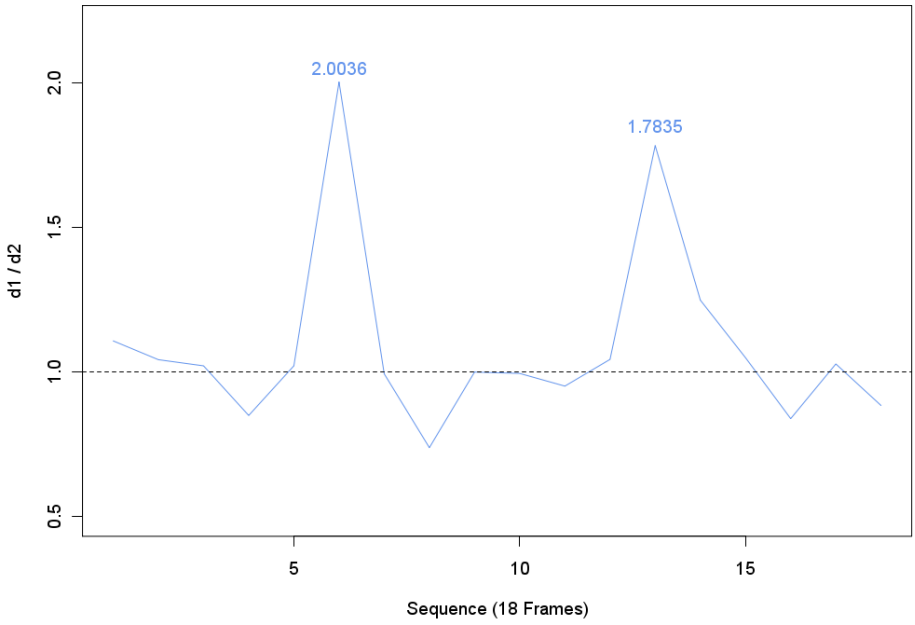


(a)

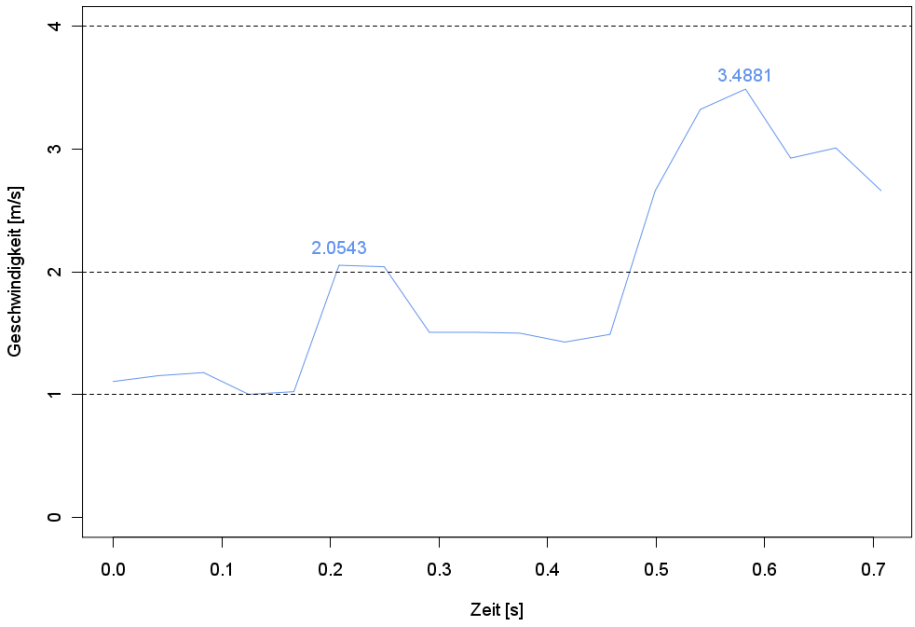


(b)

Abbildung 4.20: Strikter Translationsflug in Seitwärtsrichtung ohne Beschleunigung.**(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

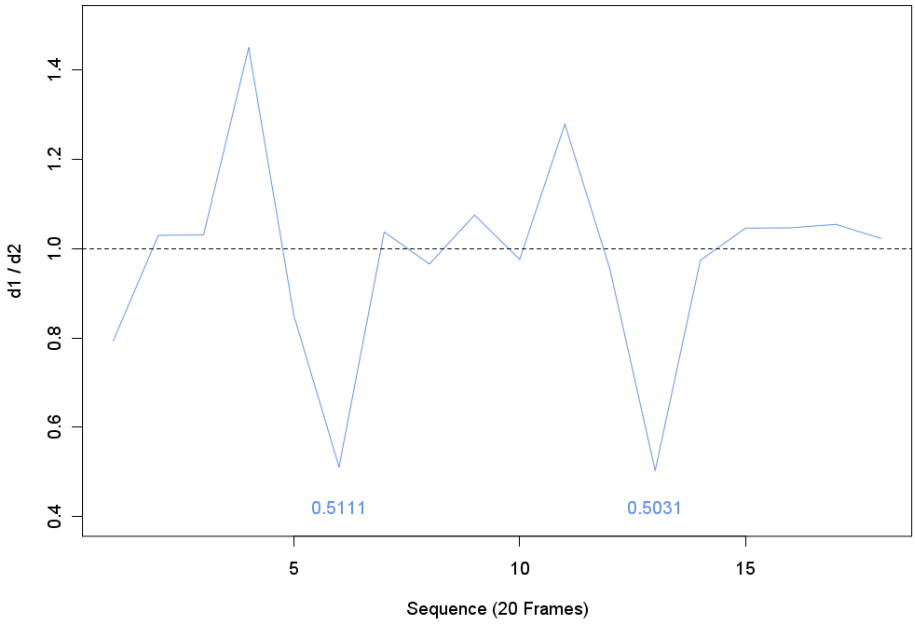


(a)

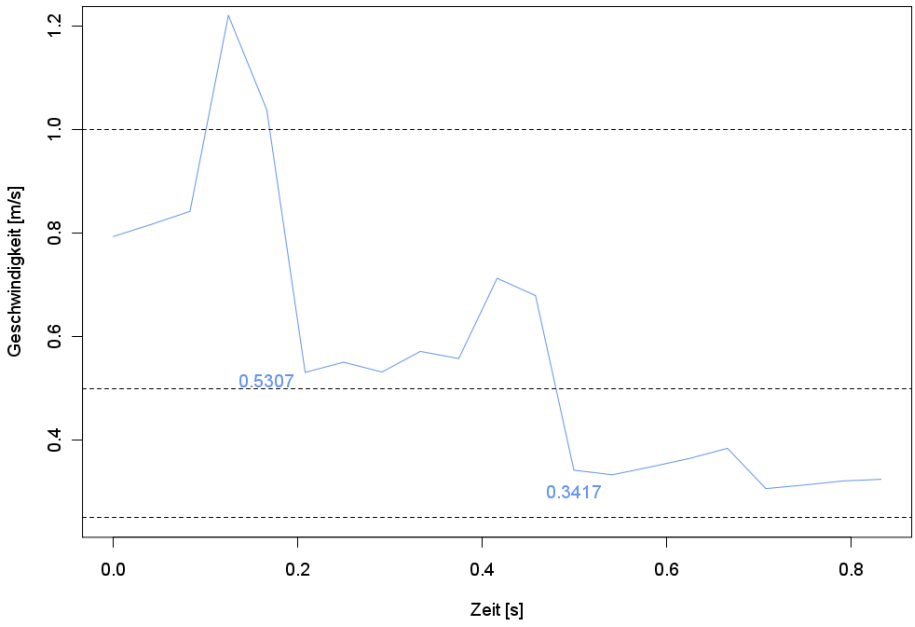


(b)

Abbildung 4.21: Strikter Translationsflug in Seitwärtsrichtung. Positive Beschleunigung.**(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.



(a)



(b)

Abbildung 4.22: Strikter Translationsflug in Seitwärtsrichtung. Negative Beschleunigung.**(a)** Beschleunigung. **(b)** Geschwindigkeitskurve.

4.2.2 Freiflugaufnahmen im *Tracking Lab*

Es folgen die Ergebnisse der Freiflüge, welche mit Hilfe des *Tracking System* protokolliert wurden. Für einen Vergleich wurde aus den protokollierten Positionsdaten eine Geschwindigkeitskurve berechnet. Die Aktualisierungsrate des *Tracking System* beträgt hierbei 60 Hz. Für die Erstellung der Geschwindigkeitskurven entsprechend der Eigengeschwindigkeitsschätzung wurde die Bildwiederholungsrate der *FinePix S5500* von 30 FPS berücksichtig. Die Anfangsgeschwindigkeit wurde auf $1\frac{m}{s}$ festgelegt.

Der vollzogene Bewegungsablauf der Freiflugaufnahmen ist sehr gut an den Geschwindigkeitskurven des *Tracking System* nachvollziehbar (siehe auch Anhang C.2 für nichtlogarithmische Darstellung). Exemplarisch sei hier der Steig-Sink-Flug in Abbildung 4.23 genannt. Beginnend an der Startposition am Boden, wurde die Kamera vertikal nach oben in Schulterhöhe gebracht. Dies ist repräsentiert durch die Geschwindigkeitskurve im Zeitintervall $[0, 10]$. In Schulterhöhe verharrt die Kamera einen Moment. Die Geschwindigkeit geht zurück auf Null. Der zweite Teil des Graphen zeigt das Herablassen der Kamera zurück auf den Boden. Eine erste visuelle Analyse ergab keine signifikante Übereinstimmung. Die Darstellung des vollständigen Datensatzes erfolgt im Anhang C.2.

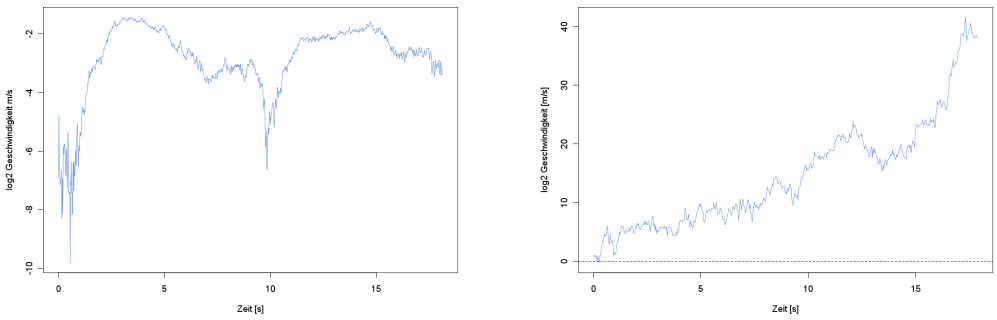


Abbildung 4.23: Steig-Sink-Flug mit Geschwindigkeitskurve laut *Tracking System* (links) und Geschwindigkeitskurve nach eigener Methode (rechts).

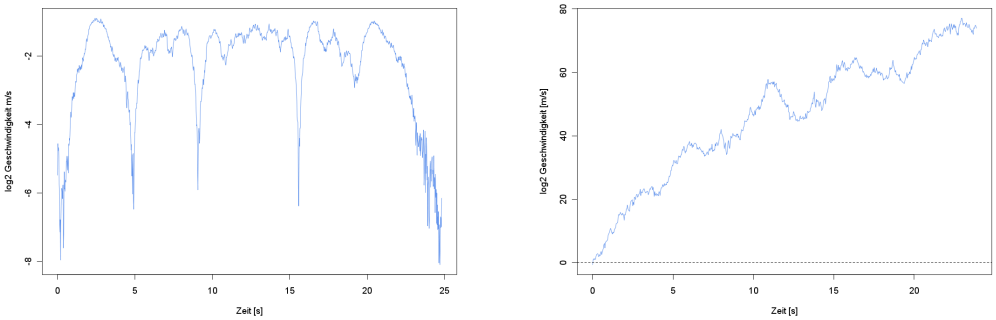


Abbildung 4.24: Links-Rechts-Flug mit Geschwindigkeitskurve laut *Tracking System* (links) und Geschwindigkeitskurve nach eigener Methode (rechts).

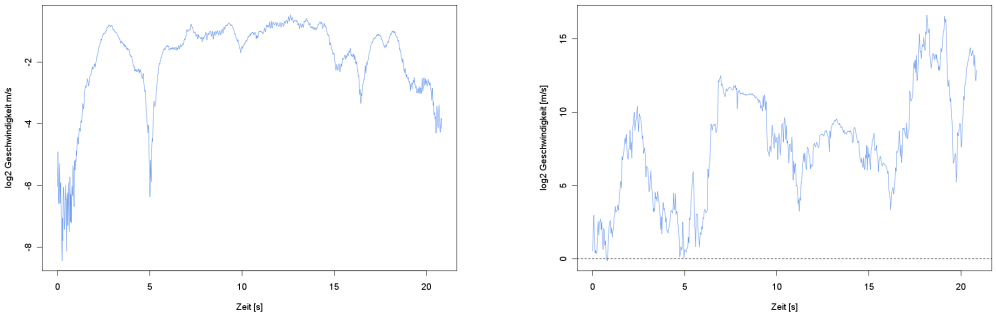


Abbildung 4.25: Kreis-Flug mit Geschwindigkeitskurve laut *Tracking System* (links) und Geschwindigkeitskurve nach eigener Methode (rechts).

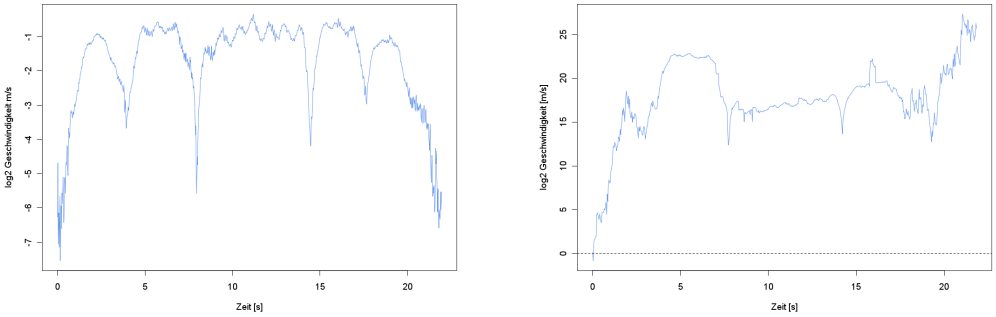


Abbildung 4.26: Vor-Zurück-Flug mit Geschwindigkeitskurve laut *Tracking System* (links) und Geschwindigkeitskurve nach eigener Methode (rechts).

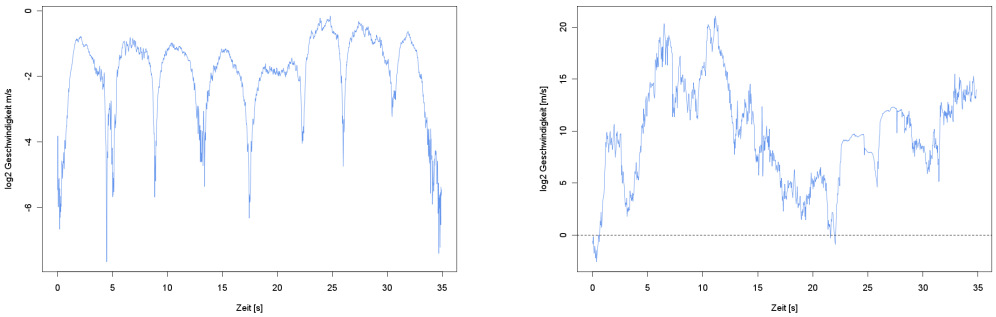


Abbildung 4.27: Kombinations-Flug mit Geschwindigkeitskurve laut *Tracking System* (links) und Geschwindigkeitskurve nach eigener Methode (rechts).

4.3 FOV-Abhängigkeit

Im Folgenden wurde die Genauigkeit der Eigengeschwindigkeitsschätzung in Abhängigkeit des Sichtfeldes untersucht. Hierzu wurden zwei Trajektorien - der Seitwärtsflug und Vorwärtsflug in der *Columns-Map* - hinsichtlich ihres Sichtfeldes weiter differenziert. Die Trajektorie ist in jedem Fall identisch und simuliert eine konstante Geschwindigkeit. Nur der horizontale Öffnungswinkel wurde in einem Intervall von 40° bis 160° verändert. Jede generierte Sequenz besteht aus 200 *Frames*. Die Initialgeschwindigkeit wurde auf $1\frac{m}{s}$ festgelegt. Die Bildwiederholungsrate beträgt 21 FPS.

Die Abbildungen 4.28 bis 4.30 zeigen die erhaltenen Ergebnisse. Es ist zu erkennen, dass mit zunehmenden Öffnungswinkel die Geschwindigkeitsschätzungen stabiler werden. Dieser Trend ist in Abbildung 4.31 noch einmal veranschaulicht. Hier wurde für jeden FOV-Wert der Median der jeweils 198 Geschwindigkeitsschätzungen eingezeichnet. Ebenfalls ersichtlich ist die kontinuierliche Verzerrung der Geschwindigkeitswerte um einen positiven Fehler. Die Trajektorien stellen eine gleichmäßige Bewegung dar. Man erwartet daher eine Beschleunigung von Null. Wie zu sehen ist, ist dies ist nicht der Fall. Alle Geschwindigkeitsquotienten sind im Mittel größer Eins. Die Plots der Geschwindigkeitsquotienten befinden sich im Anhang C.1.

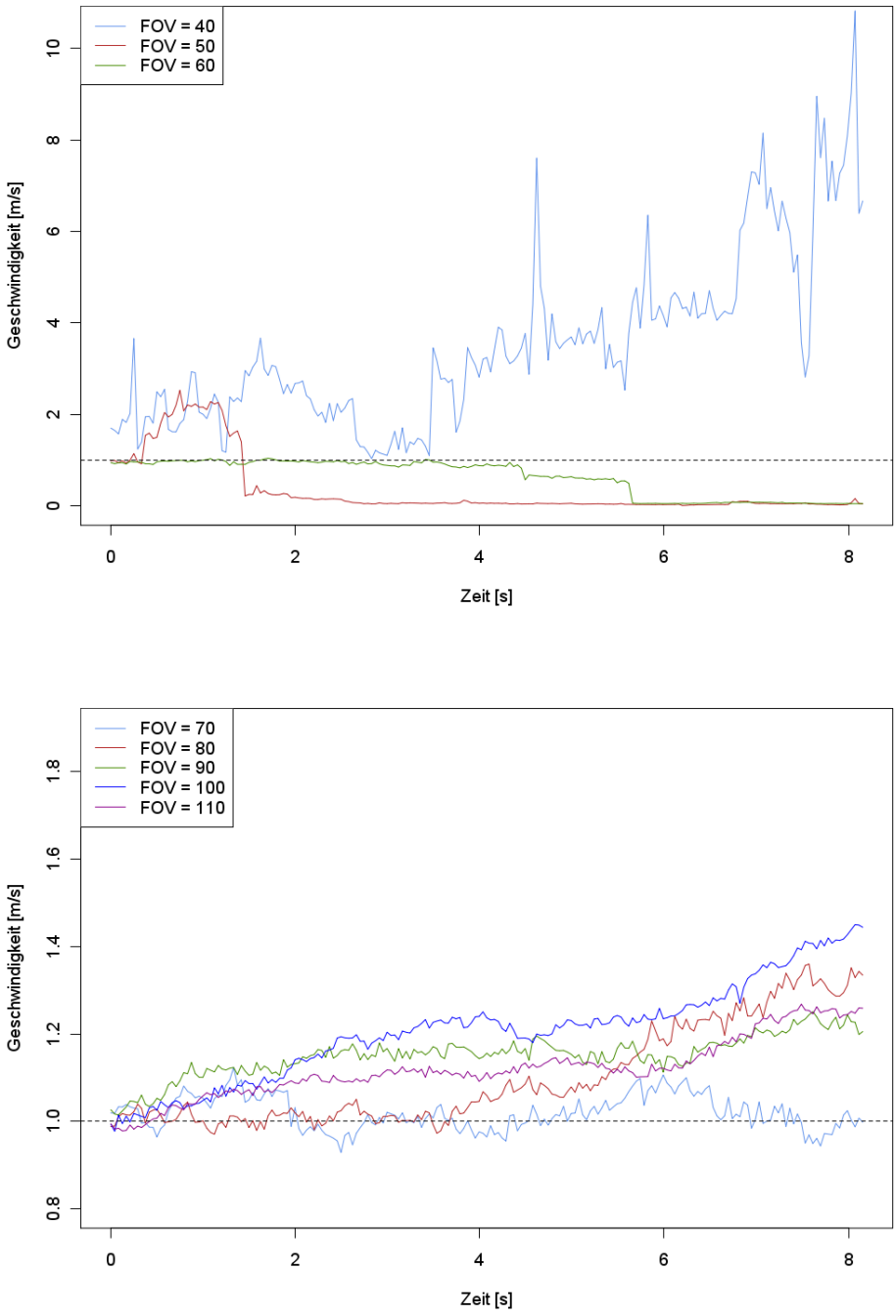
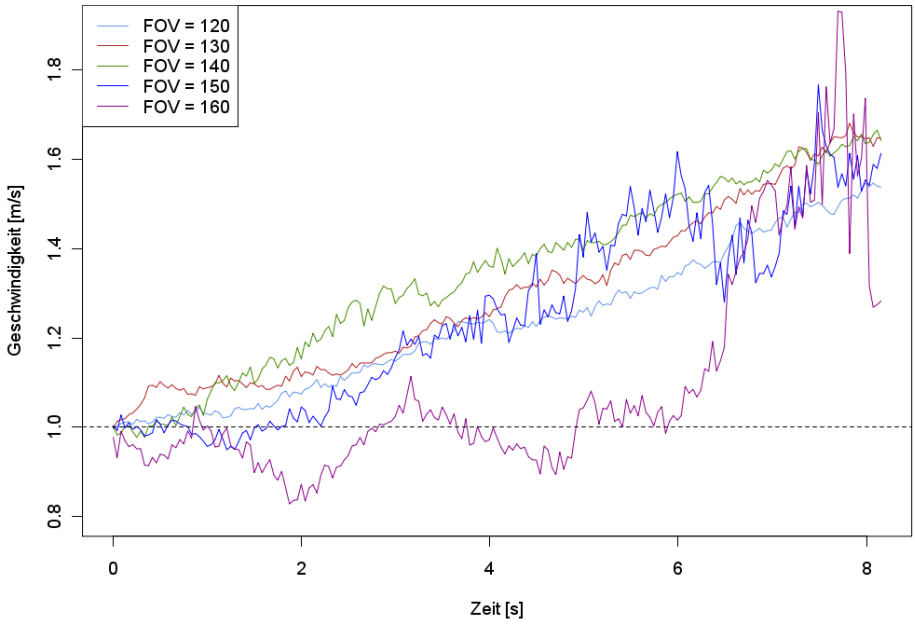
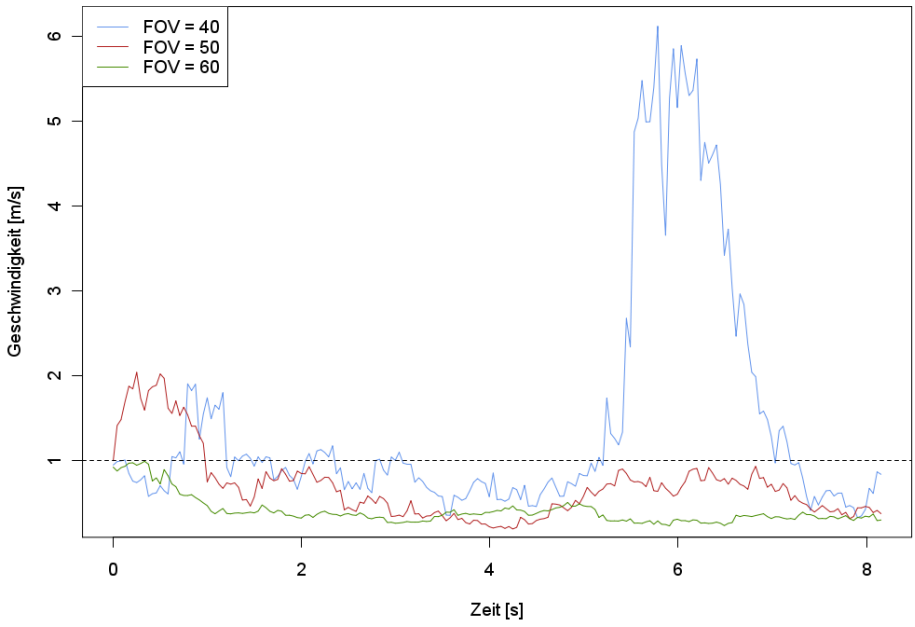


Abbildung 4.28: Geschwindigkeitskurven für verschiedene FOV-Werte eines Vorwärtsflugs ohne Beschleunigung.



(a)



(b)

Abbildung 4.29: Geschwindigkeitskurven für verschiedene FOV-Werte. **(a)** Vorwärtsflugs ohne Beschleunigung. **(b)** Seitwärtsflug ohne Beschleunigung.

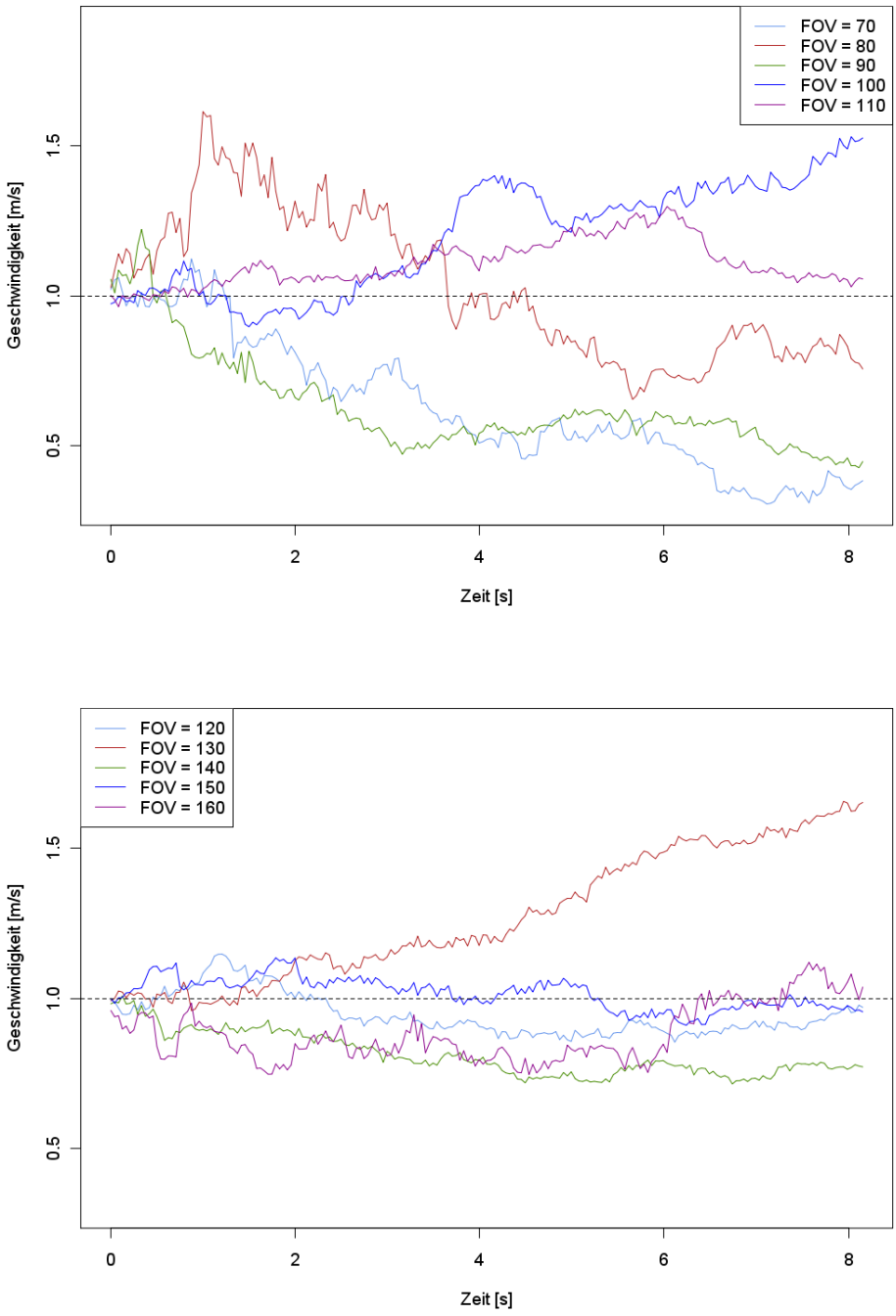


Abbildung 4.30: Geschwindigkeitskurven für verschiedene FOV-Werte eines Seitwärtsflugs ohne Beschleunigung.

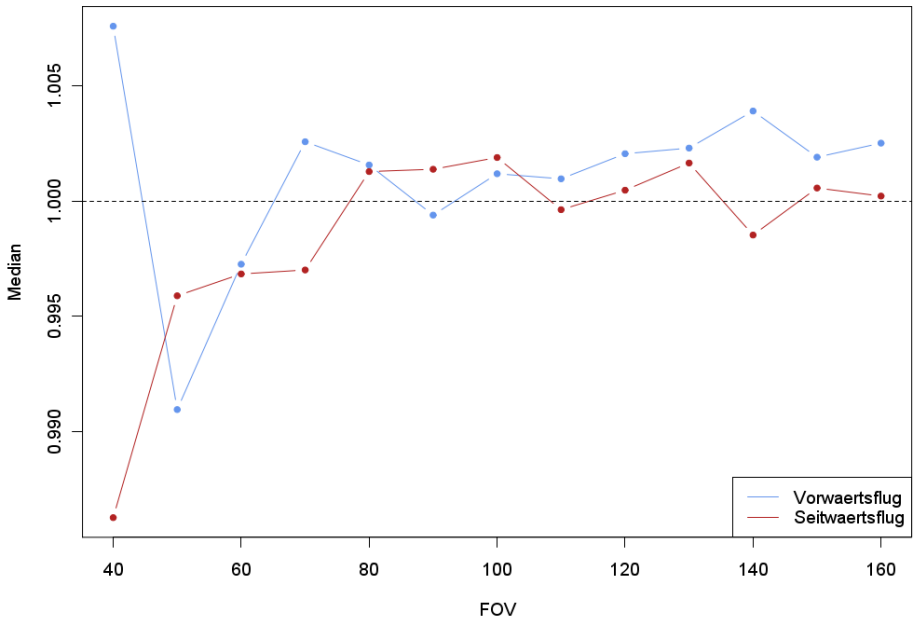


Abbildung 4.31: Die medialen Geschwindigkeitsschätzungen in Abängigkeit vom Sichtfeld. Große FOV-Werte stabilisieren die Geschwindigkeitschätzungen.

4.4 *Closed Quarter* vs. *Open Space*

Die Wahl der virtuellen Flugumgebung hatte einen leichten Einfluss auf die Genauigkeit der Berechnung der Eigengeschwindigkeit. Zusammenfassend lässt sich sagen, dass die Schätzungen der Eigengeschwindigkeit in der *Columns-Map* stärker schwanken, als der korrespondierende Flug in der *Corridor-Map*. Dieses Phänomen konnte in allen Testflügen gezeigt werden und ist in der folgenden Abbildung 4.32 an einem Testflug exemplarisch darstellt. Eine mögliche Ursache könnte die gewählte Schrittweite der Positionsänderungen sein. Da die Flüge für die *Corridor-Map* zuerst generiert wurden, wurde die Dimension in welcher der Translationsvektor verändert wurde auf eben diese angepasst. Diese Werte wurden dann ohne Modifizierung auf die *Columns-Map* übertragen. Die *Columns-Map* stellt allerdings eine weitläufige Umgebung mit großen Abständen zwischen den säulenförmigen Objekten dar. Dies hatte zur Folge, dass die hier generierten Flüsse in der Regel sehr klein waren. Befand sich zudem keine Säule in unmittelbarer Nähe, konnte nur die nähere Bodenfläche im unteren Bereich des Bildes zur erfolgreichen Berechnung von tripelkorrespondierenden nonIDMV beitragen (siehe auch Abbildung 4.33). Hier standen der Corridor-Map eindeutig mehr Informationen zur Verfügung, wo tripelkorrespondenzerfüllende nonIDMV über das ganze Bild gefunden werden

konnten (siehe Abbildung 3.6).

Es ist anzumerken, dass die resultierenden Geschwindigkeitskurven der vergleichsweise unruhigeren Kurvenverläufe der *Columns-Map* nicht in jedem Fall schlechter waren (siehe Abbildung 4.32b). Für eine genauere Untersuchung der Kurvenverläufe müssten hier längere Sequenzen miteinander verglichen werden.

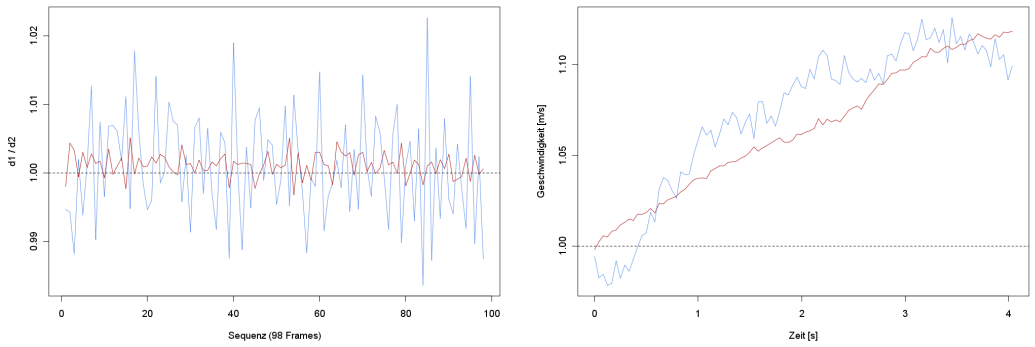


Abbildung 4.32: Strikter Translationsflug in Vorwärtsrichtung ohne Beschleunigung. Die blaue Kurve stammt aus der *Columns-Map*. Die rote Kurve entspricht der *Corridor-Map*. *Links:* Eigengeschwindigkeitsänderung. *Rechts:* Geschwindigkeitskurve.

4.5 Überschreiben der Eigenbewegungsparameter

Bei der Untersuchung der strikten Translationssequenzen (virtuell und real) wurde festgestellt, dass der Kanatani den Rotationsparameter der Eigenbewegung auf ungleich Null schätzt. Inwieweit ein manuelles Überschreiben dieses Wertes die Eigengeschwindigkeitsschätzung verbessern kann wird hier präsentiert.

Die Abbildungen 4.34 und 4.35 zeigen die Ergebnisse, die man erhält, wenn der Rotationsanteil des Kanatani-Algorithmus manuell auf Null gesetzt wird. Vergleicht man nun die in Abbildung 4.34 erhaltenen Daten mit dem unmodifizierten Flug aus Abbildung 4.1 so lässt sich erkennen, dass die aufsummierte Geschwindigkeit nach 98 *Frames* um $0,01\frac{m}{s}$ erniedrigt ist. Noch deutlicher ist dieser Effekt zu erkennen, wenn man Seitwärtsflüge analysiert. Ein Vergleich der Abbildung 4.35 mit der Abbildung 4.7 zeigt einen eindeutigen Unterschied. So ist der Kurvenverlauf in der modifizierten Kanatani-Variante wesentlich ruhiger. Nach 98 *Frames* erhalten wir eine Verbesserung von $8,974\frac{m}{s}$ im Vergleich zur nichtmodifizierten Variante.

Das manuelle Überschreiben führte bei allen virtuellen Flugsequenzen zu einem

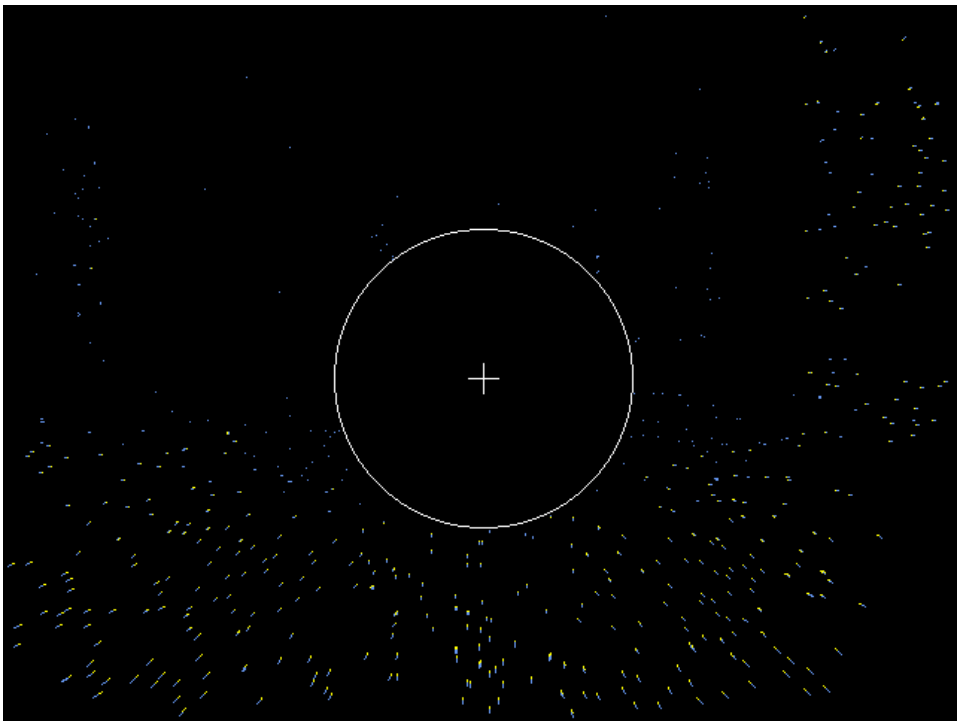


Abbildung 4.33: Günstiger Fall in der *Columns-Map*: Der Vorbeiflug an einer Säule rechts im Bild erzeugt tripelkorrespondierende nonIDMV, welche nicht von der näheren Bodenumgebung stammen.

positiven Erfolg hinsichtlich der Eigengeschwindigkeitsschätzung. Bei den realen strikten Translationsflügen konnte dieser Effekt nicht beobachtet werden. Ursache können hier die zu kurzen Testsequenzen sein, welche keinen genauen Trend erkennen lassen.

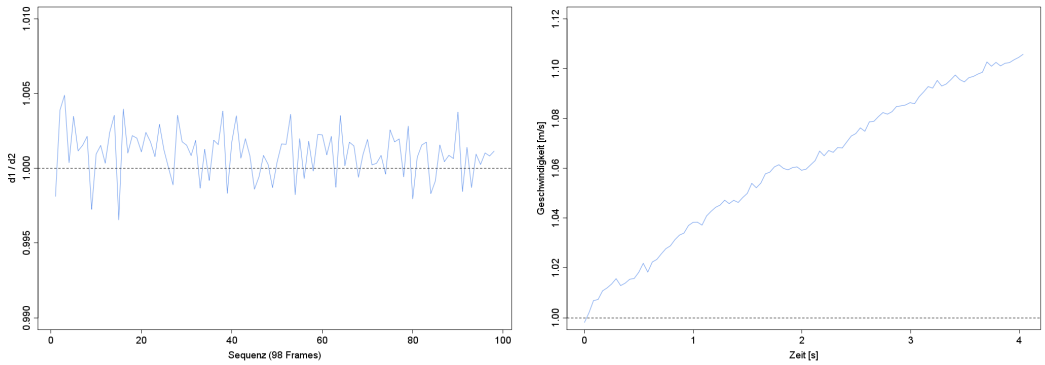


Abbildung 4.34: Strikter Translationsflug in Vorwärtsrichtung ohne Beschleunigung. Die Tiefenwerte wurden mit Hilfe der überschriebenen Rotationskomponente erneut berechnet und dann erst für die Eigengeschwindigkeitsschätzung verwendet. *Links:* Eigengeschwindigkeitsänderung. *Rechts:* Geschwindigkeitskurve.

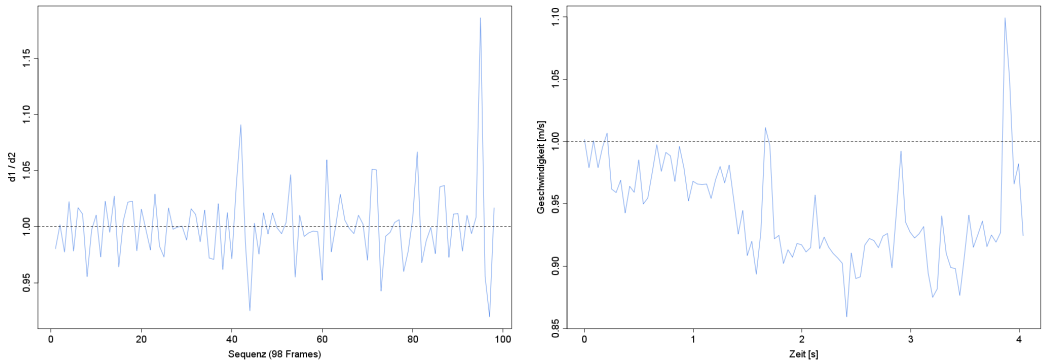


Abbildung 4.35: Strikter Translationsflug in Seitwärtsrichtung ohne Beschleunigung. Die Tiefenwerte wurden mit Hilfe der überschriebenen Rotationskomponente erneut berechnet und dann erst für die Eigengeschwindigkeitsschätzung verwendet. *Links:* Eigengeschwindigkeitsänderung. *Rechts:* Geschwindigkeitskurve.

Kapitel 5

Diskussion

5.1 Diskussion

In dieser Arbeit sollte eine Methode zur Schätzung der Eigengeschwindigkeit vorgestellt werden, die auf dem optischen Fluss der Kamerabilder basiert. Verschiedene Faktoren beeinflussen die Güte der Eigengeschwindigkeitsschätzung. Diese werden im Folgenden genannt und diskutiert.

Qualitative Flussfilterung

Für eine optimale Bestimmung von nonIDMV durch den Kanatani-Algorithmus wurden, wie in Kapitel 3 beschrieben, verschiedene qualitative Filter implementiert bzw. vorhandene Methoden optimiert. Die Anwendung dieser mehrstufigen qualitativen Filterung des optischen Flusses wurde erfolgreich demonstriert (siehe Kapitel 3 und Abbildungen 3.4 bis 3.6) und die Ergebnisse signifikant verbessert.

Kanatani-Algorithmus

Trotz der mehrstufigen qualitativen Filterung der Flussvektoren ergibt sich bereits bei den virtuellen Testsequenzen eine zu hohe Beschleunigung, was über die Zeit zu einer ansteigenden Geschwindigkeitsschätzung bei einer in Wahrheit gleichbleibenden Bewegung führt. Diesem Effekt wurde versucht entgegenzuwirken, indem die nachweislich falsch geschätzte Rotationskomponente des Kanatani aktiv überschrieben wurde (siehe Abschnitt 4.5). In den virtuellen Testsequenzen konnte damit die Eigengeschwindigkeitschätzung zwar verbessert werden, jedoch war die ausgeprägte positive Tendenz der Geschwindigkeitsschätzung immer noch vorhanden. Zudem birgt das Überschreiben der Rotationskomponente mathematische Ungenauigkeiten. Dies liegt daran, dass der Algorithmus nach Kanatani zuerst die Translationskomponente der Ei-

genbewegungsparameter bestimmt und erst danach die Rotationskomponente. Damit können durch Überschreiben der Rotationskomponente nur noch nachträglich die Tiefenwerte, der der Eigenbewegung zugeordneten Vektoren manipuliert werden. Wenn diese Menge der nonIDMV aber schon vorher aus einem falschen Modell stammt, sind die Konsequenzen nicht vorherzusehen. Soll die Eigengeschwindigkeitschätzung auf Grundlage des Kanatani-Algorithmus erfolgen, so muss geklärt werden, ob der beobachtete asymmetrische Fehlerbereich hier zu suchen ist. Die Überschreibung der Rotationskomponente bietet hierfür nur einen bedingten Ansatz.

Sehbereich

Es wurde gezeigt, dass die Erhöhung des Sehbereich einen positiven Einfluss auf die Eigengeschwindigkeitsschätzung hat (siehe Abschnitt 4.3).

Laufzeit

Die vorgestellte Implementierung benötigt zur Eigengeschwindigkeitsschätzung im Schnitt 601 Sekunden für die Bearbeitung von 200 Frames. Dies entspricht einer durchschnittlichen Bildwiederholungsrate von 0,33 FPS und ist somit nicht realzeittauglich. Die Laufzeit wird dabei unter anderem durch den in Abschnitt 3.2.1 vorgestellten Parameter zur Höchstzahl der berechnenden Flussvektoren beeinflusst. Zur Zeit können aus den maximal 4.800 berechneten Flussvektoren circa 900 tripelkorrespondierende nonIDMV (circa 1.300 für virtuelle Flugsequenzen) extrahiert werden. In Weiteren Tests könnte untersucht werden,inwieweit dieser Parameter erniedrigt oder aber auch die Qualitätskontrolle gelockert werden kann, um die Performance zu steigern. Weitere die Performance negativ beeinflussende Faktoren sind die in jedem Iterationsschritt aufgerufene Sortierfunktion für die Ermittlung des Medians, sowie die benötigte dritte Flussberechnung für die geforderte Tripelkorrespondenz. Versuchsweise wurde hier eine alternative Methode zur Bildpunktverfolgung getestet. Diese berechnet den optischen Fluss ausgehend von der Kameraposition C_{i+1} vorwärts nach C_{i+2} und rückwärts nach C_i und macht theoretisch eine Kamerakorrektur unnötig. Auch die zusätzliche Flussberechnung zwischen C_i und C_{i+2} entfällt. Es musste hier allerdings festgestellt werden, dass zusätzlich zum systematischen Fehler, alle Werte konstant in den positiven bzw. negativen Bereich verschoben waren (siehe Anhang C.3). Dieser Effekt war bekannt. Er trat auch bei der Methode der Tripelkorrespondenz auf, als die benötigte Kamerakorrektur noch nicht durchgeführt wurde (siehe Abschnitt 3.1.1). Eine Modifizierung der Tiefenwerte durch die Kamerakorrektur erbrachte keinen Erfolg.

5.2 Fazit

Die in der Einleitung gestellte Frage, ob eine Drohne autonom navigieren kann indem sie sich allein auf Informationen des optischen Flusses aus Kamerabildern verlässt, muss zum Ende dieser Arbeit mit nein beantwortet werden. Es wurden verschiedene Probleme, die bei diesem Projekt auftraten, aufgezeigt und Lösungen diskutiert. Einige Probleme, wie eine zu hohe Beschleunigungsschätzung des Kanatani-Algorithmus und die mangelnde Realzeittauglichkeit des Verfahrens, konnten jedoch nicht gelöst werden.

Literaturverzeichnis

[1] N. Ancona, M. Romano: *A Time-to-Crash Detector Based on Area Expansions: Example of an Opto-Motor Reflex.* Springer, Intelligent Perceptual Systems: New Directions in Computational Perception, pp. 354-367, (1993)

[2] N. Ancona, T. Poggio: *Optical Flow from 1-D Correlation: Application to a Simple Time-to-Crash Detector.* Artificial Intelligence Laboratory Memo No. 1375 and Center for Biological and Computational Learning Paper No. 74, (1993)

[3] J. L. Barron, D. J. Fleet, S. S. Beauchemin, and T. A. Burkitt: *Performance of optical flow techniques.* International Journal of Computer Vision, Vol. 12, No. 1, pp. 43-77, (1994).

[4] J. L. Barron, N. A. Thacker: *Tutorial: computing 2D and 3D Optical Flow.* Imaging Science and Biomedical Engineering Division, Medical School, University of Manchester, TINA Memo No. 2004-012, (2005)

[5] J. Y. Bouguet: *Pyramidal Implementation of the Lucas Kanade Feature Tracker.* Intel Corporation, Microprocessor Research Labs, (2000).

[6] N. H. Cornelius, Takeo Kanade: *Adapting Optical-Flow to Measure Object Motion in Reflectance and X-Ray Image Sequences.* IProc. of ACM SIGGRAPH/SIGART Workshop on Motion: Representation and Perception,pp. 50-58, (1983).

[7] O. D. Faugeras: *What can be seen in three dimensions with an uncalibrated stereo rig?* Proceedings of the Second European Conference on Computer Vision, pp. 563-578, Springer-Verlag, (1992).

[8] Martin A. Fischler and Robert C. Bolles: *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography.* Communications of the ACM, Volume 24, No. 6, pp. 381-395, (1981)

[9] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills: *Recovering motion fields: An evaluation of eight optical flow algorithms.* In Proceedings of the Ninth British Machine Vision Conference (BMVC '98), volume 1, pp. 195-204, Southampton, UK, (1998). http://citeseer.ist.psu.edu/galvin98recovering.html (Zugriff: 08.12.2009)

[10] J. J. Gibson: *The ecological approach to visual perception.* Boston: Houghton-Mifflin, (1979).

[11] J. J. Gibson: *The perception of the visual world.* Boston: Houghton-Mifflin, (1950).

[12] Richard Hartley, Andrew Zisserman: *Multiple View Geometry in Computer Vision.* Cambridge University Press,2th ed., (2003). ISBN: 0521540518

[13] Berthold K. P. Horn, Brian G. Schunck: *Determining Optical Flow.* Artificial Intelligence, Volume 17, pp. 185-203, (1981).

[14] Berthold K. P. Horn: *Tsai's Camera Calibration Method Revisited.* http://people.csail.mit.edu/bkph/articles/Tsai_Revisited.pdf (Zugriff: 08.12.2009)

[15] Bernd Jähne: *Digitale Bildverarbeitung.* Springer, Berlin; 6. überarbeitete und erweiterte Auflage, (2005). ISBN: 3540249990

[16] Kenichi Kanatani: *3-D interpretation of optical flow by renormalization.* International Journal of Computer Vision,Volume 11, Issue 3, pp. 267-282, (1993)

[17] E. R. Kandel, J. H. Schwartz, and T. M. Jessell: *Principles of Neural Science.* McGraw-Hill, 4th ed., (2000).

[18] J. Kittler, J. Illingworth: *Minimum error thresholding.* Elsevier Science Inc., Pattern Recognition, Vol. 19, No. 1, pp. 41-47, (1986).

[19] Holger G. Krapp, Roland Hengstenberg: *Estimation of self-motion by optic flow processing in single visual interneurons* Letters to Nature, Nature 384, pp. 463-466, (1996).

[20] M. Lappe et al.: *Optic Flow Processing in Monkey STS: A Theoretical and Experimental Approach.* Journal of Neuroscience, Volume 16, Number 19, pp. 6265-6285, (1996).

[21] D. N. Lee, H. Kalmus: *The Optic Flow Field: The Foundation of Vision.* Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences, Vol. 290, No. 1038,pp. 169-179, (1980).

[22] R. K. Lenz, R. Y. Tsai: *Techniques for Calibration of the Scale Factor and Image Center for high accuracy 3D Machine Vision Metrology.* IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 10, No.5, pp.713-720, (1987).

[23] Q. Luong and O. Faugeras: *The Fundamental Matrix: Theory, Algorithms, and Stability Analysis.* to appear IJCV, (1995). http://citeseer.ist.psu.edu/luong95fundamental.html (Zugriff: 08.12.2009)

[24] Shahriar Negahdaripour, Michael A. Gennert: *Relaxing the Brightness Constancy Assumption in Computing Optical Flow.* Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Memo No. 975, (1987).

[25] Shahriar Negahdaripour: *Revised Definition of Optical Flow: Integration of Radiometric and Geometric Cues for Dynamic Scene Analysis.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 20, Issue 9, pp. 961-979, (1998).

[26] Fabian Recktenwald: Dissertation in Bearbeitung, Universität Tübingen, Lehrstuhl für Kognitive Neurowissenschaften.

[27] Eero P. Simoncelli: *Bayesian multi–scale differential optical flow.* In Haussecker Jahne and Geissler, editors, Handbook of computer vision and applications. Academic Press, (1998).

[28] E. P. Simoncelli: *Design of multi-dimensional derivative filters.* First Int'l Conf on Image Proc, Austin Texas, volume I, pp. 790–793, (1994)

[29] M. Srinivasan et al.: *Honeybee navigation en route to the goal: visual flight control and odometry.* Journal of Experimental Biology, Vol 199, Issue 1, pp. 237-244, (1996).

[30] R. Y. Tsai: *A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses.* IEEE Journal of Robotics and Automation, Vol. 3, No. 4, pp. 323-344, (1987).

[31] William H. Warren Jr. et al: *Optic flow is used to control human walking.*Nature Neuroscience 4, 213 - 216, (2001).

[32] AirRobot GmbH & Co. KG: *AirRobot Bedienungsanleitung.* Mai 2007. http://www.airrobot.de (Zugriff: 08.12.2009)

Anhang A

Spezifikation des AirRobot AR-100

Es folgt eine Auflistung der wesentlichen Merkmale des AirRobot AR-100 nach [32].

Technische Daten:

- Vier bürsten- und getriebelose E-Antriebe (2.000 rpm)
- barometrische Höhenkontrolle
- Kreisel- und Beschleunigungssensoren zur Lageregelung
- Tageslicht Color-Sensor (470 TV Zeilen)
- 2,3 - 2,5 GHz Security Videolink (200mW, 16 Kanäle)
- 4-fach Diversity-Empfänger (2,3-2,5 GHz)
- Flugkontrolle über Kommandogerät, Laptop oder Videobrille
- LiPo-Akku (14,8 V, 2A)

Gewicht:	< 1 kg
Größe:	ø 1 m
Flugdauer:	>20 min
Nutzlasten:	200 g
Reichweite:	500 m
Windlast:	bis 5 m/s
Vmax horizontal:	10 m/s
Vmax vertikal:	2 m/s

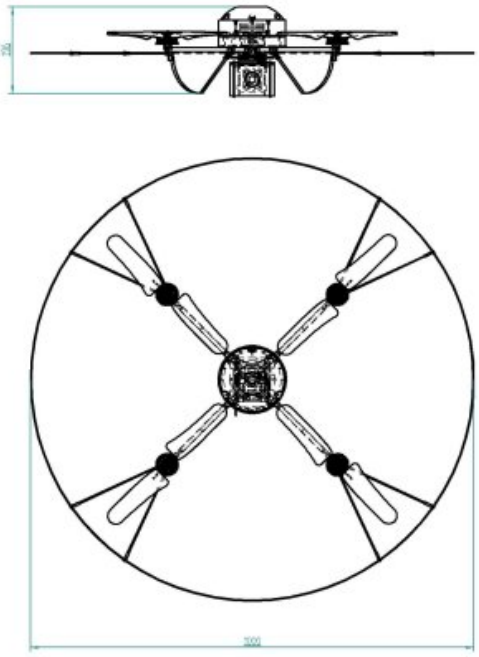


Abbildung A.1: Technische Skizze des AR-100.

Anhang B

Parametertest

B.1 Der minimale Eigenwert

Hinsichtlich der qualitativen Flussfilterung erfolgte ein Test über das Verhalten des *minEig*-Parameters. Die Motivation entstand aus der Implementierung der flussvektorgestützten Filterung (siehe Abschnitt 3.2.2) heraus. Zuvor erfolgte eine Filterung der Flussvektoren nur auf Basis der Strukturiertheit einer Pixelumgebung. Ein hoher *minEig*-Parameter lieferte gute Ergebnisse hinsichtlich der qualitativen Filterung. Jedoch bestand das Problem, dass mit wachsendem Betrag das Flussfeld zunehmend „löchrig" wurde.

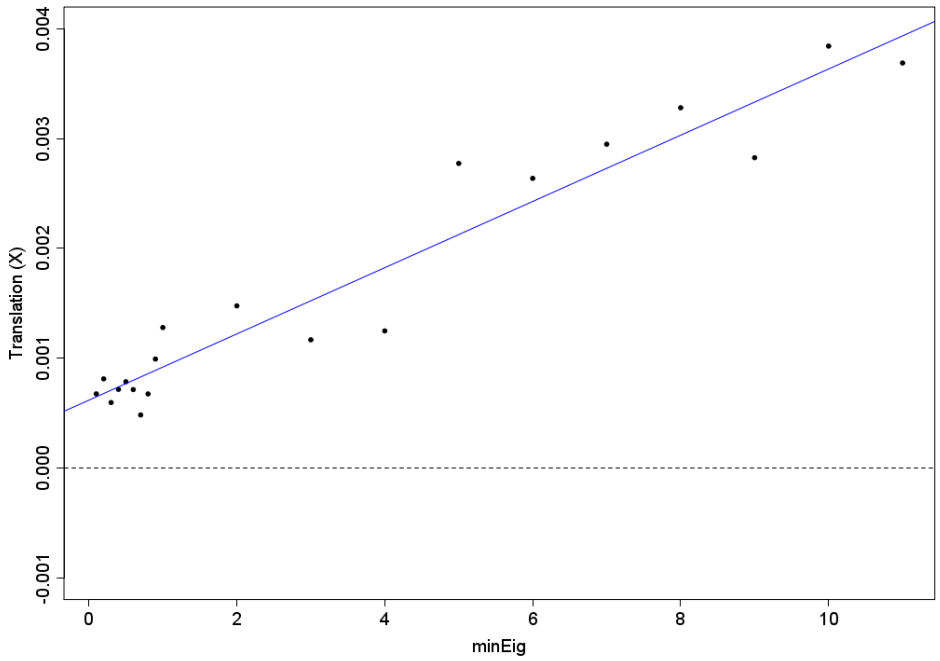
Im kombinierten Einsatz mit der flussvektorgestützten Filterung sollte nun beobachtet werden, wie sich verschiedene *minEig*-Werte auf die Eigenbewegungsschätzung nach Kanatani auswirken. Das Ergebnis ist in Abbildung B.1 an einem Beispiel tabellarisch dargestellt. Jeder Eintrag in der Tabelle entspricht hierbei dem Median aus jeweils 100 Datenwerten, welche aus einem Bildpaar einer virtuellen Flugsequenz (*Corridor-Map* im Abschnitt 3.3) extrahiert wurden. Eine graphische Aufarbeitung der Messdaten in den Abbildungen B.2 und B.3 zeigt, dass eine Verringerung des *minEig*-Parameters die Schätzung der Eigenbewegungsparameter positiv beeinflusst. Aus diesem Grund wurde für diese Arbeit der *minEig*-Parameter von 1.0 auf 0.1 herabgesetzt.

Das Ergebnis wurde an einer Stichprobe von insgesamt 4 Bildpaaren mit unterschiedlichen Geschwindigkeiten in beiden virtuellen Flugumgebungen (*Corridor-Map*, *Columns-Map*) bestätigt. Die verwendeten Flugsequenzen entsprachen jeweils einem Translationsflug entlang einer Achse ohne jeglichen Rotationsanteil. Die Ausrichtung der Kamera war nach vorn in Flugrichtung und der ideale Expansionspunkt (siehe Abschnitt 3.2.4) befand sich im Bildmittelpunkt. Diese elementare Flugbewegung in einer wohlstrukturierten virtuellen Umgebung ist für den Kanatani unproblematisch und liefert die genauesten Ergebnisse. Ein ausführlicher Parametertest war innerhalb dieser Diplomar-

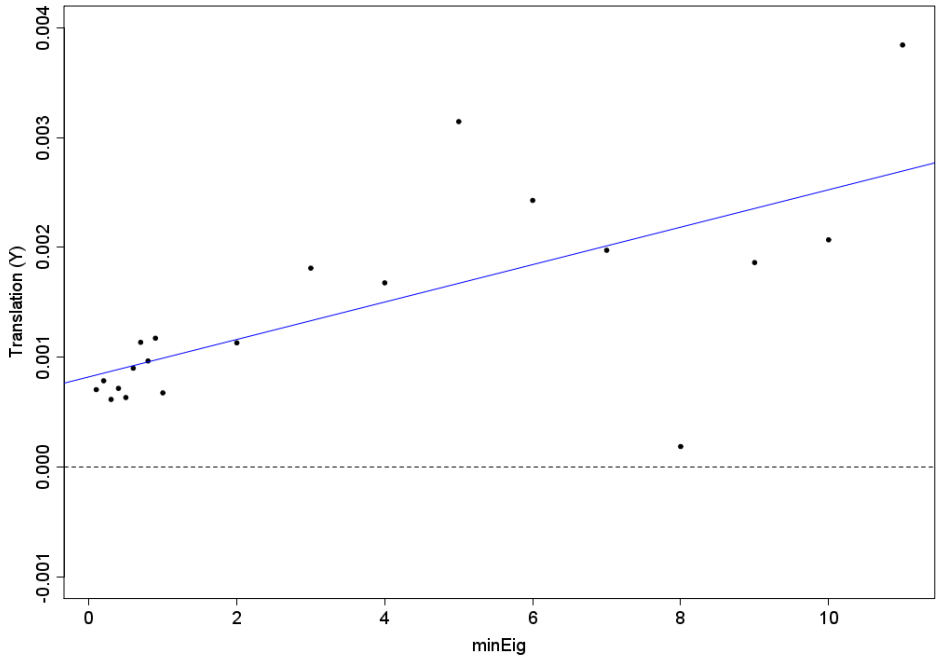
beit nicht möglich.

minEig	**Translation**			**Rotation**
	X	Y	Z	Angle
0.1	0.0006739	0.0007038	0.9999980	0.011758
0.2	0.0008112	0.0007845	0.9999977	0.012261
0.3	0.0005955	0.0006146	0.9999979	0.012124
0.4	0.0007156	0.0007154	0.9999980	0.011319
0.5	0.0007847	0.0006319	0.9999977	0.011413
0.6	0.0007133	0.0008990	0.9999972	0.012213
0.7	0.0006743	0.0011350	0.9999969	0.014197
0.8	0.0004823	0.0009651	0.9999978	0.013148
0.9	0.0009918	0.0011726	0.9999971	0.013914
1.0	0.0012791	0.0006742	0.9999971	0.012598
2.0	0.0014767	0.0011305	0.9999960	0.015381
3.0	0.0011672	0.0018110	0.9999948	0.017505
4.0	0.0012483	0.0016770	0.9999954	0.017667
5.0	0.0027745	0.0031466	0.9999893	0.025000
6.0	0.0026385	0.0024286	0.9999924	0.023830
7.0	0.0029500	0.0019740	0.9999928	0.023130
8.0	0.0032820	0.0001853	0.9999938	0.019330
9.0	0.0028270	0.0018621	0.9999943	0.020960
10.0	0.0038440	0.0020691	0.9999901	0.022170
11.0	0.0036900	0.0038450	0.9999860	0.026360
actual	0	0	1	0

Abbildung B.1: Eigengeschwindigkeitsschätzung eines Bildpaares aus der *Corridor-Map* in Abhängigkeit des minEig-Parameters. Die hinsichtlich der tatsächlichen Bewegung erwarteten Eigenbewegungsparameter sind in der letzten Zeile angegeben. Der Farbverlauf (grün, gelb, orange, rot) verdeutlicht das Ausmaß der Abweichung des jeweiligen Parameters vom erwarteten Wert.

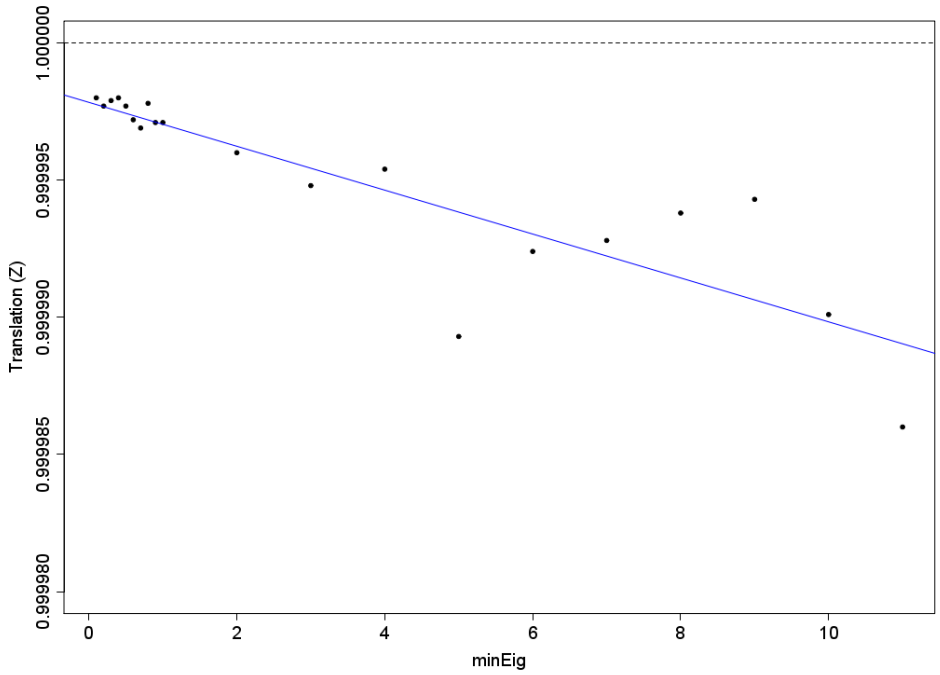


(a)

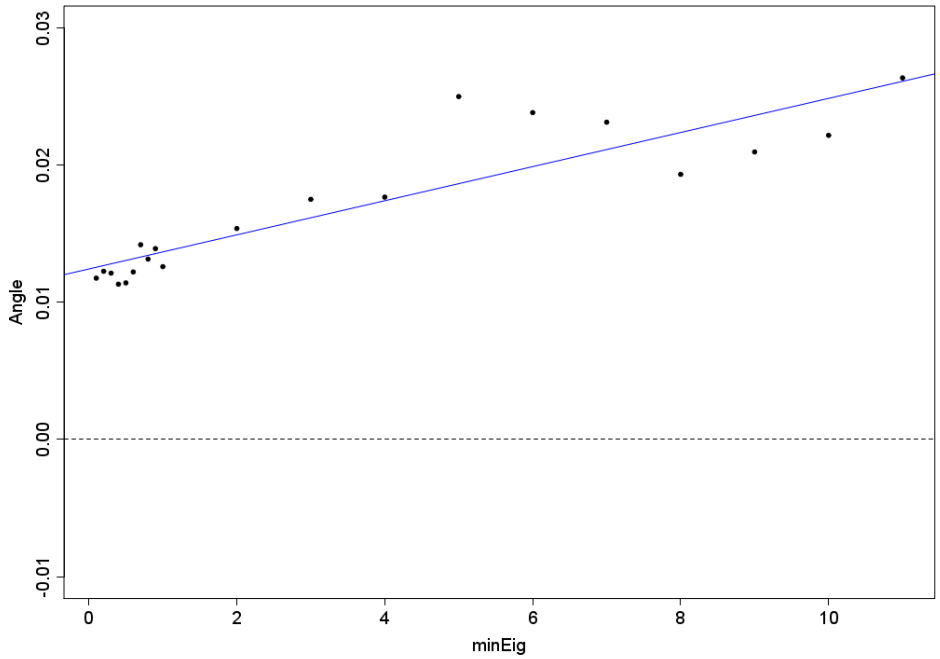


(b)

Abbildung B.2: Die gestrichelte Linie entspricht dem erwarteten Wert, die blaue Linie beschreibt die Ausgleichsgerade durch die Messdaten der jeweils betrachteten Eigenbewegungskomponente.



(a)



(b)

Abbildung B.3: Die gestrichelte Linie entspricht dem erwarteten Wert, die blaue Linie beschreibt die Ausgleichsgerade durch die Messdaten der jeweils betrachteten Eigenbewegungskomponente.

B.2 Parameter der flussvektorgestützten Filterung

Um eine Aussage über die Wahl der beiden Parameter treffen zu können, wurden beide anhand eines Bildpaares einer Flugsequenz aus der *Corridor-Map* varriert und jeweils die Anzahl der nichtverwertbaren Flussvektoren ermittelt. Die Abbildung B.4 zeigt die Abhängigkeit der Anzahl der verworfenen Vektoren von den beiden Filterparametern. Der zulässige Bereich des ersten Filterparameters *AngDev* bewegt sich von 0 bis 5. Er beschreibt die zulässige Winkelabweichung zwischen Vorwärts- und Rückwärtsschätzung. Der zweite Parameter *PixGap* wurde im Bereich 0 bis 2 getestet. Er bezieht sich auf die maximal zulässige Länge des Differenzvektors der beiden Schätzungen.

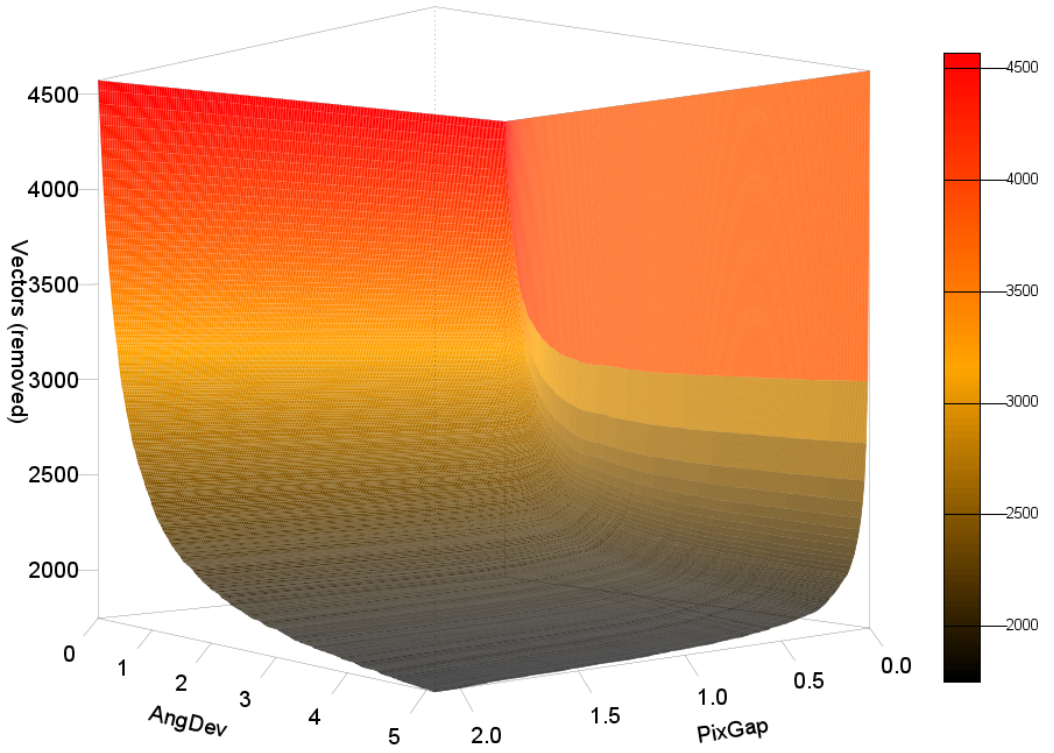


Abbildung B.4: Parametertest der flussvektorgestützten Filterung.

Es ist ersichtlich, dass eine perfekt zueinander passende Vorwärts- und Rückwärtsschätzung eines Flussvektors nicht existiert. Für die Intervalluntergrenzen der beiden Parameter werden demnach alle 4.570 berechneten Flussvektoren verworfen. Je weiter man in den Intervallen nach oben voranschreitet, umso toleranter erfolgt die Filterung. Der Graph erreicht sein Minimum bei den Intervallobergrenzen. An dieser Stelle werden nur noch 1748 Vektoren verworfen. Des Weiteren läßt sich erkennen, dass bei fest gewählter Winkelabweichung,

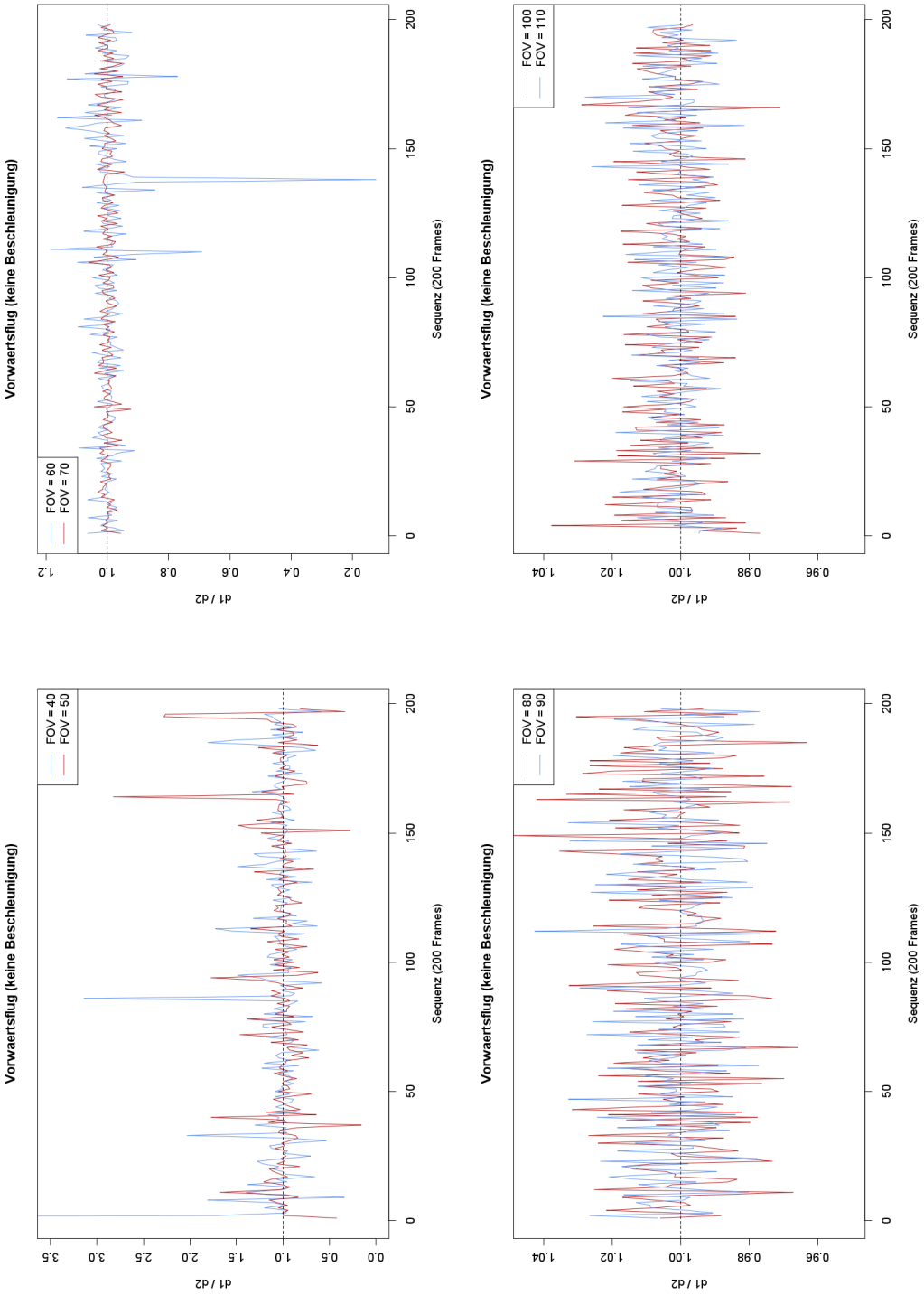
der Pixelabstand nur im Intervall $[0.01, 0.3]$ einen erkennbaren Einfluss auf die Anzahl verworfener Vektoren hat. Unter Berücksichtigung der für die Diplomarbeit verwendeten Filmsequenzen und der dabei auftretetenden Flusslängen wurde die Winkelabweichung auf 3° und der maximale Pixelabstand auf 0.1 festgelegt.

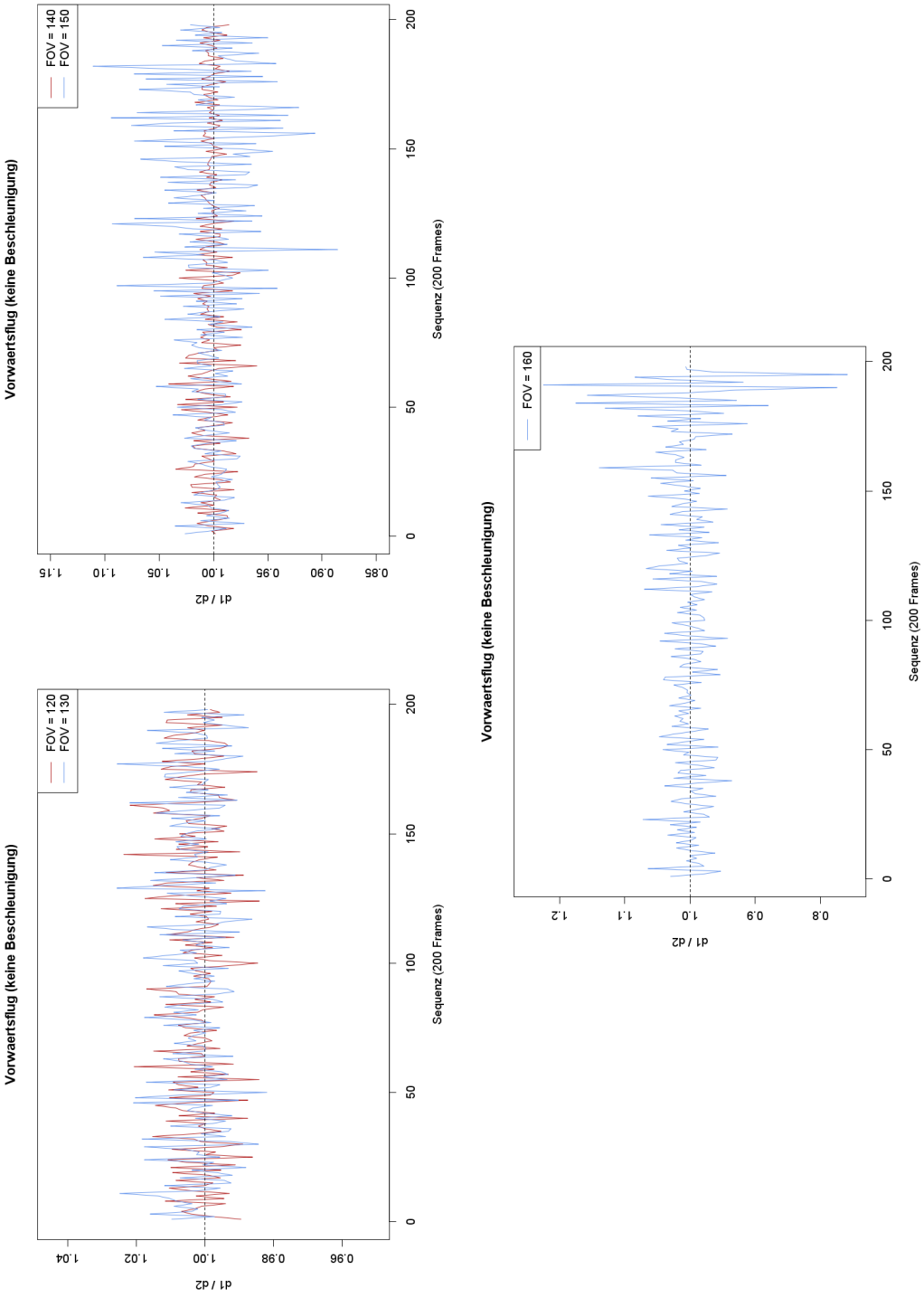
Anhang C

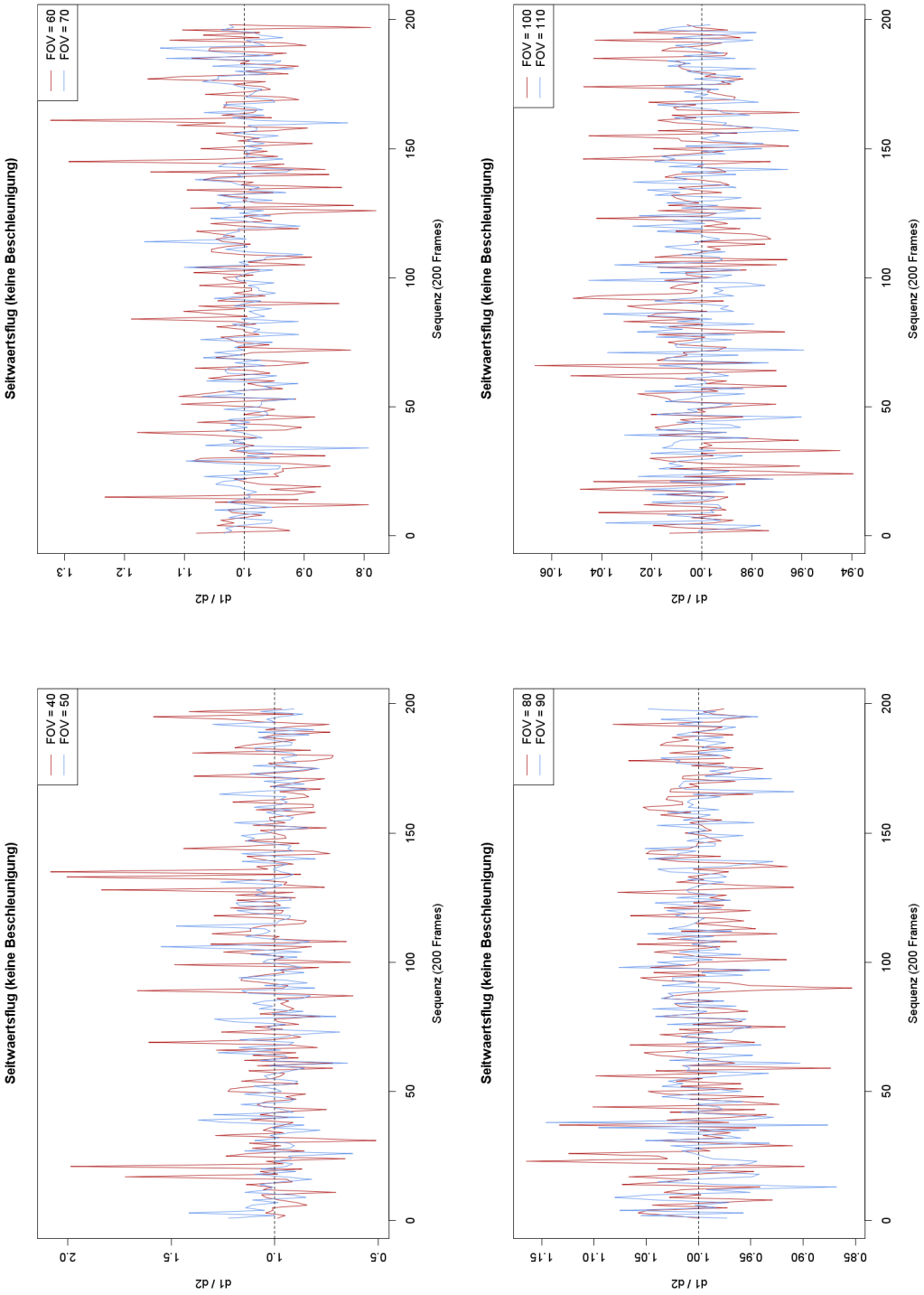
Datensätze

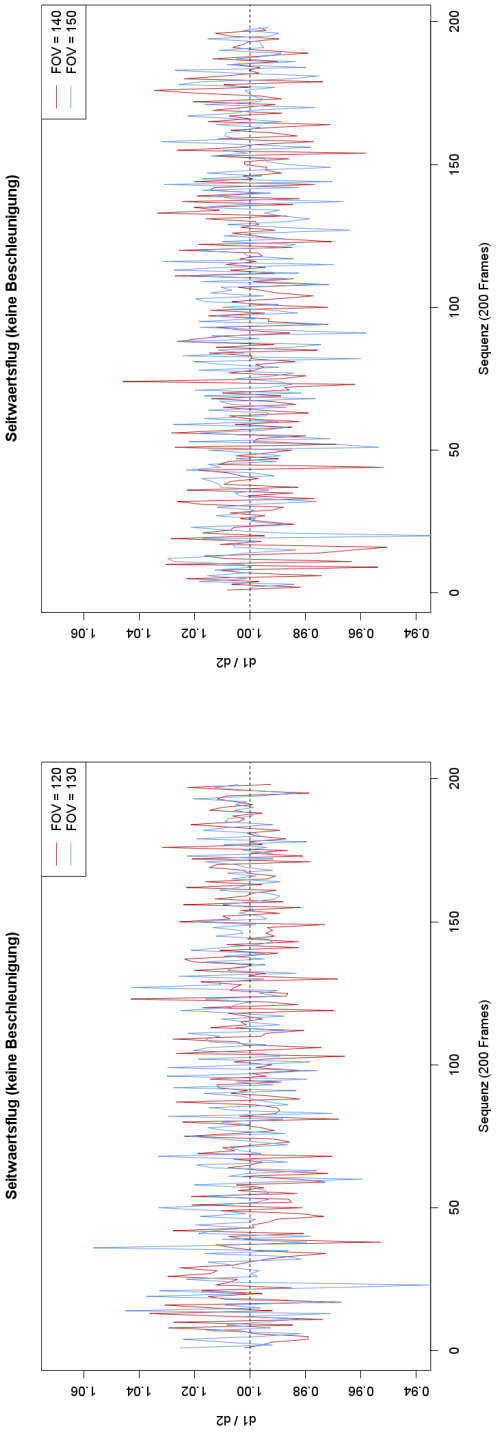
C.1 FOV-Abhängigkeit

Es werden im Folgenden die Ergebnisse des Tests der Eigengeschwindigkeitsschätzung in Abhängigkeit der Größe des FOV präsentiert. Für Erklärungen wird auf die entsprechenden Abschnitte in der Arbeit verwiesen.

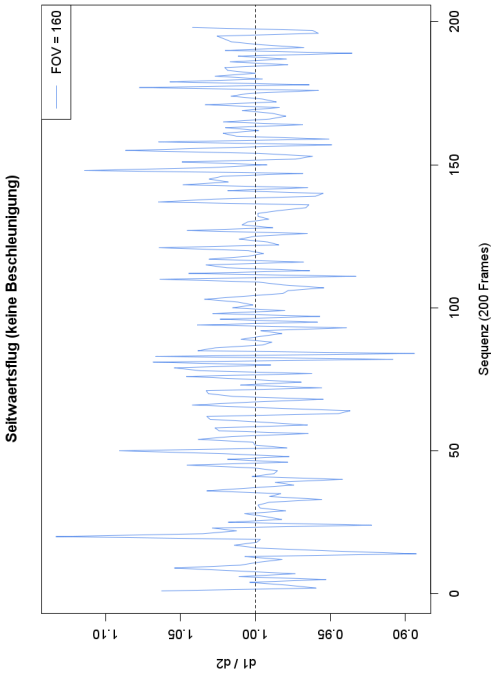

Vorwaertsflug (keine Beschleunigung)
FOV = 60
FOV = 70
FOV = 100
FOV = 110
FOV = 40
FOV = 50
FOV = 80
FOV = 90
d1 / d2
Sequenz (200 Frames)


Vorwaertsflug (keine Beschleunigung)
FOV = 120
FOV = 130
d1 / d2
Sequenz (200 Frames)
Vorwaertsflug (keine Beschleunigung)
FOV = 140
FOV = 150
d1 / d2
Sequenz (200 Frames)
Vorwaertsflug (keine Beschleunigung)
FOV = 160
d1 / d2
Sequenz (200 Frames)


Seitwaertsflug (keine Beschleunigung)
FOV = 40
FOV = 50
FOV = 60
FOV = 70
FOV = 80
FOV = 90
FOV = 100
FOV = 110
d1 / d2
Sequenz (200 Frames)


Seitwaertsflug (keine Beschleunigung)
FOV = 140
FOV = 150
d1 / d2
Sequenz (200 Frames)

Seitwaertsflug (keine Beschleunigung)
FOV = 120
FOV = 130
d1 / d2
Sequenz (200 Frames)


Seitwaertsflug (keine Beschleunigung)
FOV = 160
d1 / d2
Sequenz (200 Frames)

C.2 Daten der Freiflugaufnahmen

Es werden im Folgenden die Ergebnisse der Freiflug-Sequenzen präsentiert. Für Erklärungen wird auf die entsprechenden Abschnitte in der Arbeit verwiesen. Der Aufbau jeder Abbildung ist dabei identisch. Oben links befindet sich die Geschwindigkeitskurve, welche mit Hilfe der Positionsdaten des *Tracking System* ermittelt wurde. Oben rechts befindet sich die logarithmische Darstellung derselben. Unten links befindet sich die geschätzte Geschwindigkeitskurve, welche man nach der Methode der Tripelkorrespondenz erhält. Unten rechts ist die logarithmische Darstellung derselbigen abgebildet. Die letzte Graph stellt die während des Fluges auftretenden Eigengeschwindigkeitsänderungen dar.

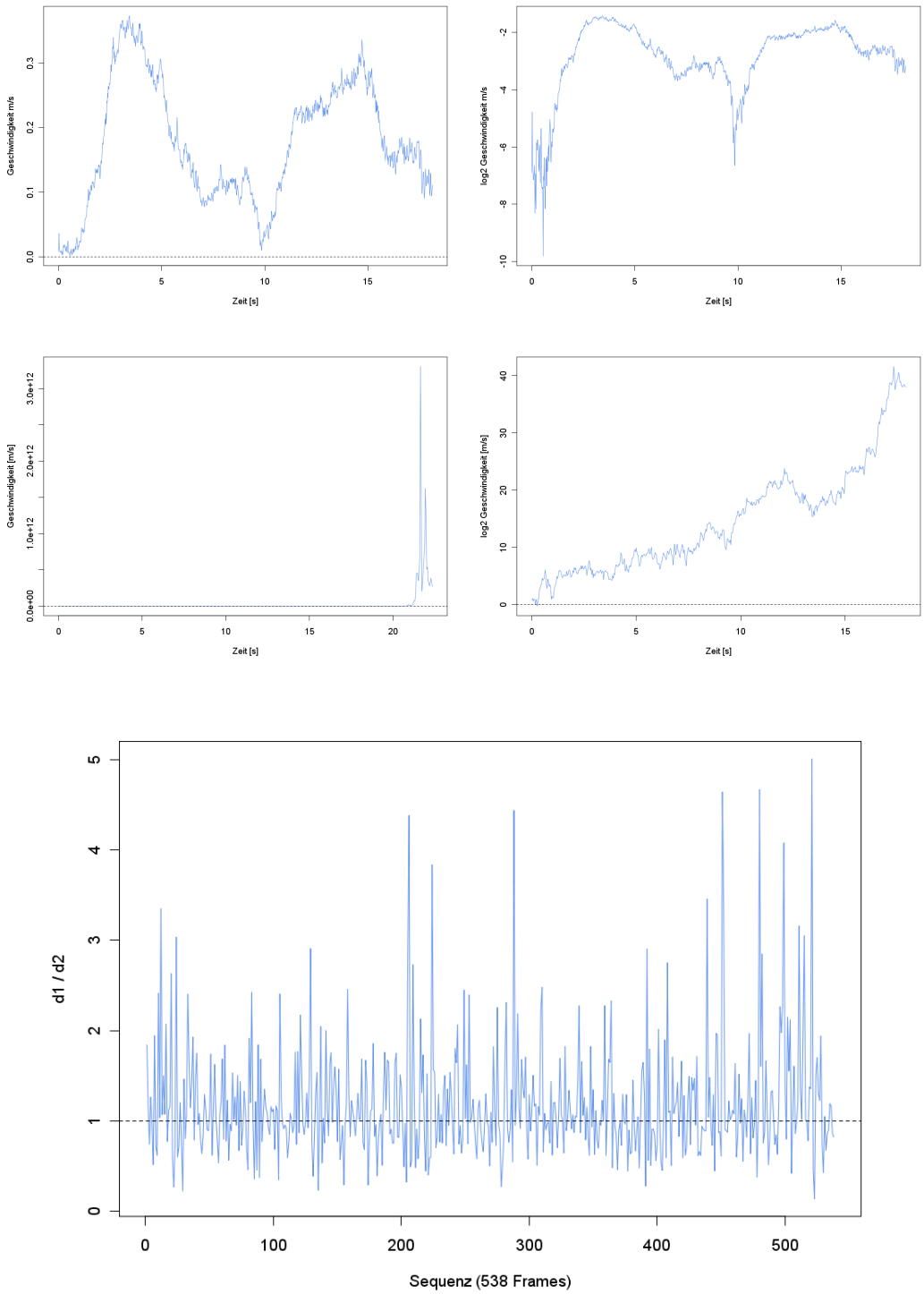


Abbildung C.1: Steig-Sink-Flug

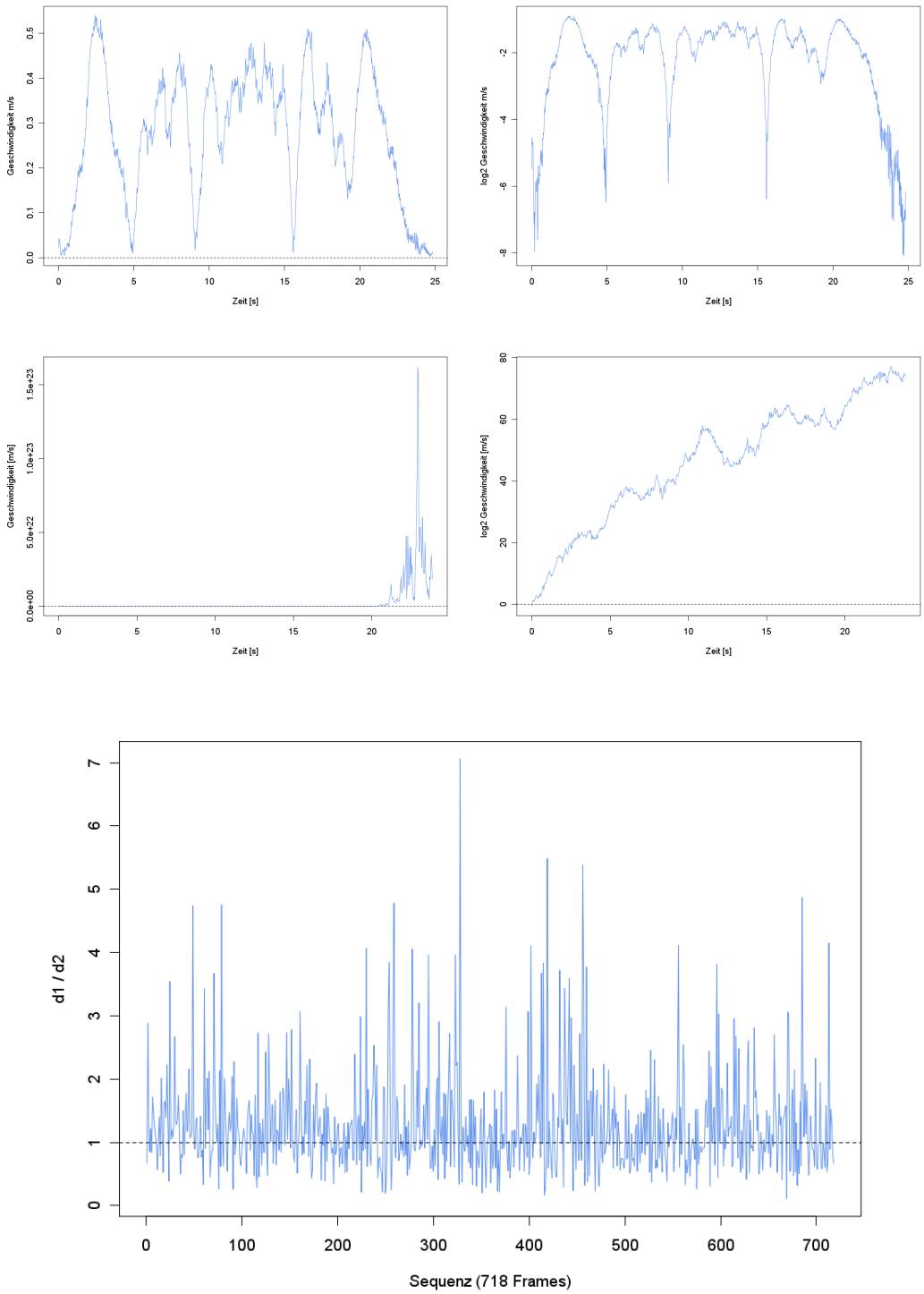


Abbildung C.2: Links-Rechts-Flug

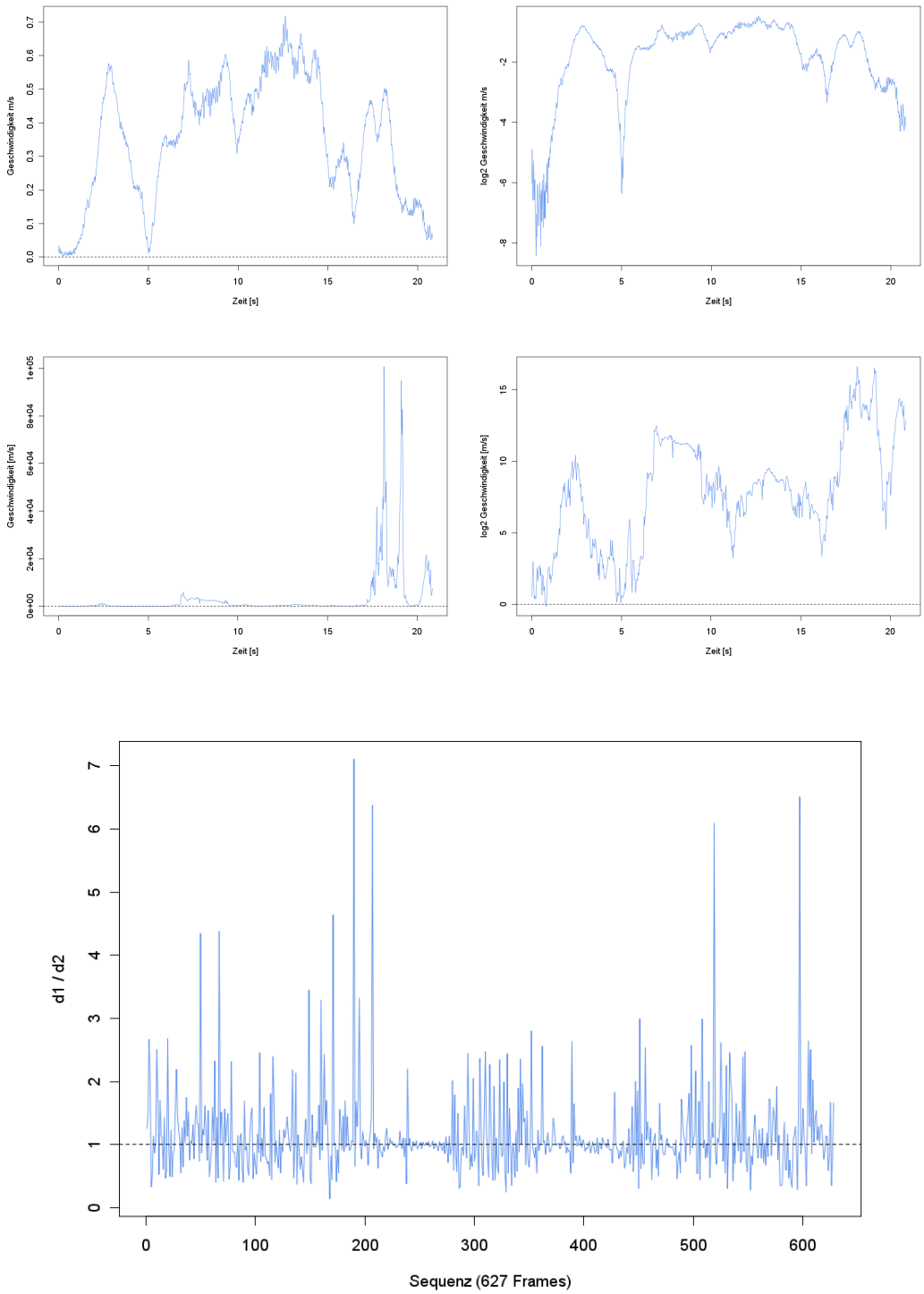


Abbildung C.3: Kreis-Flug

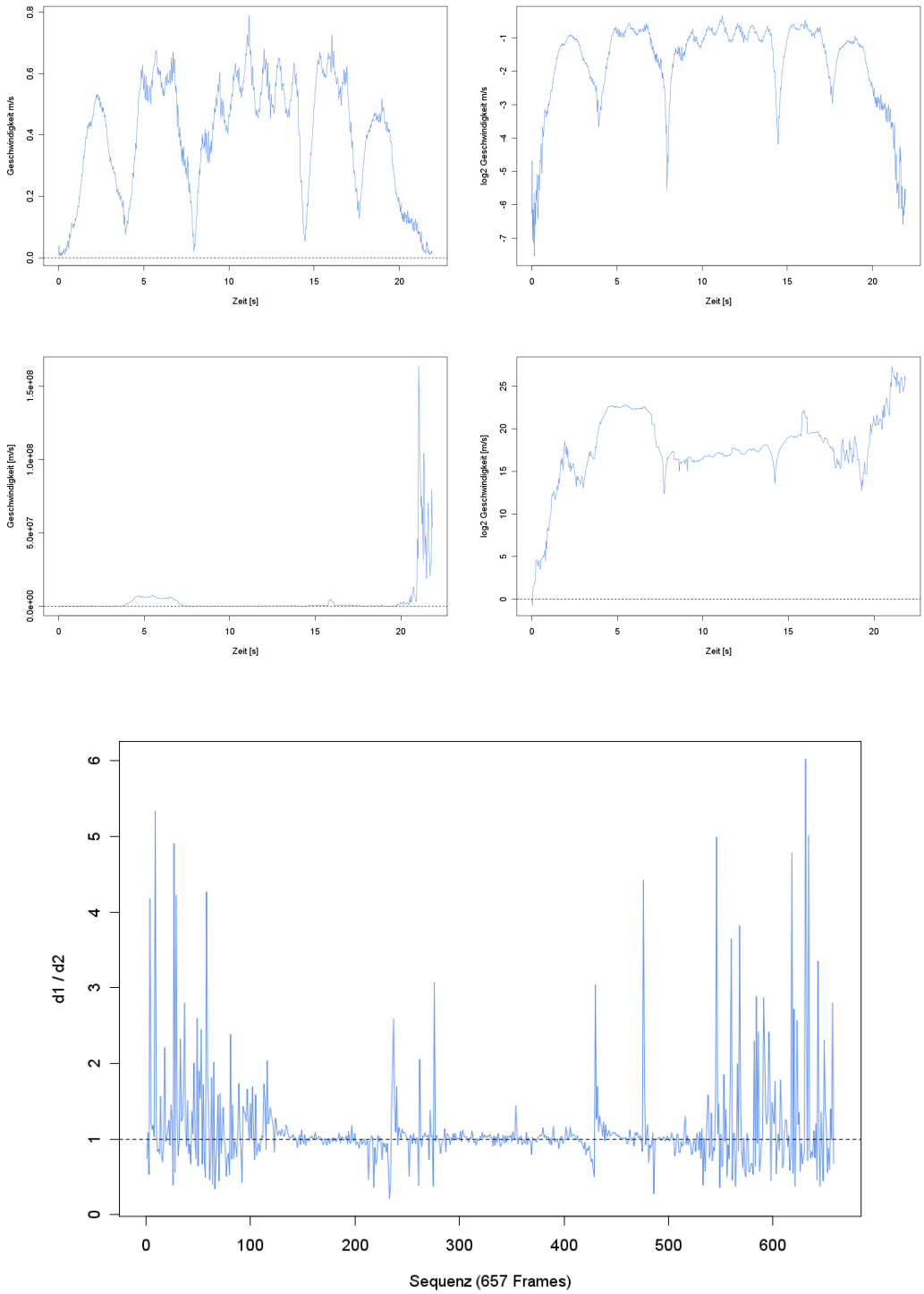


Abbildung C.4: Vor-Zurück-Flug

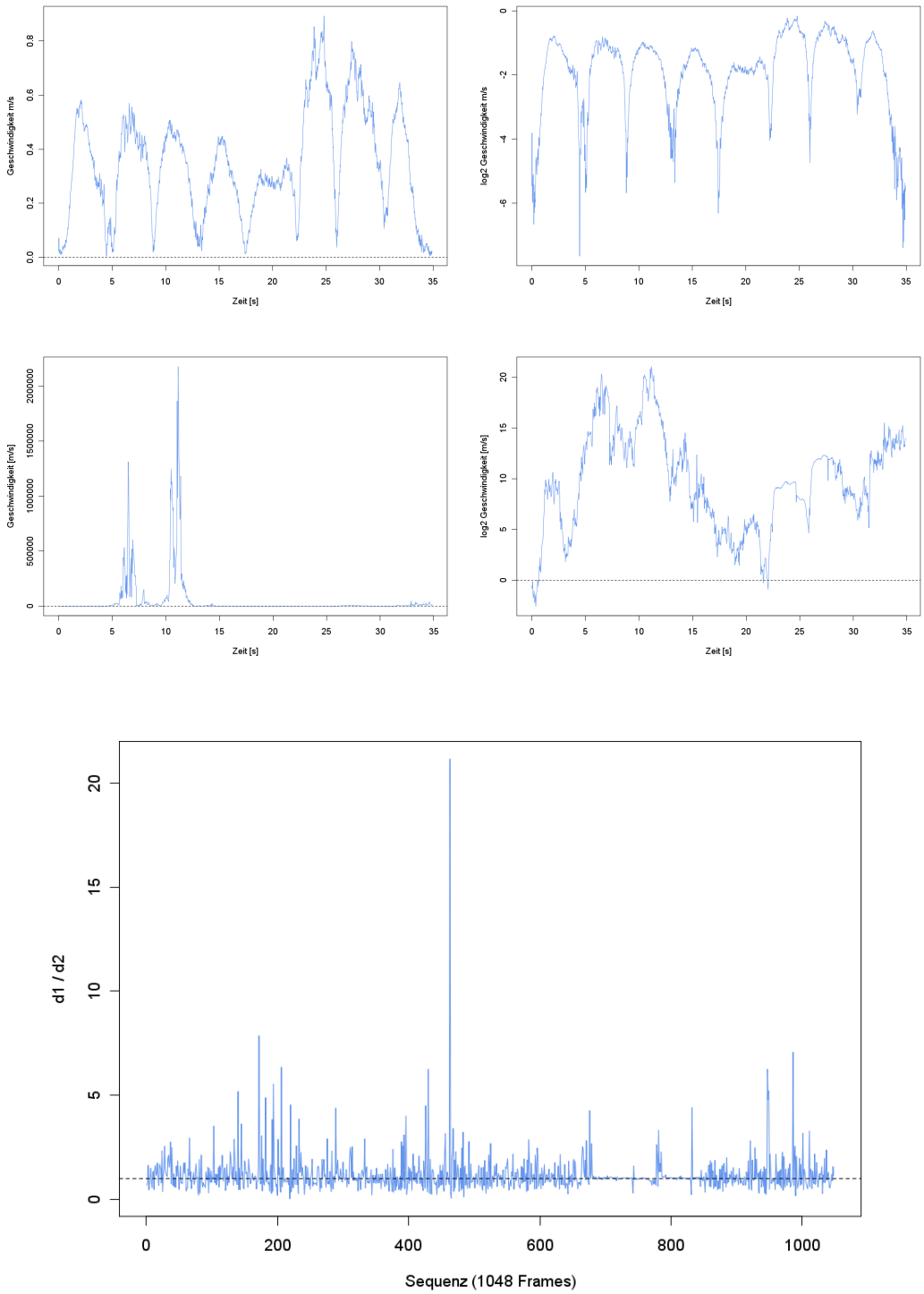


Abbildung C.5: Kombinations-Flug

C.3 mögliche Alternative zur Tripelkorrespondenzmethode

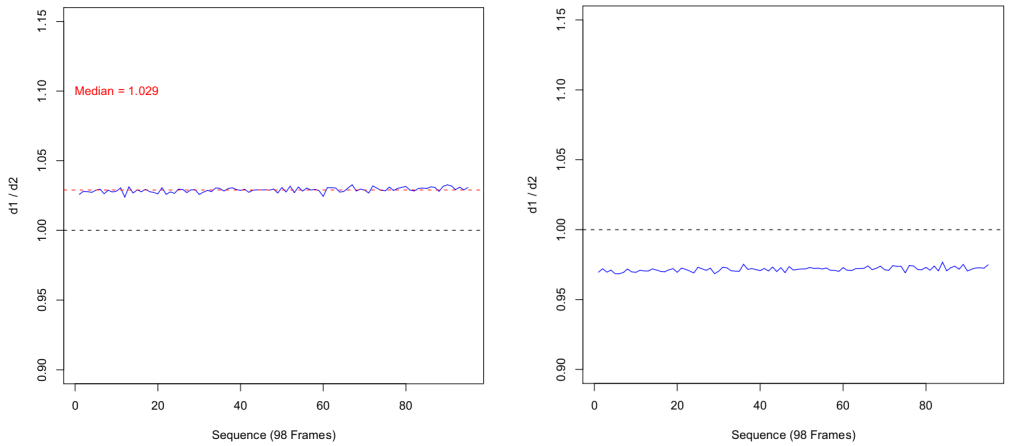


Abbildung C.6: Strikter Translationsflug ohne Beschleunigung in *Corridor-Map* mit Hilfe einer möglichen Alternative zur Tripelkorrespondenz. *(Links)* Vorwärtsflug. *(Rechts)* Rückwärtsflug.